IS-ASGD: Importance Sampling Accelerated Asynchronous SGD on Multi-Core Systems

Fei Wang\textsuperscript{a}, Xiaofeng Gao\textsuperscript{a,\textdagger}, Weichen Li\textsuperscript{c}, Guihai Chen\textsuperscript{a}, Jason Ye\textsuperscript{b}

\textsuperscript{a}Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University
\textsuperscript{b}Intel Asia-Pacific Research and Development Ltd., Shanghai, China.
\textsuperscript{c}School of Computer Science, Carnegie Mellon University

Abstract

Parallel SGD (PSGD) algorithm has been broadly used to accelerate the stochastic optimization tasks. However, its scalability is severely limited by the synchronization between threads. Asynchronous SGD (ASGD) algorithm is then proposed to increase PSGD’s scalability by allowing non-synchronized model updates. In practical, lock-free ASGD is preferable since it requires no lock operations on the concurrent update of the global model and thus achieves optimal scalability. It also maintains almost the same convergence bound if certain conditions (convexity, continuity and sparsity) are met. With the success of lock-free ASGD, researchers developed its variance reduction (VR) variants, i.e., VR-integrated lock-free ASGD to achieve superior convergence bound. We noted that the VR techniques that have been studied in lock-free ASGD context are all variance-reduced-gradient based such as SVRG, SAGA, etc. Unfortunately, the estimation of variance-reduced-gradient needs to calculate the full gradient periodically and doubles the computation cost at each iteration which decreases the scalability of ASGD to a very large extent. On the other hand, importance sampling (IS) as another elegant and practical VR technique has not been studied nor implemented in conjunction with lock-free ASGD. One important advantage of IS is that, not like variance-reduced-gradient VR algorithms, IS algorithm achieves the goal of VR through weighted sampling which does not introduce any extra on-line computation and thus preserves the original scalability of ASGD. We are thus motivated to study the application of IS in lock-free ASGD and propose our IS-ASGD algorithm to achieve a superior convergence bound while maintaining the original high scalability of ASGD. We also conduct experimental evaluations that verify the effectiveness of IS-ASGD algorithm with datasets that are popularly adopted in relative researches.

Keywords: Asynchronous SGD, Importance Sampling, Variance Reduction

\textdagger Corresponding Author

Email addresses: bomber@sjtu.edu.cn (Fei Wang), gao-xf@cs.sjtu.edu.cn (Xiaofeng Gao), weichenli@cmu.edu (Weichen Li), gchen@cs.sjtu.edu.cn (Guihai Chen), jason.ye.y@intel.com (Jason Ye)
1. Introduction

For the optimization of general finite-sum problems, stochastic gradient descent (SGD) may be the most widely adopted algorithm. It needs much less computation and empirically achieves faster convergence rate comparing to traditional gradient descent. Assume $\phi_i, i \in \{1, 2, ..., n\}$ are vector functions that maps $\mathbb{R}^d \rightarrow \mathbb{R}$, $i \sim \mathcal{D}$ means $i$ is drawn iteratively with respect to probability distribution $\mathcal{D}$. We have the following optimization problems:

$$\min_{w \in \mathbb{R}^d} F(w) := \mathbb{E}_{i \sim \mathcal{D}}(\phi_i(w) + \eta r(w))$$  \hspace{1cm} (1)

where $r(w)$ is the regularizer and $\eta$ is the regularization factor. Denote $f_i(w) = \phi_i(w) + \eta r(w)$, for stochastic gradient descent optimization, $w$ is updated as:

$$w_{t+1} \leftarrow w_t - \lambda \nabla f_i(w_t)$$  \hspace{1cm} (2)

where $i_t$ is the index of the selected sample at $t$-th iteration and $\lambda$ is the step-size. One may notice that this stochastic algorithm is a strict serial approach which is incredibly slow when facing large-scale datasets. With the emergence of ever-increasing computational capabilities brought by high concurrency of the cutting-edge hardwares, parallel SGD (PSGD) algorithms [1], [2], [3], [4], [5], [6], [7], [8], [9], etc., are proposed to accelerate the stochastic gradient descent optimization procedure by utilizing full concurrency. In order to obtain an unbiased gradient, traditional PSGD algorithms synchronize the computation results between threads at the end of each iteration. It is commonly known that its scalability is severely limited by the iteratively synchronization which causes heavy CPU waste.

To prevent such performance bottleneck caused by iteratively synchronization, two different variants of asynchronous schemes, i.e., asynchronous SGD (ASGD), have been developed depending on whether the working threads have to acquire the global lock before updating the model. These two kinds of algorithms all face the problem of stalled-model which slows down the convergence. For ASGD algorithms that perform model update in a lock-required manner e.g., [10], [11], [12], [13], [14], [15], [16], [17] and [18], the scalability may still be hampered by the frequent CPU-wasting lock contentions. Such contention of lock is especially severe on systems that have relative low data transform latency, e.g., shared-memory multi-core system. With the emergence of large-scale optimization tasks, the requirement of achieving high scalability for ASGD optimization procedure becomes even more critical. Algorithms that perform lock-free model update, i.e., lock-free ASGD such as [19], [20], [21], [22], [23], [24], [25] and [26] are preferred to further increase the scalability of ASGD on popular multi-core systems. However, due to the elimination of the lock-and-write scheme of the global model for the purpose of scalability, the worker threads compute gradients based on the model that is in an inconsistent state,
i.e., at arbitrary moment the global model is possibly the mixed value of several worker threads’ update. Indeed, such stalled and inconsistent model obviously violates the original theoretical guarantee for convergence bounding of SGD and invokes the risk of non-convergence. Fortunately, it is proved that the convergence bound remains almost as same as the SGD if certain sparsity, convexity and continuity conditions are met. With the increased scalability and proven convergence bound, lock-free ASGD algorithms quickly become indispensable in large-scale optimization tasks.

Meanwhile, using variance reduction (VR) techniques to further improve convergence bound of stochastic optimizations becomes popular and have achieved important progresses recently. VR is a kind of algorithm that is applied during the stochastic optimization process, e.g., SGD, to reduce the variance of the stochastic gradient. Most VR algorithms use historical full gradients and model snapshot to reduce the gradient variance of SGD which has been proved to improve the convergence rate\cite{27, 28, 29, 30, 31, 32, 33, 34}, i.e., a superior convergence bound. In consideration of the success of lock-free ASGD algorithms on shared-memory multi-core systems, researchers start to combine VR algorithms with lock-free ASGD algorithms such as \cite{35, 36, 37} and asynchronous stochastic coordinate descent (ASCD) \cite{38} algorithms to improve the convergence bound. These VR techniques are typically variance-reduced-gradient based, for example, SVRG \cite{39}, SAGA \cite{40}, S2CD \cite{41}, SAG \cite{42}. Reddi et al. \cite{43} conducted experimental result on SVRG, SAGA and SAG integrated lock-free ASGD in comparison with standard SGD and lock-free ASGD. In \cite{44}, the authors further compared the convergence rate improvement of the SVRG-integrated lock-free ASGD (SVRG-ASGD) between ASGD in a non-convex context. These results show that VR helps achieving superior convergence bound for lock-free ASGD and even better minimizer than SGD in some cases.

