Asynchronous Accelerated Stochastic Gradient Descent

Qi Meng,[1*] Wei Chen,[2] Jingcheng Yu,[3*] Taifeng Wang,[2] Zhi-Ming Ma,[4] Tie-Yan Liu[2]
[1] School of Mathematical Sciences, Peking University, 1501110036@pku.edu.cn
[2]Microsoft Research, {wche, taifengw, tie-yan.liu}@microsoft.com
[3]Fudan University, JingchengYu.94@gmail.com
[4]Academy of Mathematics and Systems Science, Chinese Academy of Sciences, mazm@amt.ac.cn

Abstract

Stochastic gradient descent (SGD) is a widely used optimization algorithm in machine learning. In order to accelerate the convergence of SGD, a few advanced techniques have been developed in recent years, including variance reduction, stochastic coordinate sampling, and Nesterov's acceleration method. Furthermore, in order to improve the training speed and/or leverage larger-scale training data, asynchronous parallelization of SGD has also been studied. Then, a natural question is whether these techniques can be seamlessly integrated with each other, and whether the integration has desirable theoretical guarantee on its convergence. In this paper, we provide our formal answer to this question. In particular, we consider the asynchronous parallelization of SGD, accelerated by leveraging variance reduction, coordinate sampling, and Nesterov's method. We call the new algorithm *asynchronous accelerated SGD* (AASGD). Theoretically, we proved a convergence rate of AASGD, which indicates that (i) the three acceleration methods are complementary to each other and can make their own contributions to the improvement of convergence rate; (ii) asynchronous parallelization does not hurt the convergence rate, and can achieve considerable speedup under appropriate parameter setting. Empirically, we tested AASGD on a few benchmark datasets. The experimental results verified our theoretical findings and indicated that AASGD could be a highly effective and efficient algorithm for practical use.

1 Introduction

Stochastic gradient descent (SGD) is a widely used optimization technology in many applications, due to its simplicity and good empirical performances [Bousquet and Bottou, 2008; Rakhlin *et al.*, 2012; Nemirovski *et al.*, 2009]. In this paper, we focus on the adoption of SGD to solve the following empirical risk minimization problem, whose objective is the sum of smooth and convex loss functions, i.e.

$$\min_{x\in R^d} F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{1}$$

Please note that the loss function $f_i(x)$ may take different forms in different applications. For example, suppose we have a set of training data $(a_1, b_1), \cdots, (a_n, b_n)$, where $a_i \in \mathbb{R}^d$ is an input feature vector and $b_i \in \mathbb{R}$ is its output. Then loss function f_i is usually defined as a least square function $f_i(x) = \frac{1}{2}(a_i^T x - b_i)^2$ for regression, and a logistic function $f_i(x) = \log(1 + \exp(-b_i a_i^T x))$ for classification.

SGD exploits the additive nature of the empirical risk function $F(x)$, and randomly samples an instance at each iteration to calculate the gradient and updates the model towards the direction of negative gradient. Theoretical study on SGD has revealed that it has a sublinear convergence rate, meaning that the total number of component gradient evaluations required by SGD method to find an ϵ-accurate solution is $\mathcal{O}(1/\epsilon)$. In recent years, people have been working on accelerating the convergence of SGD, by proposing new methods or leveraging existing methods in the literature of optimization. Representative acceleration methods include:

(1) *Variance Reduction (VR)* [Johnson and Zhang, 2013; Defazio *et al.*, 2014]: Besides computing the gradient of one randomly sampled instance in each iteration, the VR technique computes the full gradient in a periodical manner and uses this full gradient for multiple iterations in order to reduce the sampling variance. It can be proven that the convergence rate of SGD can become linear by using the VR technique, under the condition of strong convexity.

(2) *Stochastic Coordinate Sampling (SC)* [Nesterov, 2012; Richtárik and Takáč, 2014]: In addition to randomly sample training instances, the SC technique also performs random sampling over the feature dimensions. In particular, it randomly samples a block of coordinates, computes the partial gradients with respect to the features in the sampled block, and updates the solution only over this block. It clear that, since the computation of partial gradients are much faster, by using SC, SGD can become more efficient, especially when the number of coordinate blocks is relatively large.

(3) *Nesterov's Acceleration (NA)* [Nesterov, 2004]: The NA technique introduces one or multiple auxiliary variables, and updates the parameter according to the gradient at the

*This work was done when the author was visiting Microsoft Research Asia.

auxiliary variable. Usually, the auxiliary variable is a slide of the current parameter leaning a little towards its own momentum. It can be proven that the convergence rate of SGD is improved by using the NA technique [Hu *et al.*, 2009].

In addition to the above efforts on accelerating the convergence of SGD, there is another line of research, which speeds up the SGD algorithm by performing asynchronous parallelization. This is mainly motivated by the availability of very big training data, which makes sequential SGD very inefficient. In asynchronous parallel SGD [Recht *et al.*, 2011; Agarwal and Duchi, 2012], multiple workers compute stochastic gradients in parallel, and update the centralized model without a synchronization with each other.

Given the above literature, a natural question to ask is whether these techniques can be seamlessly integrated with each other, and whether the integration has desirable convergence property. In this paper, we will provide our formal answer to this question. In particular, we study the asynchronous parallelization of SGD, accelerated by the VR, SC, and NA techniques. We call the corresponding algorithm *asynchronous accelerated SGD* (or AASGD for short), and perform both theoretical and empirical investigations on it.

First, we prove that: 1) the three acceleration techniques are complementary to each other and can make their own contributions to the convergence rate of AASGD; 2) the asynchronous parallelization does not hurt the convergence, and one can achieve square root speed up in certain situations.

