2020 IEEE 16th International Conference on Control & Automation (ICCA)
Oct 9-11, 2020 (Virtual) Sapporo, Hokkaido, Japan

Asynchronous Stochastic Gradient Descent over Decentralized Datasets

Yubo Du, Keyou You, *Senior Member, IEEE*, Yilin Mo

***Abstract*— Asynchronous stochastic gradient descent (ASGD) usually works in the centralized setting in which workers retrieve data from a shared training set. This paper focuses on decentralized scenarios where each worker only accesses a subset of the whole training set. We find that due to the heterogeneous properties of the decentralized setting, ASGD will optimize in wrong directions and thus obtain poor solutions. To tackle the issue, a novel algorithm DASGD is proposed for above setting. Our key idea is to form an asymptotically unbiased accurate gradient estimate through reweighting stochastic gradient based on importance sampling technique. Numerical results substantiate the performance of the proposed algorithm in the decentralized setting.** 核心思想

I. Introduction

We focus on stochastic optimization problems of the following form:

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) := \mathbb{E}_\xi\left[F(\mathbf{x};\xi)\right] \tag{1}$$

where $\xi \in \mathcal{S}$ is a random variable and $\mathcal{S}$ denotes the training set. $F(\mathbf{x};\xi)$ is the loss function of a certain learning task with respect to the sample ξ. The decision vector $\mathbf{x}$ is the model parameter to be optimized and $f(\mathbf{x})$ denotes the expected risk.

Classical stochastic optimization algorithms such as stochastic gradient descent can solve problem (1). In recent years, due to the demand of solving large-scale machine learning problems, asynchronous parallel algorithms have been receiving considerable attention. The celebrated asynchronous stochastic gradient descent (ASGD) follows the master-worker architecture, as depicted in Fig. 1. In the workflow of the ASGD, each worker continuously pulls the latest model form the master, fetches a minibatch of samples from a shared training set, computes stochastic gradients, and pushes them back to the master. The master then updates the global model with stochastic gradients. The update rule of the ASGD can be formulated as follows:

minibatch size

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \sum_{m=1}^{M} \nabla F(\mathbf{x}_{k-\tau_k};\xi_{k,m}) \tag{2}$$

计算梯度的模型落后于最新模型的迭代次数

where M is the minibatch size and τ_k is the gradient delay with respect to the mth sample at the kth iteration. The delay represents how many iterations the model used to calculate gradients has fallen behind the latest model. Delay is caused by asynchronous push of workers so it is intrinsic to asynchronous algorithm. In the setting in Fig. 1, all the workers can get samples from the shared training set, which

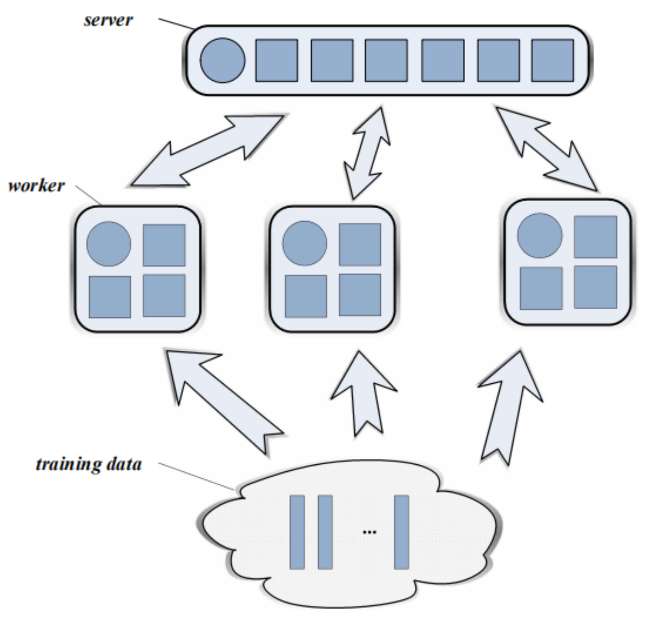

Fig. 1. Master-worker architecture for asynchronous parallel algorithms.