However, one critical issue of variance-reduced-gradient based VR algorithms is that they need the computation of the full gradient and double computation cost per iteration which decrease the scalability to a large extent. For large/huge-scale datasets, computing the full gradient is sometimes computational infeasible. On the other hand, as is well known that along with the variance-reduced-gradient VR algorithms, another elegant and practical VR technique, namely, importance sampling (IS) also achieves decreased gradient variance and obtain a superior convergence bound in SGD algorithms \cite{45, 46, 47, 48, 49}. While most recent researches focused on the application of the variance-reduced-gradient based VR techniques on lock-free ASGD, IS algorithms have neither been studied nor implemented in conjunction with lock-free ASGD algorithms by far. We emphasis that this missing part is worth studying. Noted that, the key advantage of IS based VR is that comparing to the variance-reduced-gradient based VR techniques which needs extra computation of full gradient and variance-reduced-gradient at each iteration, IS algorithm can be implemented with no requirement of extra on-line computation at all by using shadow threads to construct sample sequences (or completely in off-line manner) and let the computation threads iterate over the pre-constructed sequence. That is, the convergence bound of lock-free ASGD can be decreased with no
hurting of the scalability by using IS based VR algorithms which variance-reduced-gradient based VR is not able to. In theory, the computation cost of variance-reduced-gradient decreases the scalability more than a half and is often far more than that in practical.

With these considerations, we see crucial demand in developing the IS-integrated lock-free ASGD (IS-ASGD). We propose the IS-ASGD algorithm with detailed analysis of convergence guarantee and highly optimized implementation as the main contributions of this paper. We also conduct experimental results that verify the effectiveness of IS-ASGD. For clarity, the lock-free ASGD is hereinafter by default refer to as ASGD. We first briefly introduce the preliminaries and main results of variance reduction techniques in stochastic optimizations and study the missing part of these two algorithms, i.e., IS-ASGD.

2. Variance Reduction in Stochastic Optimizations

Before discussing the design of IS-ASGD, we first give a brief introduction of some necessary preliminaries and concepts of VR for stochastic optimizations based on the previous works [50], [47], [48], [45]. Recall the stochastic optimization problem in Equation (1) since the training sample is selected in a stochastic manner, \( \nabla f_i(w_t) \) varies with \( t \) despite of its expectation equals to \( \nabla F(w) \). Such variance of the gradient slows down the convergence rate of the optimization procedure to a large extent. VR techniques are thus proposed to reduced the variance which then accelerates the convergence rate. Like previous related literatures, we make the following assumptions for stochastic optimization problems we studied in this paper as shown in Equation (1) which is necessary for the bounding of convergence procedure in ASGD and VR.

- \( F \) is strongly convex with parameter \( \mu \), that is:
  \[
  \langle x - y, \nabla F(x) - \nabla F(y) \rangle \geq \mu \|x - y\|_2^2, \quad \forall x, y \in \mathbb{R}^d
  \]  
  (3)

- Each \( f_i \) is continuously differentiable and \( \nabla f_i \) has Lipschitz constant \( L_i \), i.e.,
  \[
  \|\nabla f_i(x) - f_i(y)\|_2 \leq L_i\|x - y\|, \quad \forall x, y \in \mathbb{R}^d
  \]  
  (4)

Where \( \|\cdot\|_2 \) is standard Euclidean norm.

2.1. Variance-reduced-gradient VR Algorithm

Variance-reduced-gradient based SGD uses historical full gradient and model snapshot to construct a new gradient estimator which yields lower variance than simply using stochastic gradient. A generic scheme of this kind of VR algorithms is shown in algorithm (1) [43].

Be noted that \( \{\alpha_i^t\}_{i=1} \) is computed based on historical full gradient and model, which is then used to evaluate the variance-reduced gradient \( v_t \). The update of \( \{\alpha_i^{t+1}\}_{i=0}^n \) does not necessarily to be performed at each iteration (depends on the update strategy). As can be seen that a frequent update of
Algorithm 1 Generic Variance-reduced-gradient Based SGD

1: procedure Update_Weight(T)
2: Generate \( I_T = \{i_0, \ldots, i_T\} \) from \( i_t \in \{i\}_{i=0}^{n} \) w.r.t uniform distribution.
3: for \( t = 0; t \neq T; t++ \) do
4: \( v_{t+1} \leftarrow \nabla f_{i_t}(w_t) - \nabla f_{i_t}(\alpha_{i_t}) + \frac{1}{n} \sum_{i=0}^{n} \nabla f_i(\alpha_i) \)
5: \( w_{t+1} \leftarrow w_t - \lambda v_{t+1} \quad \triangleright w \) is the global model.
6: \( \{\alpha_{i_t+1}\}_{i=0}^{n} = \text{Schedule}_\text{Update}(w_{t+1}, \{\alpha_i\}_{i=0}^{n}) \)
7: return

\( \{\alpha_{i_t+1}\}_{i=0}^{n} \) in line 4 leads to frequent re-computation of \( \frac{1}{n} \sum_{i=0}^{n} \nabla f_i(\alpha_i) \), i.e., the full-gradient. For large-scale datasets, the computation of the full-gradient could take extremely long time. A tradeoff between computational feasibility and theoretical optimality is updating \( \{\alpha_{i_t+1}\}_{i=0}^{n} \) at the beginning of each epoch instead of iteratively.

Consider applying variance-reduced-gradient based VR techniques in ASGD \cite{35,44}, denote \( t_n \) as the iterations of each epoch, only when \( t_n \gg n \) will the computation of the full-gradient has little effect on the scalability. However, in practical algorithms, \( t_n \) is typically set to \( n \) for each epoch which means the extra computation cost can not be ignored and still incurs significant scalability decrease. Meanwhile, the computation of \( \nabla f_{i_t}(\alpha_{i_t}) \) always doubles the computation cost of \( w_{t+1} \). That is, even if we omit the extra computation of the full-gradient, the scalability is still to be decreased to half. This means that the improvement of convergence rate should be sufficiently large to compensate the loss of scalability. However, according to the improved convergence bounds been proved, the actual speed up of convergence rate depends on delay parameter \( \tau \), residual \( \epsilon \), degree of conflict, etc., which is rather uncontrollable and sometimes the improvement is little. With these concerns, we see necessity in developing a new VR scheme for ASGD which improves the convergence rate while maintaining the optimal scalability. Importance sampling which needs no extra on-line computation (the computation of the sampling distribution \( P \) and sampling sequence w.r.t to \( P \) can be generated totally off-line) becomes our choice naturally.