Second, we tested AASGD on a few benchmark datasets. Our experimental results show: 1) the combination of VR, SC, and NA with SGD really leads to faster convergence rate as compared to conventional SGD; 2) Thanks to the asynchronous parallelization, AASGD can achieve promising speedup when running on multiple workers. These results verified our theoretical findings and indicated that AASGD could be a effective and efficient algorithm for practical use.

The remaining paper is organized as follows. In Section 2, we introduce SGD, its acceleration methods, and other related works. In Section 3, we propose the AASGD algorithm, and present our convergence analysis. Our empirical study is given in Section 4. We conclude the paper in the last section.

2 Backgrounds

2.1 SGD and its Acceleration Methods

SGD is a widely used algorithm to solve the empirical risk minimization problem (1), whose update rule is as follows:

$$x_k = x_{k-1} - \eta_k \nabla f_{i_k}(x_{k-1}), \tag{2}$$

where f_{i_k} is a randomly sampled component of F. It has been proven that, SGD has sublinear convergence rate when a decreasing learning rate is used [Rakhlin *et al.*, 2012].

In recent years, variance reduction, stochastic coordinate sampling, and Nesterov's acceleration method have been used to accelerate the convergence of SGD. In the following paragraphs, we will give them a brief introduction.

Variance Reduction (VR): It is well known that the variance introduce by stochastic sampling prevents us from adopting a constant learning rate in SGD (which makes SGD have a slower convergence rate than full gradient descent). To remedy this problem, a few variance reduction methods have been proposed to improve SGD [Johnson and Zhang, 2013; Roux *et al.*, 2012; Shalev-Shwartz and Zhang, 2013]. In this paper, we focus on the method proposed in [Johnson and Zhang, 2013], which follows a multi-stage scheme and computes the full gradient periodically to regularize the stochastic gradients. Specifically, at the beginning of stage s, we compute the full gradient $\nabla F(\tilde{x}_s)$, and then run an inner loop ($k = 1, \cdots, K$) according to the following update formula:

$$v_k = \nabla f_{i_k}(x_{k-1}) - \nabla f_{i_k}(\tilde{x}_s) + \nabla F(\tilde{x}_s),$$
$$x_k = x_{k-1} - \eta v_k.$$

The regularized gradient v_k becomes an unbiased estimate of $\nabla F(x_{k-1})$, and its variance is much smaller than $f_{i_k}(x_{k-1})$ which is used in SGD.

Stochastic Coordinate Sampling (SC): When the data is high-dimensional, even for one sampled instance, the computation of the gradient over all the dimensions is still time-consuming. To solve this problem, people have leveraged the stochastic coordinate sampling method used in stochastic coordinate descent (SCD) [Nesterov, 2012; Richtárik and Takáč, 2014; Shalev-Shwartz and Tewari, 2011]. That is, instead of computing the gradient over all the dimensions, with SC, we only compute the partial gradients over a sampled coordinate block, as shown below:

$$x_{k,l} = x_{k-1,l} - \eta_k \nabla_l F(x_{k-1}), \quad if \quad l \in C_k,$$
$$x_{k,l} = x_{k-1,l}, \quad if \quad l \notin C_k,$$

where l is the coordinate index, $C_k (k \in \{1, ..., m\})$ is the coordinate block randomly sampled in iteration k and $\nabla_l F$ is the partial gradient of F for coordinate l.

Nesterov's Acceleration (NA): Nesterov proposed an acceleration method for gradient methods in [Nesterov, 2004]. The basic idea is to consider the momentum when updating the parameters. In particular, Nesterov introduced one or several auxiliary parameters to record the current parameter plus its latest momentum. Then, the parameter is updated by the gradient at the value of the auxiliary variable. To be specific, with NA, the update rule becomes:

$$x_{k+1} = y_k - \eta_k \nabla f_{i_k}(y_k),$$
$$y_{k+1} = x_{k+1} + \alpha(x_{k+1} - x_k).$$

2.2 Existing Convergence Analysis

First, it has been shown that the three acceleration methods introduced above can make SGD converge faster. For example, [Johnson and Zhang, 2013], considered the integration of VR with SGD, and [Lee *et al.*, 2015] integrated both VR and NA with SGD. However, to our best knowledge, it is unknown what will happen if we simultaneously combine all the three acceleration methods with SGD. Please note that, it is non-trivial to combine SC and NA with the involvement of VR, and the update rule of the combined method makes the theoretical analysis very difficult.

Second, as mentioned in the introduction, asynchronous parallelization of SGD and its convergence rate has been studied in recent years. For example, in [Recht *et al.*, 2011; Lian *et al.*, 2015; Agarwal and Duchi, 2012; Langford *et al.*, 2009; Zhang *et al.*, 2013; Avron *et al.*, 2014], the convergence rate of asynchronous SGD was studied in different settings. In [Reddi *et al.*, 2015] the asynchronous parallelization

of SVRG was investigated. However, as far as we know, the asynchronous parallelization of SGD accelerated by VR, SC, and NA has not been studied yet. This is exactly what we are going to do in this paper.

3 Asynchronous Accelerated SGD

In this section, we propose a novel algorithm, which integrates all the techniques mentioned in the introduction to improve SGD. That is, it is an asynchronous parallel SGD algorithm, and accelerated by VR, SC, and NA at the same time. For ease of reference, we call this algorithm AASGD, whose details can be found in Algorithm 1.

Assume there are P workers and one master in the computational cluster. The workers are making updates to the master in an asynchronous manner. Each worker has full access to the data, and stores a non-overlapping partition S_p $(p = 1, ..., P)$ of the data index. In addition, the features are also partitioned into m non-overlapping blocks, denoted as $\{C_1, C_2, ..., C_m\}$. .