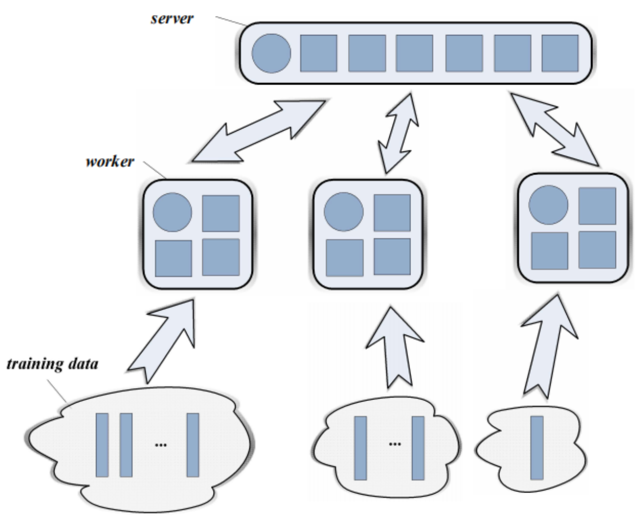

Fig. 2. Master-worker architecture in the decentralized setting.
采用去中心化的设计，每一个worker只能访问数据集的yi bu

is crucial to the deployment of the ASGD algorithm. We refer to it as the centralized setting. This paper focuses on the decentralized setting where each worker can only access a subset of the training set. This setting is illustrated in Fig. 2.

Our choice of scenarios is not contrived. When the training data are dispersedly collected or stored, considering the high

Y. Du and K. You are with the Department of Automation, and BNRist, Tsinghua University, Beijing 100084, China. E-mail: duyb19@mails.tsinghua.edu.cn, youky@tsinghua.edu.cn, ylmo@tsinghua.edu.cn

cost of communication or some privacy policy, it is not sensible to bring them together, and thus the ASGD can not be deployed in these situations. With rapidly expanding training data and growing demand for distributed computation, the decentralized setting will be more common. In the decentralized setting, the training set contains several local subsets, and each worker can only access one of them. The ith subset is denoted by $\mathcal{S}_i$ and can only be accessed by worker i. The corresponding update rule of ASGD can be written as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \sum_{m=1}^{M} \nabla F(\mathbf{x}_{k-\tau_k}; \xi_{k,m}), \ \xi_{k,m} \in \mathcal{S}_{i_k}$$

where i_k is the index of the worker which pushes gradients at the ith iteration. The only difference from the update rule in Equ. (2) is that in the decentralized setting sample $\xi_{k,m}$ is selected from the ith subset. Our claim is that this subtle distinction may cause performance loss of the ASGD algorithms. Due to the possible heterogeneous properties of distributed systems, including unbalanced usage of training samples and varying computing power among workers, the ASGD in the decentralized setting would obtain poor solutions to Problem (1). This can translate into generalization performance loss of trained models in machine learning. Details will be discussed later.

In this paper, we propose a modified asynchronous stochastic gradient descent with adaptive stepsize to solve Problem (1) in the decentralized setting. We refer to our proposed algorithm as the DASGD. As we will show in the next section, simple stochastic gradient is a biased estimate of the gradient of expected risk, which is caused by heterogeneous properties of the decentralized setting. To address this issue, the DASGD constructs a new gradient estimate through counting distributions of training samples to model update and reweighting stochastic gradient. It can be verified that the new estimate is an asymptotic unbiased estimate of the full gradient, which ensures the asynchronous SGD converges correctly.