2.2. Importance Sampling VR Algorithms

Importance sampling tries to reduce the gradient variance through an non-uniform sampling procedure instead of drawing sample uniformly as conventional stochastic optimization procedures do. For standard stochastic optimizations, the sampling probability of \( i \)-th sample at \( t \)-th iteration, namely, \( p_i^t \), equals to \( 1/n \) while in an importance sampling scheme, \( p_i^t \) is endowed with an importance factor \( I_i^t \) and thus the \( i \)-th sample is sampled at \( t \)-th iteration with a weighted probability:

\[
p_i^t = \frac{I_i^t}{n}, \quad s.t. \sum_{i=0}^{n} p_i^t = 1 \quad (5)
\]
With this non-uniform sampling procedure, to obtain an unbiased expectation, the update of $w_t$ is modified as:

$$w_{t+1} \leftarrow w_t - \frac{\lambda}{np_{i_t}} \nabla f_{i_t}(w_t)$$  \hspace{1cm} (6)

where $i_t$ is drawn i.i.d w.r.t the weighted sampling probability distribution $P^t = \{p^t_i\}_{i=0}^n$. The generic scheme of IS-integrated SGD algorithm is shown in Algorithm 2.

**Algorithm 2 Naive Importance Sampling For SGD Algorithm**

1: procedure IS-SGD($w^0$, $\{f_i\}_{i=0}^n$, $\lambda$)
2: for $t = 0; t \neq T; t++$ do
3: Construct Sampling Distribution $P_t$
4: Sample $i_t$ from $D = \{i\}_{i=0}^n$ w.r.t distribution $P_t$.
5: $w_{t+1} \leftarrow w_t + \frac{\lambda}{np_{i_t}} \nabla f_{i_t}(w_t)$

---

*Importance Sampling For Variance Reduction.* Recall the stochastic problem in Equation (1), using the analysis results from [45], we have the following lemma:

**Lemma 1.** Set $\sigma^2 = \mathbb{E}\|\nabla f_i(w_\star)\|_2^2$ where $w_\star = \arg\min_w F(w)$. Suppose that $\lambda \leq \frac{\mu}{\mu}$, with the update scheme defined in Algorithm 2, the following inequality satisfy:

$$\mathbb{E}[F(w_t) - F(w_\star)] \leq \frac{\lambda t}{\mu} \mathbb{E}V((np_{i_t})^{-1} \nabla f_{i_t}(w_t))$$ \hspace{1cm} (7)

where the variance is defined as $V((np_{i_t})^{-1} \nabla f_{i_t}(w_t)) = \mathbb{E}\|np_{i_t}^{-1} \nabla f_{i_t}(w_t) - \nabla F(w_t)\|_2^2$, and the expectation is estimated w.r.t distribution $P^t$.

It is easy to verify that the optimal sampling probability $p^t_i$ is:

$$p^t_i = \frac{\|\nabla f_i(w_t)\|_2}{\sum_{j=1}^n \|\nabla f_j(w_t)\|_2}, \forall i \in \{1, 2, ..., n\}.$$ \hspace{1cm} (8)

However, such iteratively re-estimation of $P^t$ is completely computational infeasible. The authors propose to use the upper-bound of $\|\nabla f_i(w_t)\|_2$ as an approximation. Recall the definition of Lipschitz constant, we have $\|\nabla f_i(w_t)\|_2 \leq L_i$ when $f_i(w)$ is $L_i$-Lipschitz with respect to $\|\cdot\|_2$. Thus the actual sampling probability $p_i$ is calculated according to:

$$p_i = \frac{L_i}{\sum_{j=1}^n L_j}, \forall i \in \{1, 2, ..., n\}. \hspace{1cm} (9)$$

The authors prove that using such approximated IS scheme still decreases the convergence bound, we skip repeating their analysis for clarity. It is obvious that $L_i$ of each $f_i$ can be analyzed and computed beforehand. For example,
for L1-regularized optimization problem with squared hinge loss, \( x_i \) as the \( i \)-th sample. The \( \| \cdot \|_2 \) norm of the gradient of \( f_i \) can be bounded as:

\[
\| \nabla f_i(w) \|_2 \leq 2\left[ 1 + \| x_i \|_2 / \sqrt{\lambda} \right] + \| x_i \|_2
\]

That is, we have the Lipschitz constant as \( L_i = 2(1 + \| x_i \|_2 / \sqrt{\lambda}) \| x_i \|_2 \). For L2-regularized squared hinge loss we have \( L_i = 2(1 + \| x_i \|_2 / \sqrt{\lambda}) \| x_i \|_2 + \sqrt{\lambda} \), etc. In fact, \( \| x_i \|_2 \) is the only variable in the calculation of \( L_i \) if the above mentioned convexity and Lipschitz-continuity conditions are met. With these analysis, the sampling distribution \( P \) can be constructed off-line for only once. The pseudo code of practical IS-SGD algorithm can be written as the following:

\section*{Algorithm 3 Practical Importance Sampling For SGD}

1. \textbf{procedure} IS-SGD\((w_0, \{f_i\}_{i=0}^n, \lambda)\)
2. \hspace{1em} Construct Sampling Distribution \( P \) According to Equation \ref{equation:9}
3. \hspace{1em} for \( i = 0; i \neq T; i++ \) \textbf{do}
4. \hspace{2em} Sample \( i_t \) from \( D = \{i\}_{i=0}^n \) w.r.t distribution \( P \).
5. \hspace{2em} \( w_{t+1} \leftarrow w_t + \frac{1}{\lambda p_{i_t}} \nabla f_{i_t}(w_t) \)

Noted that line 4 can also be performed by a dedicated shadow thread or completely off-line which means that there can be no extra computation at each iteration. Since ASGD is developed with the aim of spending less absolute time on training large-scale data, preserving the high scalability is greatly desired when VR algorithms are applied. As we discussed above, variance-reduced-gradient based VR algorithm needs at least doubled on-line computation cost while for importance sampling there can be theoretically no extra on-line computation. We thus consider IS as a very attractive VR technique to be combined with ASGD in the goal of improving convergence bound with high scalability.