The AASGD algorithm works in a multi-stage manner. At the beginning of stage s, each worker calculates the gradient over all its local data, i.e., $\sum_{i\in S_p} \nabla f_i(\tilde{x}_s)$, and then sends it to the master. The master averages these updates and sends the full gradient back to the workers. After that, the workers and the master update the global vector for K rounds ($k = 1, \cdots, K$) according to the following rules:

Worker: Read parameter x_k from the master, randomly sample a mini-batch of data I_k of size b without replacement and a block of coordinates C_k, compute the VR-regularized gradient u_k based on l_k over C_k, and then send u_k to the master.

Master: Update parameter x_k and auxiliary parameters y_k and z_k according to steps (5)-(7) in Algorithm 1.

The following steps in Algorithm 1 are related to the acceleration technique: (i) the full gradient calculation in step (1) is used by VR; (ii) the master's actions in steps (5)-(7) and the mini-batch sampling in step (2) are used by NA[1]; (iii) the coordinate sampling in step (3) is used by SC.

Please note that the workers run the above procedures in parallel, without synchronization with each other. As a consequence, the master will receive delayed gradients. Specifically, in Algorithm 1, the regularized gradient u_k in Step (4) is calculated with respect to a delayed parameter $x_{k-\tau_k}$. In practical, τ_k is a random variable, dependent of the computational cluster (e.g., the heterogeneity of the computational nodes and their inter-connections). It is clear that, the delay is linearly increasing with the number of workers.

4 Convergence Analysis

In this section, we analyze the convergence rate of SASGD and AASGD. To this end, we make the following assumptions, which are commonly used in previous works [Agarwal and Duchi, 2012; Zhao *et al.*, 2014; Recht *et al.*, 2011].

Assumption 1 (Smoothness): The components $\{f_i(x); i \in [n]\}$ of $F(x)$ are differentiable and have Lipschitz continuous partial gradients. That is to say, there exists a positive constant L_{max}, such that $\forall i \in [n], j \in [d]$, for $\forall x, y \in \mathbb{R}^d$ with $x_j \neq y_j$, we have

$$\|\nabla_j f_i(x) - \nabla_j f_i(y)\| \leq L_{max}\|x_j - y_j\|.^2$$

Therefore, the components $f_i(x)$ have Lipschitz continuous gradients, as shown below:

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_{res}\|x - y\|, \forall i.$$

Assumption 2 (Convexity): $F(x)$ is strongly convex with convexity parameter μ, i.e.,

$$F(y) \geq F(x) + \nabla F(x)^T(y - x) + \frac{\mu}{2}\|y - x\|^2, \forall x, y \in \mathbb{R}^d.$$

Assumption 3 (Sparseness): The sparseness in mini-batch setting is defined as follows[3]:

$$\Delta = \max_{j\in[d]} \frac{\sum_{v=1}^{C_n^b} \mathbb{I}_{[\exists i \in I_v, s.t. x_{i,j}\neq 0]}}{C_n^b},$$

where $\mathbb{I}$ denotes the indicator function and I_v denotes the mini-batch with index v which is a subset of $[n]$. Intuitively, the sparseness measures the maximal frequency for arbitrary coordinate appears in the mini-batch inputs. It is commonly assumed that the sparseness is upper bounded.

Assumption 4 (Bounded Delay): The delays $\tau_1, \tau_2, ...$ are independent random variables, and $\tau_k \leq \tau$ for all k.

Based on above assumptions, we proved two theorems. The first one is for the sequential version of SGD accelerated by VR, SC, and NA (which we call SASGD sequential accelerated SGD), and the second one is regarding its asynchronous parallelization. In order to distinguish the parameters for SASGD and AASGD, we use subscript 0 to denote

Algorithm 1 Asynchronous Accelerated SGD (AASGD)

Require: initial vector $[\tilde{x}_0, \eta, \beta, b, K, S]$
Ensure: $\tilde{x}_S$
  for $s = 1, 2, ..., S$ **do**
    $\tilde{x} = \tilde{x}_{s-1}$; $x_0 = y_0 = z_0 = \tilde{x}$
    For *local worker* p: calculate $\nabla F_p(x_0) = \sum_{i\in S_p} \nabla f_i(x_0)$ and send it to the center.
    (1) For *master*: calculate $\nabla F(x_0) = \frac{1}{P}\sum_{p=1}^{P} \nabla F_p(x_0)$
    for $k = 1, ..., K$ **do**
      For *local worker* p: read x_k and calculate u_k as below:
      (2) Randomly select a mini-batch of data I_k with size b,
      (3) Randomly select a coordinate block C_{j_k},
      (4) Calculate $u_{k+1,l} = \nabla_l f_{I_k}(x_{k-\tau_k}) - \nabla_l f_{I_k}(x_0) + \nabla_l F(x_0)$ if $l \in C_{j_k}$; $u_{k+1,l} = 0$ if $l \notin C_{j_k}$ and send it to the master.
      For *master*: update the parameters as below:
      (5) $y_{k,l} = x_{k,l} - \eta u_{k,l}$ if $l \in C_{j_k}$; $y_{k,l} = y_{k-1,l}$ if $l \notin C_{j_k}$.
      (6) $z_k = z_{k-1} - \alpha u_k$
      (7) $x_{k+1} = (1 - \beta)y_k + \beta z_k$
    end for
    $\tilde{x}_s = \frac{1}{K}\sum_{k=1}^{K} x_k^s$
  end for

[1] It has been proved in [Nitanda, 2014], the mini-batch strategy is necessary for the faster convergence of SGD with NA.

[2] In this paper, if there is no specification, $\|\cdot\|$ is the L_2-norm.

[3] In this paper, for simplicity and without confusion, we omit the mini-batch size b in sparseness.

parameters for SASGD (and parameters without subscript for AASGD).