Research on asynchronous parallel algorithms can be traced back at least to seminal work of Bertsekas and Tsitsiklis on asynchronous computation [1], [2]. In the past decade many asynchronous parallel algorithms have been developed to solve large-scale machine learning tasks. Compared with synchronous algorithms, asynchronous algorithms are characterized by high flexibility and efficiency since they have no idle time and can fully utilize computation resources. For these reasons, asynchronous parallelism is exploited to achieve the speedup of many serial optimization algorithms, including stochastic gradient descent [3]–[6], stochastic coordinate descent [7], block coordinate descent [8] and dual stochastic coordinate ascent [9]. It is worth noting that all these algorithms are based on the master-worker architecture and the centralized setting.

Few works on asynchronous parallel algorithms in the decentralized setting can be found. [10] proposed a decentralized parallel algorithm D^2 whose decentralized setting is identical to ours. But D^2 is a peer-to-peer algorithm where each worker iteratively updates its local model with local data, communicates with each other and finally achieves a consensus model. Large data variance across workers will make this progress converge slowly, and D^2 applies a variance reduction technique to tackle this issue. However, in the ASGD based on the master-worker framework, since all the workers share the same global model and all the samples are selected randomly to update this model, data variance among workers will be accounted into the variance of the whole dataset and thus we meet with some different issues. Specifically, our nuisance is that uneven usage of training samples will alter data distribution and incur optimization deviation from the original objective in Problem (1).

The rest of this paper is organized as follows. In Section II, we interpret the heterogeneous issues of the ASGD in the decentralized setting in detail and show their influences on the algorithm performance. In Section III, we construct a new gradient estimate step by step and obtain the DASGD algorithm. In Section IV, we give some preliminary results of convergence anaysis for the DASGD. Section V conducts several numerical experiments to validate the performance of the DASGD in the decentralized setting. Finally, we some draw conclusive remarks in Section VI. $\longleftarrow$ 文章结构讲解

Notation: We use a bold letter $\mathbf{x}$ to denote a vector. $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inner product of $\mathbf{x}$ and $\mathbf{y}$. We use $\|\cdot\|$ to denote the l_2-norm. $|\mathcal{S}|$ denotes the cardinality of set $\mathcal{S}$. $\mathbb{R}^n$ denotes the set of all n-dimensional real numbers. We use $\nabla f(\mathbf{x})$ to denote the gradient of a differentiable function f at $\mathbf{x}$. $\mathbf{x}^*$ denotes the global optimal solution to problem (1).

II. Heterogeneous Issues over Decentralized Datasets

As mentioned above, the ASGD in the decentralized setting will obtain poor solutions. In this section we will discuss those heterogeneous issues thoroughly. Repeat our decentralized setting: the training set $\mathcal{S}$ consists of m local subsets, each of which is accessible to only one worker. Note that workers may vary significantly in subset size and computing power. For the sake of clarity, we explore these two aspects respectively.

Suppose that workers run equally fast but with varying subset sizes. Figure 3 gives an illustration of the ASGD workflow. In this example worker 1 has one sample and worker 2 has three. During the first 6 iterations of the ASGD, both of two workers delivered gradients to the master twice (we assume batch size is 1). However, the only sample of worker 1 was used 3 times while each sample of worker 2 was used once. It reveals that due to varying subset sizes, some samples are utilized more frequently than others. Then we suppose that workers have equal subset sizes but vary in operational speed. Figure 4 provides an example of this situation. Both workers have 2 samples, but worker 1 ran faster and delivered gradients to the master 4 times during the first 6 iterations while worker 2 only delivered twice. Thus each sample of worker 1 was used twice while each sample of worker 2 was used only once. It means that varying

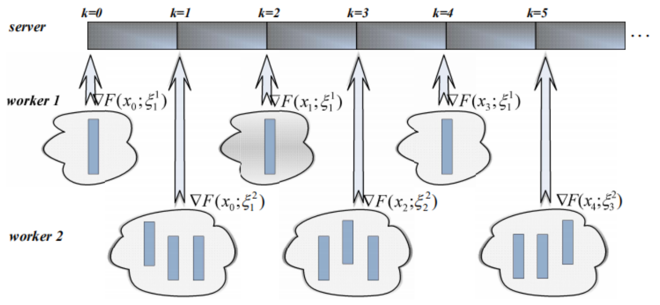

Fig. 3. In this situation, workers in ASGD run equally fast but with varying subset size.