\section{3. Importance Sampling For ASGD}

In this section we propose IS-ASGD, i.e., applying IS in ASGD as a variance reduction technique to achieve a superior convergence bound. We first give the implementation details of IS-ASGD as the following:

\subsection*{3.1. Implementation of IS-ASGD}

On the implementation of IS-ASGD, we have two schemes based on when sampling sequence is generated. The pseudo code of the generic scheme of IS-ASGD is shown in Algorithm \ref{algorithm:4}.

In the first scheme, the generation of sampling sequence is performed by a shadow thread \( t_{\text{gen}} \), i.e., for \( \text{num\_threads} \) threads and \( \text{num\_epochs} \) epoch training, \( t_{\text{gen}} \) generates \( \text{num\_epochs} \times \text{num\_threads} \) sampling sequences w.r.t \( P \). For each working thread, it switches to corresponding sequence at the start of epoch. While for the second scheme, the generation of sampling sequences for each working thread w.r.t \( P \) is performed totally off-line.
Algorithm 4 IS-ASGD Algorithm

1: procedure IS-ASGD(num\textit{threads}, num\textit{epochs})
2: Construct Sampling Distribution $P$ according to Algorithm 9
3: Spawn IS-Generation thread $t_{\text{gen}}$
4: Parallel do with num\textit{threads}
5: $\mathit{tid} \leftarrow \text{GetTid}()$
6: for $n = 0; n \neq \text{num\textit{epochs}}; n++$ do
7: for $i = 0; i \neq \text{len}; i++$ do
8: $i_t = s[n][\mathit{tid}][i]$
9: $w_{t + 1} \leftarrow w_t - \frac{\lambda}{\mathit{np}_t} \nabla f_i(w_t)$
10: return

Since we use shadow thread to generate the sampling sequence by default, if the generation of sampling sequence is faster than grad computation then the first scheme is preferred, otherwise, we will use the off-line scheme.

Optimized IS-sequence Generation. The IS-sequence generation could be a new performance bottleneck of IS-ASGD which decreases the scalability if its completion time is too long to be hided. To avoid this problem, we use the following sampling scheme to generate the IS-sequence. See Algorithm 5

Algorithm 5 FAST-IS

\textbf{procedure} IS-Sequence Generation(seq, x)
\hspace{1em} Constructing Sampling Structure according to $P$
\hspace{1em} $\mathit{cursor} \leftarrow 0$
\hspace{1em} for $i = 0; i \neq \text{len}; i++$ do
\hspace{2em} $x[i]_s \leftarrow \mathit{cursor}$
\hspace{2em} $x[i]_e \leftarrow x[i]_s + p[i]$
\hspace{2em} $\mathit{cursor} \leftarrow x[i]_e$
\hspace{1em} for $i = 0; i \neq \text{len}; i++$ do
\hspace{2em} $p \leftarrow \text{random}(0, 1)$
\hspace{2em} $\mathit{left} \leftarrow 0$
\hspace{2em} $\mathit{right} \leftarrow n - 1$
\hspace{2em} while $\mathit{left} \leq \mathit{right}$ do
\hspace{3em} $\mathit{idx} \leftarrow (\mathit{left} + \mathit{right})/2$
\hspace{3em} if $p > x[\mathit{idx}]_s$ and $p \leq x[\mathit{idx}]_e$ then
\hspace{4em} seq[i] = idx
\hspace{3em} else if $p > x[\mathit{idx}]_e$ then
\hspace{4em} $\mathit{left} \leftarrow \mathit{idx} + 1$
\hspace{3em} else
\hspace{4em} $\mathit{right} \leftarrow \mathit{idx} - 1$
\hspace{2em} return

It is easy to verify that the computational complexity of Algorithm 5 is
Meanwhile, the computation of $n$ `grads`, i.e., one epoch also has a computational complexity of $o(n/\text{num\_threads})$. Since the execution unit of `grads` computation includes several vector additions and multiplications depending on the dimensionality while IS-sequence generation only has comparison, using the first scheme will not cause working thread waiting in practical according to our observation. Based on the above discussion, we follow scheme one in our practical implementation of IS-ASGD.

4. Convergence Analysis of IS-ASGD

With the preserving of the high scalability, we analyze the convergence bound of IS-ASGD in this section. Analysis of lock-free asynchronous algorithm is not easy due to the inconsistent state of the global model, i.e., the model could be none of any result produced by a working thread. To avoid this difficulty, previous related literatures give various convergence bound analysis of ASGD with bounded variance and concurrency assumptions. Be noted that sparsity assumption is made in the original version [19] of ASGD while in some other researches which provide more general forms of convergence bound it is not required. We follow the analysis of Horia et al. [51] which models ASGD as SGD with perturbed inputs i.e., the inconsistent state of the model is treated as consistent model with noise added. This analysis scheme is more general, compact and most importantly, makes the analysis of the effect of IS relative simple in ASGD context. We first give a brief introduction of the perturbed iterate analysis of ASGD [51] which serves as the foundation of our analysis. Be noted that for the ease of analysis, we assume the optimizer, $w_*$ reaches the global optimal, i.e., $\nabla F(w_*) = 0$.

4.1. Perturbed Iterate Analysis

We follow the analysis result of Horia et al. [51], where the inconsistent state caused by lock-free update can be deemed as consistent input with stochastic noise applied in a SGD context. Follow this scheme, the update of $w_t$ can be modeled as:

$$w_{t+1} = w_t - \lambda \nabla f_i(w_t + \theta_t)$$

where $\theta_t$ is the stochastic error term. Define:

$$\hat{w}_t = w_t + \theta_t.$$  \hspace{1cm} (12)

We have:

$$\|w_{t+1} - w_*\|_2^2 = \|w_t - \lambda \nabla f_i(\hat{w}_t) - w_*\|_2^2$$

$$= \|w_t - w_*\|_2^2 - 2\lambda \langle w_t - w_*, \nabla f_i(\hat{w}_t) \rangle + \lambda^2 \|\nabla f_i(\hat{w}_t)\|_2^2$$

$$= \|w_t - w_*\|_2^2 - 2\lambda \langle \hat{w}_t - w_*, \nabla f_i(\hat{w}_t) \rangle +$$

$$\lambda^2 \|\nabla f_i(\hat{w}_t)\|_2^2 + 2\lambda \langle \hat{w}_t - w_t, \nabla f_i(\hat{w}_t) \rangle$$