Theorem 4.1 *Under Assumptions 1-3, if we set step size* $\eta_0 \leq \frac{1}{L_{max}}$, $\beta_0 = \frac{\eta_0}{\eta_0 + m\alpha}$, *and K sufficiently large so that*

$$\rho_0 = \frac{1}{1-\alpha_0 A}\left(\frac{m\alpha_0}{\eta_0 K} + \frac{m}{\mu\alpha_0 K} + \alpha_0 A\right) < 1,$$

where $0 < \alpha_0 < \frac{1}{A}$ *and* $A = \frac{4L_{res}(n-b)}{b(n-1)}$, *then SASGD algorithm has geometric convergence in expectation:*

$$\mathbb{E}F(\tilde{x}_s) - F(x^*) \leq \rho_0^s[F(\tilde{x}_0) - F(x^*)].$$

Proof: First, we take expectation of the VR-regularized gradients v_{k+1} with respect to I_k, j_k as below,

$$\mathbb{E}_{I_k,j_k} v_{k+1} = \frac{1}{m}\mathbb{E}_{I_k}\bar{v}_{k+1} = \frac{1}{m}\nabla F(x_{k+1}).$$

Then, for the auxiliary parameter z, we have:

$$\begin{aligned}
&\mathbb{E}_{I_k,j_k}\|z_{k+1} - x^*\|^2 \\
&= \|z_k - x^*\|^2 - 2\alpha_0\mathbb{E}_{I_k,j_k} v_{k+1}^T(z_k - x^*) + \alpha_0^2\mathbb{E}_{I_k,j_k}\|v_{k+1}\|^2 \\
&= \|z_k - x^*\|^2 - \frac{2}{m}\alpha_0\nabla F(x_{k+1})^T(z_k - x^*) \\
&\quad + \alpha_0^2\mathbb{E}_{I_k,j_k}\|v_{k+1}\|^2.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\nabla F(x_{k+1})^T(z_k - x^*) &= \frac{m\alpha_0}{2}\mathbb{E}_{I_k,j_k}\|v_{k+1}\|^2 \\
&\quad + \frac{m}{2\alpha_0}\|z_k - x^*\|^2 - \frac{m}{2\alpha_0}\mathbb{E}_{I_k,j_k}\|z_{k+1} - x^*\|^2. \quad (3)
\end{aligned}$$

According to the smoothness of partial gradients, we have the following:

$$\begin{aligned}
&\mathbb{E}_{I_k,j_k} F(y_{k+1}) \\
&\leq F(x_{k+1}) - \eta_0\mathbb{E}_{I_k,j_k}\nabla F(x_{k+1})^T v_{k+1} \quad (4) \\
&\quad + \frac{L_{max}\eta_0^2}{2}\mathbb{E}_{I_k,j_k}\|v_{k+1}\|^2 \\
&= F(x_{k+1}) - \frac{\eta_0}{m}\|\nabla F(x_{k+1})\|^2 + \frac{L_{max}\eta_0^2}{2}\mathbb{E}_{I_k,j_k}\|v_{k+1}\|^2.
\end{aligned}$$

Thus, we have the following inequality:

$$\begin{aligned}
0 \leq &\frac{m\alpha_0}{\eta_0}(F(x_{k+1}) - \mathbb{E}_{I_k,j_k}F(y_{k+1})) \\
&- \alpha_0\|\nabla F(x_{k+1})\|^2 + \frac{L_{max}m\eta_0\alpha_0}{2}\mathbb{E}_{I_k,j_k}\|v_{k+1}\|^2.
\end{aligned}$$

After that, by following the proof of Theorem 3 in [Lee *et al.*, 2015], we proved this theorem. □

Based on Theorem 4.1, we have the following corollary for the complexity of SASGD, which is defined as how many partial gradients we need to calculate so as to guarantee the optimization error less than ϵ.

Corollary 4.2 *Under Assumptions 1-3, and the parameter setups in Theorem 4.1, if we set* $\alpha_0 = \min\{\frac{1}{3A}, \frac{1}{\sqrt{L_{max}\mu}}\}$, *and the mini-batch size* $b = \frac{12n\sqrt{\kappa_1}}{(n-1)+12\sqrt{\kappa_1}}$, *then the SASGD can achieve its optimal overall complexity,which is*

$$\mathcal{O}((nm + \frac{nm\sqrt{\kappa_2}\sqrt{\kappa_1}}{n+\sqrt{\kappa_1}})\log\frac{1}{\epsilon}),$$

where $\kappa_1 = \frac{L_{res}^2}{L_{max}\mu}$ *and* $\kappa_2 = \frac{L_{max}}{\mu}$.

Proof: For SASGD, by setting $\alpha_0 = \min\{\frac{1}{3A}, \frac{1}{\sqrt{L_{max}\mu}}\}$, $\eta_0 = \frac{1}{L_{max}}$, we can get $\rho_0 < \frac{1}{2/3}\left(\frac{m\sqrt{\kappa_2}}{K} + \frac{m}{\mu\alpha_0 K} + \frac{1}{3}\right)$. Choosing mini-batch size $b = \frac{12n\sqrt{\kappa_1}}{(n-1)+12\sqrt{\kappa_1}}$, then we can get $\frac{1}{3A} = \frac{1}{\sqrt{L_{max}\mu}}$. Putting $\alpha_0 = \frac{1}{\sqrt{L_{max}\mu}}$ in ρ_0, then $\rho_0 = \frac{1}{2/3}\left(\frac{2m\sqrt{\kappa_2}}{K} + \frac{1}{3}\right)$. Then we can choose $K = \mathcal{O}(m\sqrt{\kappa_2})$. Then the overall complexity (number of partial gradients evaluation) is $\mathcal{O}(nm + bm\sqrt{\kappa_2})$. Putting $b = \frac{12n\sqrt{\kappa_1}}{(n-1)+12\sqrt{\kappa_1}}$ in $\mathcal{O}(nm + bm\sqrt{\kappa_2})$ can get the result. □

Remarks: (1) The convergence rate of SASGD in Theorem 4.1 is linear, much faster than the sublinear convergence rate of SGD. This clearly demonstrates the advantages of adopting the acceleration techniques.