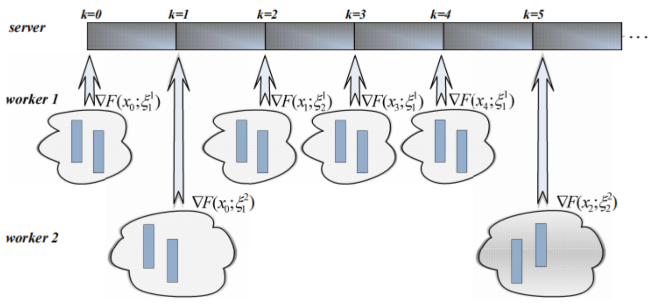

Fig. 4. In this situation, workers in ASGD have equal subset sizes but with varying operational speed.

operational speed also incurs uneven utilization of training samples.

子集的大小和处理速度导致了训练样本利用率的不均衡

In a nutshell, both the heterogeneity of subset sizes and operational speed among workers cause uneven utilization of training samples. In most learning tasks such as deep neural network training, given the training set, it is usually hoped that samples can be used uniformly. This requirement can be translated into the assumption of unbiased stochastic gradient in convergence analysis of the classical stochastic gradient descent and its variants, which can be written as follows:

$$\mathbb{E}_{\xi}\left[G(\mathbf{x};\xi)\right] = \nabla f(\mathbf{x}) \tag{3}$$

where $G(\mathbf{x};\xi)$ is the update term. In the classical SGD and its asynchronous versions, $G(\mathbf{x};\xi)$ is exactly the stochastic gradient $\nabla F(\mathbf{x};\xi)$). Obviously uneven utilization of training samples can not guarantee the validity of this assumption.

In equation (3), ξ is randomly selected from the whole training set $\mathcal{S}$ and the expectation is taken with respect to ξ. The actual probability distribution of ξ depends on how frequently the master uses stochastic gradients calculated with these samples to update the model. In principle we hope that all of them are selected with equal probability so the ideal probability distribution of ξ is uniform. If the actual and ideal probability distributions are equal or just their expectations are equal, assumption (3) will be definitely satisfied. In the decentralized setting, considering the heterogeneous issues illustrated above, neither the probability distributions nor the expectations are equal, so assumption (3) is clearly not met. This could be expressed as follows:

$$\mathbb{E}_{\xi\in\mathcal{S}'}\left[\nabla F(\mathbf{x};\xi)\right] \neq \mathbb{E}_{\xi\in\mathcal{S}}\left[\nabla F(\mathbf{x};\xi)\right] = \nabla f(\mathbf{x})$$

where $\mathcal{S}'$ is an imaginary training set corresponding to actual probability distribution of ξ, which means if we choose samples uniformly from $\mathcal{S}'$ we obtain a ξ distribution equivalent to its actual distribution in the ASGD. $\nabla F(\mathbf{x};\xi)$ is an unbiased estimate of gradient $\nabla f'(\mathbf{x})$ corresponding to $\mathcal{S}'$ where $f'(\mathbf{x}) := \mathbb{E}_{\xi\in\mathcal{S}'}\left[F(\mathbf{x};\xi)\right]$. It indicates that in the decentralized setting ASGD optimizes a different objective and naturally obtains poor solutions.

III. The Asynchronous SGD on Decentralized Datasets

Based above analysis, this section construct a better estimate of gradient and give a simple algorithm to calculate it. This leads to the novel DASGD algorithm.