\hspace{1cm} (13)
Recall the convexity assumption i.e., $F$ is strongly convex with parameter $\mu$, we have:

$$\langle \hat{w}_t - w_*, \nabla f_i(\hat{w}_t) \rangle \geq \mu \|\hat{w}_t - w_*\|^2 \geq \frac{\mu}{2} \|w_t - w_*\|^2 - \mu \|\hat{w}_t - w_t\|^2 \quad (14)$$

Further define $\epsilon_t$ as:

$$\epsilon_t = \mathbb{E} \|w_t - w_*\|^2 \quad (15)$$

According to Equation 13 and 14, we obtain

$$\epsilon_{t+1} \leq (1-\lambda \mu)\epsilon_t + \lambda^2 \mathbb{E} \|\nabla f_i(\hat{w}_t)\|^2 + 2\lambda \mu \mathbb{E} \|\hat{w}_t - w_t\|^2 + 2\lambda \mathbb{E} \langle \hat{w}_t - w_*, \nabla f_i(\hat{w}_t) \rangle$$

$$\leq R_0^t + R_1^t + R_2^t \quad (16)$$

Among the three labeled terms, notice that $R_0^t$ is a common term that exists in both serial SGD and ASGD while $R_1^t$ and $R_2^t$ are additional error terms introduced by the inconsistency of the model. It can be noted that $R_1^t$ reflects the difference between the true model and the perturbed (noise added) one, and $R_2^t$ measures the projection of such noise on the gradient of each iteration. It is easy to verify that for bounded $R_0^t$, $R_1^t$ and $R_2^t$, convergence bound can be obtained through simple algebras. With such scheme of modeling, the difficulty of the analysis of convergence bound caused by the asynchrony can be greatly simplified.

The authors first bound $\mathbb{E} \|\nabla f_i(\hat{w}_t)\|_2 \leq M$, i.e., $R_0^t \leq M^2$. Next, to bound $R_1^t$ and $R_2^t$, the concept of conflict graph is introduced as the following.

**Conflict graph.** Denote $c_i \subseteq \{j\}_{j=0}^d$ as the set of feature index of data sample $x_i$, i.e., $j \in c_i$ only if the $j$-th feature exists in $x_i$. In a conflict graph $G = \{e_{ij}, v_i\}$, $i, j \in \{0, 1, ..., n\}$, vertexes $v_i$ and $v_j$ are connected with edge $e_{ij}$ if and only if $c_i \cap c_j \neq \emptyset$.

With the above definitions, two factors that reflect the extent of conflict update are defined:

- **Delay parameter**, $\tau$, i.e., the maximum lag between when a gradient is computed and when it is used. It is easy to verify that $\tau$ can be deemed as a value linear to the number of working threads, i.e., the concurrency.

- **Conflict parameter**, $\bar{\Delta}$, which is the average degree of the conflict graph $G$, obviously, datasets with higher $\bar{\Delta}$ suffers severer extent of conflict updates and vice versa.

As can be seen that, these two parameters measure the extent of inconsistency from two aspects. $\tau$ is set as the proxy of concurrency of ASGD (Be noted that when concurrency is 1, i.e., the serial SGD, $\tau$ should be 0.) which can be controlled by the users while $\bar{\Delta}$ measures the intrinsic potentials of conflict update of dataset which is non-relevant of the algorithm’s settings. The authors prove that $R_1^t$ is bounded as:

$$R_1^t \leq \lambda^2 M^2 \left(2\tau + 8\tau^2 \frac{\bar{\Delta}}{n}\right) \quad (17)$$
and $R_t^2$ bounded as:

$$R_t^2 \leq 4\lambda M^2 \tau \frac{\Delta}{n}$$  \hspace{1cm} (18)

Plugging Equation 17 and 18 into Equation 16, we have,

$$\epsilon_{t+1} \leq (1 - \lambda\mu)\epsilon_t + \lambda^2 M^2 \left[8\tau \frac{\Delta}{n} + 4\lambda\mu\tau + 16\lambda\mu^2\tau^2 \frac{\Delta}{n}\right]$$  \hspace{1cm} (19)

Since for the SGD case which $\tau = 0$, i.e., $\xi$ term only, we have:

$$\epsilon_{t+1} \leq (1 - \lambda\mu)\epsilon_t + \lambda^2 M^2$$  \hspace{1cm} (20)

And we know that for $\xi$ (i.e., SGD), with the assumption that $\nabla F(w_*) = 0$, the convergence bound is obtained as

$$k = \log\left(\frac{2\epsilon_0}{\epsilon}\right) \frac{2M^2}{\epsilon \mu^2}$$  \hspace{1cm} (21)

where $k$ is the number of iterations to ensure $E\|w_k - w_*\|_2^2 \leq \epsilon$. By setting the stepsize $\lambda = \frac{\epsilon_0}{2M^2}$, it is easy to verify that when $\tau$ is bounded as the following

$$O\left(\min\left\{n/\Delta, M^2/\epsilon \mu^2\right\}\right)$$  \hspace{1cm} (22)

which bounds $\delta$ as an order-wise constant, then Equation 19 (ASGD) achieves nearly same recursion as in Equation 20 (SGD), i.e.,

$$k = O(1) \frac{M^2 \log\left(\frac{\epsilon_0}{\epsilon}\right)}{\epsilon \mu^2}$$  \hspace{1cm} (23)

which implies a nearly linear speedup. It is obvious that this conclusion also holds in IS-ASGD since we have bounded $\delta$ as constant which is independent of IS. Following this analysis, we consider IS takes effect on term $\xi$ in Equation 19 only. Knowing the fact that

$$M^2 := \max_{0 \leq t \leq T} E\|\nabla f_i(\hat{w}_t)\|_2^2 \leq \frac{\mu L_i^2 \epsilon_0}{\epsilon}, \quad i \in \{0, 1, \ldots, n\}$$  \hspace{1cm} (24)

where $\epsilon_0 := \max_{0 \leq t \leq T} E\|\hat{w}_t - w_*\|_2^2$. For SGD, according to Equation 21, we have a less tight $k$:

$$k = \log\left(\frac{2\epsilon_0}{\epsilon}\right) \frac{2\mu L^2 \epsilon_0}{\epsilon \mu^2}$$  \hspace{1cm} (25)

According to the convergence bound result of IS in [48], when IS procedure described in Algorithm 4 is applied in SGD, i.e., IS-SGD, the convergence bound can be reduced from quadratic dependence to linear dependence on the average conditioning as the following:

$$k = \log\left(\frac{2\epsilon_0}{\epsilon}\right) \left(\frac{2\mu \epsilon_0}{\mu \epsilon}\right)$$  \hspace{1cm} (26)
when stepsize is set as $\lambda = 1/(2 \sup L)$, where $\sup L$ is the supremum of $L_i$. By taking the additional error term caused by asynchrony, i.e., $\delta$ into account, it is easy to obtain the following convergence bound by using the same analysis trick in ASGD directly.