(2) From the complexity of SASGD in Theorem 4.1 and Corollary 4.2, we can observe the complementary contributions of three acceleration methods. As we know, the complexity of SGD with a decreasing step size $\eta_k = 1/\mu k$ is $\mathcal{O}(1/\epsilon\mu)$[Rakhlin *et al.*, 2012]. Comparing the complexity of SASGD to that of SGD, i) VR method improves the complexity from $\mathcal{O}(1/\epsilon\mu)$ to $\mathcal{O}\left((nm + mL_{res}/\mu)\log\frac{1}{\epsilon}\right)$; ii) NA method improves the coefficient in the complexity from $(nm + mL_{res}/\mu)$ to $\left(nm + \frac{nmL_{res}/\mu}{n+\sqrt{L_{res}/\mu}}\right)$; iii) SCD method improve the coefficient from $\frac{nmL_{res}/\mu}{n+\sqrt{L_{res}/\mu}}$ to $\frac{nmL_{res}/\mu}{n+\sqrt{L_{res}/\mu}\sqrt{L_{res}/L_{max}}}$, which is faster since $L_{res} \geq L_{max}$.

Next, the following Theorem states the convergence rate after introducing asynchronous parallelization to SASGD.

Theorem 4.3 *Under Assumptions 1-4, and* Δ *satisfies* $\sqrt{\Delta} < \min\{\mu, \frac{1}{4}\}$, *if we set the step size* $\eta \leq \frac{1}{2L_{max}\tau}$, $\beta = \frac{\eta}{m\alpha+\eta}$, *and K is sufficiently large so that*

$$\rho = \frac{1}{1-\alpha AD}\left(\frac{m\alpha}{\eta L_{max}K} + \frac{m}{\mu\alpha K} + \alpha AD\right) < 1,$$

where $0 < \alpha < \frac{1}{AD}$, $A = \frac{4L_{res}(n-b)}{b(n-1)}$, $D = 2(1 - \frac{\sqrt{\Delta}}{2})$, *then AASGD algorithm has geometric convergence in expectation:*

$$\mathbb{E}F(\tilde{x}_s) - F(x^*) \leq \rho^s[F(\tilde{x}_0) - F(x^*)].$$

Proof: At stage s, let v_{k+1} and u_{k+1} be two vectors whose l-th dimension defined as follows:
$v_{k+1,l} = \nabla_l f_{I_k}(x_{k+1}) - \nabla_l f_{I_k}(\tilde{x}_{s-1}) + \nabla_l F(\tilde{x}_{s-1})$ if $l \in C_{j_k}$; otherwise, $v_{k+1,l} = 0$.
$u_{k+1,l} = \nabla_l f_{I_k}(x_{k+1-\tau_k}) - \nabla_l f_{I_k}(\tilde{x}_{s-1}) + \nabla_l F(\tilde{x}_{s-1})$ if $l \in C_{j_k}$;otherwise,$u_{k+1,l} = 0$.
Let $\bar{v}_{k+1} = \nabla f_{I_k}(x_{k+1}) - \nabla f_{I_k}(\tilde{x}_{s-1}) + \nabla F(\tilde{x}_{s-1})$
and $\bar{u}_{k+1} = \nabla f_{I_k}(x_{k+1-\tau_k}) - \nabla f_{I_k}(\tilde{x}_{s-1}) + \nabla F(\tilde{x}_{s-1})$.
According to Inequality (6) and Inequality (7) in paper [Lee *et al.*, 2015], we have:

$$\begin{aligned}
&F(x_{k+1}) - F(x^*) \\
&\leq \frac{1-\beta}{\beta}(F(y_k) - F(x_{k+1})) - \frac{1-\beta}{\beta}\frac{\mu}{2}\|y_k - x_{k+1}\|^2 \\
&\quad + \nabla F(x_{k+1})^T(z_k - x^*) - \frac{\mu}{2}\|x_{k+1} - x^*\|^2.
\end{aligned}$$

Let $\mathbb{E}_{I_k,j_k}$ be the conditional expectations conditioned on $\{x_s, y_s, z_s\}_{s=0,1,\ldots,k}$ in stage s. Since $z_{k+1} - z_k = -\alpha u_{k+1}$, and $\mathbb{E}_{I_k,j_k} u_{k+1} = \frac{1}{m}\nabla F(x_{k+1-\tau_k})$, we can get:

$$\begin{aligned}
&\nabla F(x_{k+1})^T(z_k - x^*) \\
&= \mathbb{E}_{I_k}(\bar{v}_{k+1} - \bar{u}_{k+1})^T(z_k - x^*) + \frac{m\alpha}{2}\mathbb{E}_{I_k,j_k}\|u_{k+1}\|^2 \\
&\quad + \frac{m}{2\alpha}\|z_k - x^*\|^2 - \frac{m}{2\alpha}\mathbb{E}_{I_k,j_k}\|z_{k+1} - x^*\|^2. \qquad (5)
\end{aligned}$$

According to the smoothness of F, we have:

$$\begin{aligned}
&\mathbb{E}_{I_k,j_k} F(y_{k+1}) \\
&\le F(x_{k+1}) - \frac{\eta}{m}\|\nabla F(x_{k+1})\|^2 + \frac{L_{max}\eta^2}{2}\mathbb{E}_{I_k,j_k}\|u_{k+1}\|^2 \\
&\quad + \frac{\eta}{m}\mathbb{E}_{I_k}\nabla F(x_{k+1})^T(\bar{v}_{k+1} - \bar{u}_{k+1}). \qquad (6)
\end{aligned}$$

The difference between AASGD and SASGD mainly lies in the communication delay τ. To characterize its influence on the convergence rate, we take the following three steps.