A. *Asymptotically Unbiased Gradient Estimate*

As demonstrated in the previous sections, in the decentralized setting, stochastic gradient $\nabla F(\mathbf{x};\xi)$ is a biased estimate of the real gradient $\nabla f(\mathbf{x})$, which prevents ASGD from converging well. We wish to construct an unbiased estimate or just an asymptotically unbiased estimate of the gradient, which may still be better than $\nabla F(\mathbf{x},\xi)$. Actually, the imaginary training set $\mathcal{S}'$ introduced in the last subsection may give us an insight into this problem. Since ξ is equivalently selected from $\mathcal{S}'$ in ASGD, we hope to obtain a $G(\mathbf{x};\xi)$ to make the following equation hold:

$$\mathbb{E}_{\xi\in\mathcal{S}}\left[\nabla G(\mathbf{x};\xi)\right] = \mathbb{E}_{\xi\in\mathcal{S}}\left[\nabla F(\mathbf{x};\xi)\right]$$

where $G(\mathbf{x};\xi)$ is some variant of $\nabla F(\mathbf{x};\xi)$. Denote probability distributions of $\mathcal{S}$ and $\mathcal{S}'$ by $p(\xi)$ and $p'(\xi)$ respectively. Now we define $G(\mathbf{x};\xi)$ as follows:

$$G(\mathbf{x};\xi) = \frac{p(\xi)}{p'(\xi)}\nabla F(\mathbf{x};\xi)$$

It is easy to verify that $G(\mathbf{x};\xi)$ is unbiased:

$$\mathbb{E}_{\xi\in\mathcal{S}'}\left[\frac{p(\xi)}{p'(\xi)}\nabla F(\mathbf{x};\xi)\right] = \mathbb{E}_{\xi\in\mathcal{S}}\left[\nabla F(\mathbf{x};\xi)\right]$$

This procedure is reminiscent of importance sampling technique. Indeed, the construction of unbiased gradient estimate can be depicted in the view of importance sampling: we aim to sample stochastic gradient from distribution $\mathcal{S}$ with only samples $\nabla F(\mathbf{x};\xi)$ generated from another different distribution $\mathcal{S}'$.

Note that the probability distributions mentioned above are not pertinent to the generation process of data. Instead they just describe how frequently each sample in the training set is used to update the model. Since we hope that each sample would be selected with equal possibility, we have $p(\xi) = 1/N$ where N is the size of the training set. Denote by n_i each subset's size, and assume there are m workers. Obviously we have $\sum_{i=1}^{m} n_i = N$.

It is intractable to compute the exact probability distribution of $\mathcal{S}'$ because there is no prior information about how frequently each sample will be used in any instantiation of ASGD procedure. It may vary in different scenarios or even keep changing within a running instance. Alternatively, we wish to estimate the distribution dynamically. Suppose that the master keeps one counter for every worker to count how many times it has received the stochastic gradients pushed by this worker. Denote by $c_i(k)$ the counter variable corresponding to worker i at the kth iteration. Each time worker i pushes a minibatch of stochastic gradients, c_i will increase by one. Thus at the kth iteration, the frequency of each sample in worker i 's subset being used would be $\frac{c_i(k)}{n_i k}$(In the subset of each worker uniform selection of samples can be easily realized). The law of large numbers tells us that this frequency will converge to the real probability of the sample in $\mathcal{S}'$ as k goes to infinity:

有一个主计数器，为每一个worker保留了一个计数器，计算这worker更新梯度的次数

$$p'(\xi) = \frac{c_i(k)}{n_i k}, \quad k \to \infty, \quad \xi \in \mathcal{S}$$

When k is large enough, the approximation error will be still acceptable. So we can estimate $p'(\xi)$ in this way:

$$p'(\xi) \leftarrow \frac{c_i(k)}{n_i k}, \quad \xi \in \mathcal{S}_i \tag{4}$$

Then we obtain a stochastic gradient variant $G(\mathbf{x};\xi)$:

$$G(\mathbf{x};\xi) = \frac{n_i k}{N c_i(k)} \nabla F(\mathbf{x};\xi), \quad \xi \in \mathcal{S}_i \tag{5}$$

Thus $G(\mathbf{x};\xi)$ is an asymptotically unbiased estimate of $\nabla f(\mathbf{x})$:

$$\mathbb{E}_{\xi\in\mathcal{S}'}\left[G(\mathbf{x};\xi)\right] = \mathbb{E}_{\xi\in\mathcal{S}}\left[\nabla F(\mathbf{x};\xi)\right] = \nabla f(\mathbf{x}), \quad k \to \infty$$

A novel algorithm is obtained by replacing $\nabla F(\mathbf{x};\xi)$ in ASGD with $G(\mathbf{x};\xi)$ defined in equation (5) and we refer to it as ASGD on decentralized datasets(or DASGD for short).