**Lemma 2.** For IS-ASGD algorithm that follows the scheme of Algorithm 4, by satisfying the convexity and continuity conditions in Equation 3 and Equation 4. With a proper stepsize as $\lambda = 1/(2 \sup L)$, when $\tau$ is bounded as $O\left( \min \left\{ n/\bar{\Delta}, \frac{\sup L}{\mu} \right\} \right)$. The iteration steps $k$ which is sufficient to achieve accuracy of $\epsilon$, i.e., $\mathbb{E}\|w_k - w^*\|^2 \leq \epsilon$, is defined as:

$$k = O(1) \log(\epsilon_0/\epsilon) \left( \frac{L \epsilon_0}{\mu \epsilon} \right)$$  \hspace{1cm} (27)

Obviously this bound inherits the superiority of IS-SGD over ASGD, and it shows that a nearly linear speedup is achievable for IS-ASGD comparing to IS-SGD, which is similar to the previous result\[45\] that shows SVRG-ASGD reaches nearly linear speedup of SVRG-SGD.

In brief, the key of the convergence bound analysis is the serialization of the asynchrony which divides the update scheme into two independent terms, namely, $\xi$ and $\delta$. Such independence makes the analysis much simpler, that is, the IS decrease the convergence bound of $\xi$ as the same in SGD case while the two bounded error terms caused by the asynchrony, i.e., $R_1^t$, $R_2^t$, increase the convergence bound linearly up to a constant when certain conditions are met (concurrency limitation and proper stepsize). We next evaluate the IS-ASGD algorithm to show its effectiveness in convergence acceleration and maintaining high scalability in the meanwhile.

### 5. Experimental Results

In this section we evaluate the performance of IS-ASGD through the comparison between IS-ASGD, SVRG-ASGD, ASGD and SGD. The evaluation includes the measurement on convergence bound, gradient variance and scalability. Our testbed is a shared-memory multi-core system with XeonE5-2699V4 (2 sockets and 44 cores) and 128G main memory. The datasets we choose are from LibSVM\[1\] which is popularly adopted as the evaluation set for stochastic optimization related researches. Be noted that we implement IS-ASGD based on the source code of Hogwild! algorithm\[2\] which can be deemed as the standard implementation of ASGD in C++. The source code of our proposed IS-ASGD along with demo data can be accessed from author’s git\[3\] repository. The description of evaluation dataset is shown in Table 1. With the intention to show a full spectrum performance, we choose datasets from relative sparse to extremely sparse.

---

1. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
2. http://i.stanford.edu/hazy/victor/hogwildtl-v03a-source.tar.gz
3. https://github.com/faywwww/IS-ASGD

---

12
5.1. Convergence Bound Improvement

We conduct four different concurrency settings for each dataset, \{2, 4, 8, 16\} for relative small datasets, i.e., \texttt{w8a} and \texttt{madelon} (scaled by the authors) since we observe few speedup when concurrency grows sufficiently large; \{24, 32, 38, 44\} for relative large-scale dataset \texttt{kdd2010\_algebra} due to its huge amount of data samples. Figure 1 to Figure 3 shows the experimental results on datasets \texttt{w8a}, \texttt{madelon} and \texttt{kdd2010\_algebra} between standard SGD, ASGD, SVRG-ASGD and IS-ASGD respectively. Each row shows the result of one concurrency setting, the first column shows the rooted mean square error (RMSE) on train set while the second column shows the RMSE on test set. The last column shows the gradient variance, i.e., \( \mathbb{E}\| (n p_t)_{i=1}^{t} \nabla f_i (w_t) - \nabla F (w_t) \|_2 \).

From the results, we firstly noted that IS-ASGD does achieve superior convergence bound than ASGD in all cases and concurrency settings which verify the effectiveness of IS for ASGD. On the comparison between IS-ASGD and SVRG-ASGD, we see that in \texttt{w8a}, SVRG-ASGD achieves better convergence bound than IS-ASGD in concurrency 2 and 4. However with the increasing of concurrency, the convergence bound of SVRG-ASGD deteriorates quickly while IS-ASGD is almost not affected. In concurrency 8, the convergence bound between SVRG-ASGD and IS-ASGD are very close while in concurrency 12, the convergence bound of SVRG-ASGD is inferior to IS-ASGD. In \texttt{madelon}, the result is different, IS-ASGD outperform SVRG-ASGD consistently in all concurrency settings, similar to \texttt{w8a}, the performance of SVRG-ASGD again deteriorates quickly as the concurrency grows. For \texttt{kdd2010\_algebra} which has huge dimensionality and extremely sparse, SVRG-ASGD remains the most superior in all concurrency settings with large advantage. However, as we shall see in the scalability result, this comes at the price of drastic decrease of scalability.

Meanwhile, the variance shown in the last column confirms the effectiveness of variance reduction of SVRG and IS for ASGD. It can be seen that the convergence rate is highly related to the variance, i.e., a lower variance does result in a higher convergence rate.
Figure 1: The evaluation on dataset $w8a$ with 4 concurrency settings $\tau = 2, 4, 8, 12$. The first column shows the RMSE of training set, the second column shows the RMSE of test set and the last row shows the gradient variance, i.e., $E\|[(np_{t_j})^{-1}\nabla f_t(w_t) - \nabla F(w_t)]\|_2$. 
Figure 2: The evaluation on dataset *madelon* with 4 concurrency settings $\tau = 2, 4, 8, 12$. The first column shows the RMSE of training set, the second column shows the RMSE of test set and the last row shows the gradient variance, i.e., $E\|((np_t^i)^{-1} \nabla f_t^i(w_t) - \nabla F(w_t))\|_2$. 
Figure 3: The evaluation on dataset kdd2010_algebra with 4 concurrency settings $\tau = 32, 38, 40, 44$. The first column shows the RMSE of training set, the second column shows the RMSE of test set and the last row shows the gradient variance, i.e., $E\| (np_{ti})^{-1} \nabla f_{ti}(w_t) - \nabla F(w_t) \|^2$. 
5.2. Scalability