Step 1: Due to the delay, Equation (5) has an extra term $\mathbb{E}_{I_k}(\bar{v}_{k+1} - \bar{u}_{k+1})^T(z_k - x^*)$ as compared with Equation (3). According to the fact that $z_k = x_{k+1} - \frac{1-\beta}{\beta}(y_k - x_{k+1})$ and the Cauthy-Schwarz inequality, we have

$$\begin{aligned}
&m\mathbb{E}_{I_k,j_k}(v_{k+1} - u_{k+1})^T(z_k - x^*) \\
&\le \mathbb{E}_{I_k}\|\bar{v}_{k+1} - \bar{u}_{k+1}\|\|x_{k+1} - x^*\|_{I_k} \\
&\quad + \frac{1-\beta}{\beta}\mathbb{E}_{I_k}\|\bar{v}_{k+1} - \bar{u}_{k+1}\|\|x_{k+1} - y_k\|_{I_k},
\end{aligned}$$

where $\|x\|_{I_k}$ denote the norm of x with respect to the union of non-zero coordinates of $a_i \in I_k$, i.e., $\|x\|_{I_k} = (\sum_{l=1}^d |x_l|^2 \mathbb{I}_{[\exists a_i \in I_k s.t. a_{il} \neq 0]})^{\frac{1}{2}}$. Using the AM-GM inequality, the sparseness condition and x_{k+1} is independent of I_k, we can obtain:

$$\begin{aligned}
&\mathbb{E}_{I_k}\|\bar{v}_{k+1} - \bar{u}_{k+1}\|\|x_{k+1} - x^*\|_{I_k} \\
&\le \frac{\sqrt{\Delta}\mathbb{E}_{I_k}\|\bar{v}_{k+1} - \bar{u}_{k+1}\|^2}{2} + \frac{\mathbb{E}_{I_k}\|x_{k+1} - x^*\|_{I_k}^2}{2\sqrt{\Delta}} \\
&\le \sqrt{\Delta}\mathbb{E}_{I_k}\|\bar{v}_{k+1}\|^2 + \sqrt{\Delta}\mathbb{E}_{I_k}\|\bar{u}_{k+1}\|^2 \\
&\quad + \frac{\sqrt{\Delta}\mathbb{E}_{I_k}\|x_{k+1} - x^*\|^2}{2}.
\end{aligned}$$

Since we also have $E_{I_k,j_k}\|u_{k+1}\|^2 = E_{I_k}\|\bar{u}_{k+1}\|^2$, and $E_{I_k,j_k}\|v_{k+1}\|^2 = E_{I_k}\|\bar{u}_{k+1}\|^2$, Equation (5) can be re-written as

$$\begin{aligned}
&\nabla F(x_{k+1})^T(z_k - x^*) \le -\frac{m}{2\alpha}\mathbb{E}_{I_k,j_k}\|z_{k+1} - x^*\|^2 \\
&+ \frac{m}{2\alpha}\|z_k - x^*\|^2 + \sqrt{\Delta}\frac{1}{\beta}\mathbb{E}_{I_k}\|\bar{v}_{k+1}\|^2 + \frac{\sqrt{\Delta}}{2}\|x_{k+1} - x^*\|^2 \\
&+ \frac{\sqrt{\Delta}(1-\beta)}{2\beta}\|x_{k+1} - y_k\|^2 \\
&+ (\frac{\alpha}{2} + \sqrt{\Delta}\frac{1}{\beta})\mathbb{E}_{I_k}\|\bar{u}_{k+1}\|^2. \qquad (7)
\end{aligned}$$

Step 2: Due to the delay, Inequality (6) has an extra term $\frac{\eta}{m}\mathbb{E}_{I_k}\nabla F(x_{k+1})^T(\bar{v}_{k+1} - \bar{u}_{k+1})$ as compared with Equation (4). We bound this term using the same technique as *Step 1*.

$$\begin{aligned}
\frac{\eta}{m}\mathbb{E}_{I_k}\nabla F(x_{k+1})^T(\bar{v}_{k+1} - \bar{u}_{k+1}) &\le \frac{\eta\sqrt{\Delta}}{2m}\|\nabla F(x_{k+1})\|^2 \\
&+ \frac{\eta\sqrt{\Delta}}{m}\mathbb{E}_{I_k}\|\bar{v}_{k+1}\|^2 + \frac{\eta\sqrt{\Delta}}{m}\mathbb{E}_{I_k}\|\bar{u}_{k+1}\|^2.
\end{aligned}$$

Then Inequality (6) can be re-written as

$$\begin{aligned}
&0 \le (F(x_{k+1}) - \mathbb{E}_{I_k,j_k}F(y_{k+1})) \\
&- \left(\frac{\eta}{m} - \frac{\eta\sqrt{\Delta}}{2m}\right)\|\nabla F(x_{k+1})\|^2 \\
&+ \left(\frac{L_{max}\eta^2}{2m} + \frac{\eta\sqrt{\Delta}}{m}\right)\mathbb{E}_{I_k}\|\bar{u}_{k+1}\|^2 + \frac{\eta\sqrt{\Delta}}{m}\mathbb{E}_{I_k}\|\bar{v}_{k+1}\|^2. \qquad (8)
\end{aligned}$$

By multiplying $\frac{2d\alpha}{\eta}$ on both sides of (8) and adding in (7), we can get:

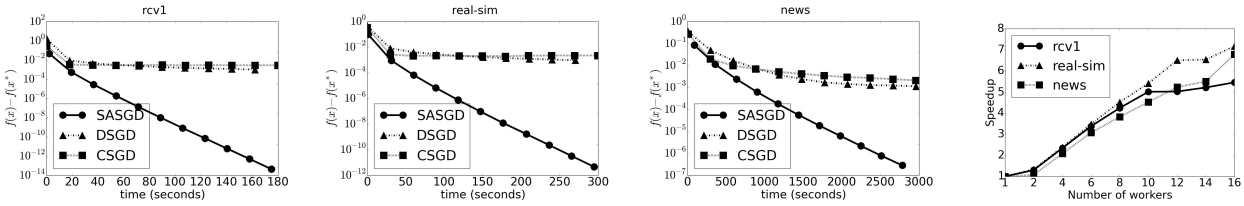
$$\begin{aligned}
&\nabla F(x_{k+1})^T(z_k - x^*) \le \\
&\quad \frac{m}{2\alpha}\|z_k - x^*\|^2 - \frac{m}{2\alpha}\mathbb{E}_{I_k,j_k}\|z_{k+1} - x^*\|^2 \\
&\quad + \frac{\sqrt{\Delta}}{2}\|x_{k+1} - x^*\|^2 + \frac{\sqrt{\Delta}}{2}\left(\frac{1-\beta}{\beta}\right)\|x_{k+1} - y_k\|^2 \\
&\quad + \frac{2m\alpha}{\eta}(F(x_{k+1}) - \mathbb{E}_{I_k,j_k}F(y_{k+1})) \\
&\quad + 2\left(\alpha(\frac{T\eta}{2} + \sqrt{\Delta}) + \frac{\sqrt{\Delta}}{\beta}\right)\mathbb{E}_{I_k}\|\bar{u}_{k+1}\|^2 \\
&\quad - 2\alpha\left(1 - \frac{\sqrt{\Delta}}{2}\right)\|\nabla F(x_{k+1})\|^2 \\
&\quad + 2\left(\alpha\sqrt{\Delta} + \frac{\sqrt{\Delta}}{\beta}\right)\mathbb{E}_{I_k}\|\bar{v}_{k+1}\|^2. \qquad (9)
\end{aligned}$$

Step 3: Let $\frac{1-\beta}{\beta} = \frac{2\alpha m}{\eta}$. We take expectation with respect to τ_k. Under the assumption $\sqrt{\Delta} < \mu$, by summing inequality (9) over $k = 0, ..., K-1$, we can get:
$\sum_{k=0}^{K-1} F(x_{k+1}) - F(x^*) \le \frac{m}{2\alpha}\|z_0 - x^*\|^2 + \sum_{k=0}^{K-1}\alpha H\mathbb{E}_{I_k}\|\bar{u}_k\|^2$
$- \sum_{k=0}^{K-1}\alpha D\|\nabla F(x_{k+1})\|^2 + \frac{2m\alpha}{\eta}(F(y_0) - F(x^*))$,
where $H = 2\tau\left[\left(\frac{L_{max}\eta}{2} + \sqrt{\Delta}\right) + \left(\frac{1}{2\tau} + \frac{m\sqrt{\Delta}}{\eta}\right)\right]$ and $D = 2\left(1 - \frac{\sqrt{\Delta}}{2}\right)$. According to Inequality (12) in paper [Lee *et al.*, 2015], under the condition $D \ge H$, we can get:

$$(1 - \alpha AD)\delta_s \le \left(\frac{m}{2K\mu\alpha} + \frac{2m\alpha}{K\eta} + \alpha AD\right)\delta_{s-1},$$

where $\delta_s = F(\tilde{x}_s) - F(x^*)$. □

As for Theorem 4.2, we have the following discussions. The introduction of asynchronous parallelization does hurt the convergence of SASGD by a little (mainly due to the communication delay). However, when we run the AASGD algorithm on multiple machines, we can still achieve significant speed up under certain conditions, as shown in the following corollary.



(a) comparison on rcv1 (b) comparison on real-sim (c) comparison on news20 (d) speedup

Figure 1: Figure(a), Figure(b) and Figure(c) are comparisons of SASGD and SGD on three datasets respectively; Figure(d) is the speedups for AASGD on three datasets.

Corollary 4.4 *Assume* $\sqrt{\Delta} < \min\{\mu, \frac{1}{4}\}$. *If we set the step size* $\eta = \frac{1}{2L_{max}\tau}$,$\alpha = \min\{\frac{1}{6A}, \frac{1}{\sqrt{L_{max}\mu\tau}}\}$ *and select the block size* $m < \frac{1}{16L_{max}\Delta^{1/4}}$, *then AASGD will achieve at least square root speedup with respect to the number of workers when* $\tau < \frac{1}{4\Delta^{1/8}}$.

Proof: Compared with SASGD, the convergence rate in Theorem 4.2 is slower due to the delay. In order to get speedup for AASGD, the order of inner loop K cannot be τ times larger than SASGD. The order of K is related to two parameters η and α. In SASGD, $\eta_0 = \frac{1}{L_{max}}$. For AASGD, we set $\eta = \frac{1}{2L_{max}\tau}$. Here we are interested in the situation where the $\Delta \ll 1$. Considering $\sqrt{\Delta} < \min\{\mu, \frac{1}{4}\}$, the condition $\left(1 - \frac{\sqrt{\Delta}}{2}\right) > \tau\left(\frac{L_{max}\eta}{2} + \sqrt{\Delta} + \frac{1}{2\tau} + \frac{m\sqrt{\Delta}}{\eta}\right)$ can be satisfied with $m < \frac{1}{16L_{max}\Delta^{1/4}}$ and $\tau < \frac{1}{4\Delta^{1/8}}$. For AASGD, we set $\alpha = \min\{\frac{1}{\sqrt{L_{max}\mu\tau}}, \frac{1}{6A}\}$ and $b = \frac{24n\sqrt{\kappa_1/\tau}}{(n-1)+24\sqrt{\kappa_1/\tau}}$, the inner loop K is in the same order of $\mathcal{O}\left(\left(nm + \frac{nm\sqrt{\kappa_2}\sqrt{\kappa_1}}{n+\sqrt{\kappa_1/\tau}}\right)\log\frac{1}{\epsilon}\right)$. Compared with SASGD, it is at most $\sqrt{\tau}$ larger but the computing time is τ times smaller. The full gradient calculation can be distributed to multiple machine to get P times speedup. As a result, AASGD can achieve at least $\sqrt{P}$ times speedup since τ is in proportion to the number of local worker P. □

Remark: Data sparsity serves as a condition for the speedup. If the data is sparse, the condition is more likely to hold, and we will get the speed-up.