Figure 4 shows the scalability results between the targeting algorithms. As can be seen that the scalability of SVRG-ASGD drops 79.7% in \texttt{kdda2010\_algebra} to the utmost comparing to the baseline, i.e., ASGD, due to the additional computation of full-gradient and variance-reduced gradient $v_t$ at each iteration. Particularly, for \texttt{kdda2010\_algebra}, the iteratively zero-setting of the local gradient buffer (for saving the calculation result of the variance-reduced-gradient) becomes another bottleneck since each local copy of $v_t$ has a size of approximately 154MB and can not be parallelized since the code is already in thread kernel. It is doable to get rid of the local gradient buffer by aggregating every result of each thread to the global model directly. However such approximated scheme deteriorates the convergence bound significantly since the chances of conflict upgrade actually doubled. The consequent cache misses and bandwidth contention is very severe and further hurt the scalability. Such negative impact of scalability might not be that intolerable in relative small datasets, e.g., \texttt{w8a} and \texttt{madelon} while in large/huge-scale datasets such as \texttt{kdda2010\_algebra}, the additional time cost due to the calculation of the full-gradient and variance-reduced-gradient $v_t$ is that much which makes SVRG-ASGD less attractive. On the other hand, IS-ASGD yields almost the same scalability with standard ASGD which is critical in the training of large/huge-scale datasets.

From an empirical point of view, the convergence bound of IS-ASGD is obviously superior to ASGD and outperforms SVRG-ASGD in some datasets that are not extremely large. In large-scale datasets, e.g., \texttt{kdda2010\_algebra}, the convergence bound of SVRG-ASGD is the most superior. Unfortunately performing SVRG-ASGD training on large/huge-scale datasets is extremely slow and requires large capacity of main memory (It’s zero-setting or allocation/release becomes another performance bottleneck.) due to the calculation of full-gradient periodically. Since there is no free lunch, it is up to the users to balance their requirements of better convergence bound (both SVRG-ASGD and IS-ASGD are better than ASGD) or a higher scalability (SVRG-ASGD is drastically lower).
6. Conclusion

In this paper we study the application of IS in ASGD algorithms and analysis its theoretical guarantee to achieve superior convergence bound than ASGD while maintaining high scalability. The motivation lies in the fact that conventional variance-reduced-gradient based VR algorithms such as SVRG which has been applied in ASGD (SVRG-ASGD) to achieve improved convergence bound is at the price of scalability due to its requirement of the full-gradient. With the emergence of ever-increasing large-scale dataset, the calculation of the full-gradient deteriorates the scalability of SVRG-ASGD even worse. Meanwhile, importance sampling, as another elegant and practical VR algorithm which can be implemented with no hurting of the scalability has not been studied in ASGD context yet. In considering of its advantage in maintaining the optimal scalability, we propose IS-ASGD as a practical algorithm that combines IS and ASGD. Our proposed IS-ASGD algorithm takes the advantage of IS to accelerates the convergence rate of ASGD and preserves the same scalability which is critical in large-scale training. By following a serialized form of convergence analysis, we prove that the proposed IS-ASGD achieves nearly linear speed up of IS-SGD (which is superior to SGD). The empirical evaluation conducted on datasets with relative small to extremely large dimensionality confirms the effectiveness of IS-ASGD in achieving superior convergence bound of ASGD with no hurting of the scalability at all. Comparing to SVRG-ASGD, the convergence bound of IS-ASGD is worse in some cases especially in dataset that is large-scale, e.g., kdd2010. However, as we emphasized above, the improvement of the convergence bound of SVRG-ASGD is achieved with the price of drastically decrease of scalability which is much severer in large-scale datasets. After all, we can achieve superior convergence bound by using IS in ASGD with no hurting of the scalability which is critical in large-scale datasets. We consider IS-ASGD as an effective and practical algorithm for accelerating large-scale asynchronous stochastic optimization problems.

Further researches may include the application of IS-ASGD on distributed systems which has higher latency and is bandwidth sensitive. In such systems, the calculation of the full-gradient may be even much more slow since the synchronization of the full-gradient is quite bandwidth consuming and consequently incurs high latency especially in large-scale problems. On the other hand, current IS-ASGD algorithm requires full copy of the dataset on each node otherwise the sampling is biased. The study on how local sampling, i.e., the IS is performed on part of (disjoint) the data on each node would affect the effectiveness of IS-ASGD would be interesting and worthy studying.

ACKNOWLEDGMENT

The first author thank Professor Guihai Chen for helpful suggestions, Professor Xiaofeng Gao and Weichen Li for critical readings of the original version of the paper, Jun Ye from MLT Group of Intel Asia R&D. for necessary resources and suggestions.
References

[1] A. S. M. Zinkevich, M. Weimer, L. Li., Parallelized stochastic gradient descent., in: Advances in Neural Information Processing Systems, 2010, pp. 2737–2745.

[2] M. Li, D. G. Andersen, A. J. Smola, K. Yu, Communication efficient distributed machine learning with the parameter server, in: Advances in Neural Information Processing Systems, 2014, pp. 19–27.

[3] A. J. S. J. Langford, M. Zinkevich., Slow learners are fast., in: Advances in Neural Information Processing Systems, 2009, pp. 2737–2745.

[4] R. Gemulla, E. Nijkamp, P. J. Haas, Y. Sismanis, Large-scale matrix factorization with distributed stochastic gradient descent, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 69–77.

[5] B. Recht, C. Ré, Parallel stochastic gradient algorithms for large-scale matrix completion, Mathematical Programming Computation 5 (2) (2013) 201–226.

[6] A. A. J. Duchi, M. J. Wainwright., Distributed dual averaging in networks., in: Advances in Neural Information Processing Systems, 2010, pp. 2737–2745.

[7] J. Alspector, R. Meir, B. Yuhas, A. Jayakumar, D. Lippe, A parallel gradient descent method for learning in analog vlsi neural networks, in: Advances in neural information processing systems, 1993, pp. 836–844.

[8] J. Tsitsiklis, D. Bertsekas, M. Athans, Distributed asynchronous deterministic and stochastic gradient optimization algorithms, IEEE transactions on automatic control 31 (9) (1986) 803–812.

[9] M. Vorontsov, G. Carhart, J. Ricklin, Adaptive phase-distortion correction based on parallel gradient-descent optimization, Optics letters 22 (12) (1997) 907–909.

[10] A. Agarwal, J. C. Duchi., Distributed delayed stochastic optimization., in: Advances in Neural Information Processing Systems, 2011, pp. 873–881.

[11] O. S. O. Dekel, R. Gilad-Bachrach, L. Xiao., Optimal distributed online prediction using minibatches., in: The Journal of Machine Learning Research 13., Vol. 1.