5 Experiments

In this section, we report our experimental results to demonstrate the efficiency of AASGD, and validate our theoretical findings. We conducted binary classification tasks on three benchmark dataset: *rcv1, real-sim, news20*, where *rcv1* is the densest dataset and *news20* is of the highest dimension. The detailed information about the three datasets can be found from LibSVM website. The objective function used in our experiments is the L_2-regularized logistic regression loss. We mainly compared AASGD, SASGD, and standard SGD algorithms, in terms of their convergence properties. In the AASGD algorithm, we set the number of block partitions as $m = d/100$, the mini-batch size as $\sqrt{n}/\sqrt{P}$ (P is the number of threads), and the inner loop $K = 2mn$. The stopping criterion in our experiments is the training error smaller than 10^{-10} (i.e., $F(x_k) - F(x^*) < 10^{-10}$). For the datasets we used, $L_{max} = L_{res} < 0.25$ since the input data is normalized [Reddi *et al.*, 2015], $\tau \approx P$, $\mu = 1/\sqrt{n} = 0.01$ [Shamir *et al.*, 2014]. In SASGD and AASGD, we set stepsizes $\eta_0 = 0.2$ and $\eta = 0.1/P$, which satisfy our assumptions in the theorems and corollaries.

First, we investigate the SASGD algorithm. We compare it with standard SGD with different step sizes: a constant step size (denoted as C-SGD) and a decreasing step size as $1/\sqrt{k}$ where k is the iteration [Reddi *et al.*, 2015](denoted as D-SGD). We plot the training curves of SASGD, C-SGD, and D-SGD in Figure 1(a),1(b),and 1(c)(for the three datasets respectively). From the figure, we have the following observations: D-SGD and C-SGD are faster than SASGD when we want to get a low-precise solution,i.e., $\epsilon < 10^{-2}$. As the training process continues, SASGD converges much faster than D-SGD and C-SGD on all the three datasets when we want to get a high-precise solution. This is consistent with our theoretical results (complexity analysis).

Second, we examine the benefit of AASGD by running on multiple workers (when there is only one local worker, AASGD reduces to SASGD). We implement AASGD with 2,4,..,16 workers, and plot its speedups in Figure 1(d) (for the three datasets respectively). The speedup is defined as the ratio of the runtime with a single thread to the runtime with P threads. From these figures, we have the following observations:(1) on all the three datasets, AASGD has considerable speedup with respect to the number of workers, in an order lower than linear but higher than square root; (2) the smallest speedup is observed on *rcv1*. This is consistent with our theoretical results, since *rcv1* is the densest dataset. [Reddi *et al.*, 2015; Recht *et al.*, 2011] get a similar result on *rcv1*.

6 Conclusions

In this paper, we focus on asynchronous SGD with acceleration methods, such as variance reduction, Nesterov's acceleration, and stochastic coordinate sampling. We prove the convergence rate for AASGD, and analyze the conditions for its speedup with respect to the number of workers. Our experiments on benchmark datasets well validated our theoretical findings. In the future, we will investigate the convergence rate of SGD with more acceleration methods, and AASGD in distributed network architectures.

References

[Agarwal and Duchi, 2012] Abhishek Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5451–5452. IEEE, 2012.

[Avron *et al.*, 2014] Haim Avron, Alex Druinsky, and Arpan Gupta. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 198–207. IEEE, 2014.

[Bousquet and Bottou, 2008] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 161–168, 2008.

[Defazio *et al.*, 2014] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1646–1654, 2014.

[Hu *et al.*, 2009] Chonghai Hu, Weike Pan, and James T Kwok. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 781–789, 2009.

[Johnson and Zhang, 2013] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013.

[Langford *et al.*, 2009] John Langford, Alexander Smola, and Martin Zinkevich. Slow learners are fast. *arXiv preprint arXiv:0911.0491*, 2009.

[Lee *et al.*, 2015] Jason Lee, Tengyu Ma, and Qihang Lin. Distributed stochastic variance reduced gradient methods. *arXiv preprint arXiv:1507.07595*, 2015.

[Lian *et al.*, 2015] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2719–2727, 2015.

[Nemirovski *et al.*, 2009] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[Nesterov, 2004] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[Nesterov, 2012] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[Nitanda, 2014] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1574–1582, 2014.

[Rakhlin *et al.*, 2012] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 449–456, 2012.

[Recht *et al.*, 2011] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 693–701, 2011.

[Reddi *et al.*, 2015] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2629–2637, 2015.

[Richtárik and Takáč, 2014] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[Roux *et al.*, 2012] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *arXiv preprint arXiv:1202.6258*, 2012.

[Shalev-Shwartz and Tewari, 2011] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l 1-regularized loss minimization. *The Journal of Machine Learning Research*, 12:1865–1892, 2011.

[Shalev-Shwartz and Zhang, 2013] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[Shamir *et al.*, 2014] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1000–1008, 2014.

[Zhang *et al.*, 2013] Shanshan Zhang, Ce Zhang, Zhao You, Rong Zheng, and Bo Xu. Asynchronous stochastic gradient descent for dnn training. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6660–6663. IEEE, 2013.

[Zhao *et al.*, 2014] Tuo Zhao, Mo Yu, Yiming Wang, Raman Arora, and Han Liu. Accelerated mini-batch randomized block coordinate descent method. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3329–3337, 2014.