[12] A. J. S. M. Li, D. G. Andersen, K. Yu., Communication efficient distributed machine learning with the parameter server., in: Advances in Neural Information Processing Systems, 2014, pp. 19–27.
[13] D. B. A. Nedic, V. Borkar., Distributed asynchronous incremental subgradient methods., in: Inherently Parallel Algorithms in Feasibility and Optimization and their Applications, Vol. 8, 2001, pp. 318–407.

[14] A. Nedic, A. Ozdaglar., Distributed subgradient methods for multi-agent optimization., in: IEEE Transactions on Automatic Control., Vol. 54, 2009, pp. 48–61.

[15] A. N. S. S. Ram, V. V. Veeravalli., Distributed stochastic subgradient projection algorithms for convex optimization., in: Journal of Optimization Theory and Applications, Vol. 3, 2010, pp. 516–545.

[16] S. Zhang, C. Zhang, Z. You, R. Zheng, B. Xu, Asynchronous stochastic gradient descent for dnn training, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 6660–6663.

[17] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, M. Bacchiani, Asynchronous stochastic optimization for sequence training of deep neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, IEEE, 2014, pp. 5587–5591.

[18] O. Shamir, N. Srebro., On distributed stochastic optimization and learning., in: Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing, Curran Associates, Inc., 2014.

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

[20] J. T. Ben Recht, Christopher Re, V. Bittorf., Factoring nonnegative matrices with linear programs., in: Advances in Neural Information Processing Systems, 2012, pp. 1214–1222.

[21] Y.-C. J. Yong Zhuang, Wei-Sheng Chin, C.-J. Lin., A fast parallel sgd for matrix factorization in shared memory systems., in: Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 249–256.

[22] C.-J. H. S. V. Hyokun Yun, Hsiang-Fu Yu, I. Dhillon., Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion., in: arXiv preprint arXiv:1312.0193, 2013.

[23] M. I. J. John Duchi, B. McMahan., Estimation, optimization, and parallelism when data is sparse., in: Advances in Neural Information Processing Systems, 2013, pp. 2832–2840.

[24] H.-F. Y. Cho-Jui Hsieh, I. S. Dhillon., Passcode: Parallel asynchronous stochastic dual co-ordinate descent., in: arXiv preprint arXiv:1504.01365, 2015.
[25] Y. Zhuang, W.-S. Chin, Y.-C. Juan, C.-J. Lin, A fast parallel sgd for matrix factorization in shared memory systems, in: Proceedings of the 7th ACM conference on Recommender systems, ACM, 2013, pp. 249–256.

[26] F. Petroni, L. Querzoni, Gascgd: stochastic gradient descent for distributed asynchronous matrix completion via graph partitioning., in: Proc. ACM Conf. on Recommender Systems, 2014, pp. 241–248.

[27] L. Xiao, T. Zhang., A proximal stochastic gradient method with progressive variance reduction., in: SIAM Journal on Optimization., Vol. 24, 2014, pp. 2057–2075.

[28] S. De, T. Goldstein, Efficient distributed sgd with variance reduction, in: Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE, 2016, pp. 111–120.

[29] P. Richtarik, M. Takac., Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function., in: Math. Program., 2012, pp. 1–38.

[30] L. Y. Tat, A. Sidford., Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems., in: Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on., 2012, pp. 341–362.

[31] H. Xi, M. Takáč., Dual free sdca for empirical risk minimization with adaptive probabilities., in: arXiv preprint arXiv:1510.06684, 2015.

[32] Z. Q. Csiba Dominik, P. Richtárik., Stochastic dual coordinate ascent with adaptive probabilities., in: arXiv preprint arXiv:1502.08053, 2015.

[33] C. Dominik, P. Richtárik., Primal method for erm with flexible mini-batching schemes and non-convex losses., in: arXiv preprint arXiv:1506.02227, 2015.

[34] A. S. C. C. Alain G., Lamb, B. Y., Variance reduction in sgd by distributed importance sampling., in: Computer Science., Vol. 22, 2015, pp. 341–362.

[35] S.-Y. Zhao, W.-J. Li, Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 2379–2385.

[36] Y. L. Xiangru Lian, Yijun Huang, J. Liu, Asynchronous parallel stochastic gradient for nonconvex optimization, in: Advances in Neural Information Processing Systems, 2015, pp. 2737–2745.

[37] K. O. Christopher De Sa, Ce Zhang, C. Re, Taming the wild: A unified analysis of hogwild!-style algorithms, in: Advances in Neural Information Processing Systems, 2015, pp. 2656–2664.
[38] S. J. R. C. B. V. Liu, J.; Wright, S. Sridhar, An asynchronous parallel stochastic coordinate descent algorithm., in: Journal of Machine Learning Research 16., 2015, pp. 285–322.

[39] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Advances in Neural Information Processing Systems, 2013, pp. 315–323.

[40] A. Defazio, F. Bach, S. Lacoste-Julien, Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, in: Advances in Neural Information Processing Systems, 2014, pp. 1646–1654.

[41] R. P. Kökény J, Qu Z, S2cd: Semi-stochastic coordinate descent, in: NIPS Optimization in Machine Learning workshop, Curran Associates, Inc., 2014.

[42] N. L. Roux, M. Schmidt, F. R. Bach, A stochastic gradient method with an exponential convergence rate for finite training sets, in: Advances in Neural Information Processing Systems, 2012, pp. 2663–2671.

[43] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, A. J. Smola, On variance reduction in stochastic gradient descent and its asynchronous variants, in: Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2647–2655.

[44] C. Fang, Z. Lin, Parallel asynchronous stochastic variance reduction for nonconvex optimization, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 794–800.

[45] P. Zhao, T. Zhang., Stochastic optimization with importance sampling for regularized loss minimization., in: Proceedings of the 32nd International Conference on Machine Learning., 2015.

[46] T. Zhang, R. EDU, Stochastic optimization with importance sampling for regularized loss minimization.

[47] T. Strohmer, R. Vershynin., A randomized kaczmarz algorithm with exponential convergence, in: The Journal of Fourier Analysis and Applications., Vol. 2, 2009, pp. 262–278.

[48] D. Needell, R. Ward, N. Srebro, Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2014, pp. 1017–1025.

[49] D. Csiba, P. Richtárik, Importance sampling for minibatches, arXiv preprint arXiv:1602.02283.

[50] E. Moulines, F. R. Bach, Non-asymptotic analysis of stochastic approximation algorithms for machine learning, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2011, pp. 451–459.

22
[51] D. P. B. R. K. R. Horia Mania, Xinghao Pan, M. I. Jordan., Perturbed iterate analysis for asynchronous stochastic optimization., in: arXiv preprint arXiv:1507.06970, 2015.