B. The DASGD: Algorithm Description

We are ready to present the DASGD algorithm. As shown above, $G(\mathbf{x};\xi)$ is a feasible estimate of the gradient $\nabla f(\mathbf{x})$ and we can use it to update the model. So the update rule of the DASGD can be written as follows:

$$c_i(k+1) = c_i(k) + 1 \tag{6a}$$

$$G(\mathbf{x}-\tau_k;\xi_{k,m}) = \frac{n_i(k+1)}{N c_i(k+1)} \nabla F(\mathbf{x}-\tau_k;\xi_{k,m}), \quad \xi_{k,m} \in \mathcal{S}_i \tag{6b}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \sum_{m=1}^{M} G(\mathbf{x}-\tau_k;\xi_{k,m}) \tag{6c}$$

The workflow of the DASGD is described in Algorithm 1 and 2. We assume that the DASGD is implemented with parameter server framework. According to Algorithm 1, each worker runs in the identical way to ASGD: local worker i pulls the latest model $\mathbf{x}_k$ from the parameter server, randomly chooses samples from its subset $\mathcal{S}_i$, calculates a minibatch of stochastic gradients $\sum_{m=1}^{M} \nabla F(\mathbf{x}_k;\xi_{k,m})$ and pushes it back to the server. In Algorithm 2, with gradients from worker i received, the parameter server updates corresponding counter c_i, calculates gradient estimate and conducts gradient descent.

Algorithm 1 The DASGD: worker i

```
• Repeat
  1: Pull $\mathbf{x}_k$ from the parameter server.
  2: Randomly select $M$ samples $\{\xi_m\}_{m=1,\cdots,M}$ from $\mathcal{S}_i$.
  3: Calculate a minibatch of stochastic gradients $\sum_{m=1}^{M} \nabla F(\mathbf{x}_k;\xi_{k,m})$.
  4: Push $\sum_{m=1}^{M} \nabla F(\mathbf{x}_k;\xi_{k,m})$ to the parameter server.
• Until forever.
```

Algorithm 2 The DASGD: Parameter server

```
• Initialization: $\{\gamma_k\}_{k=0,\dots}$, $k = 0$, $\mathbf{x}_0$, $c_i = 0, i = 1,\cdots,m$.
• Repeat
  – If receive $\sum_{m=1}^{M} \nabla F(\mathbf{x}-\tau_k;\xi_{k,m})$ from worker $i$
    1: $c_i(k+1) = c_i(k) + 1$.
    2: $\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \frac{n_i(k+1)}{N c_i(k+1)} \sum_{m=1}^{M} \nabla F(\mathbf{x}-\tau_k;\xi_{k,m})$.
    3: $k := k+1$.
  – Else if receive "pull request" from worker $i$
    1: Push $\mathbf{x}_k$ to worker $i$.
```

Note that compared to the ASGD, the ASGD has no additional computational and communication cost for workers. For the server, the additional cost of updating counter variables and modifying gradients is negligible. So the computational complexity of the DASGD does not worsen substantially. Only some extra information about subset sizes is required.

IV. Convergence Analysis

In this section, we directly give some of the convergence results, and the complete and detailed analysis is deferred to the journal version of this work.

To analyse the convergence rate of the DASGD, we make the following assumptions:

Assumption 1 (Poisson running process): The arrival of worker i's push events follows a stationary poisson process denoted by $\{N_i(t), t \geq 0\}$ with rate λ_i, $i = 1,\dots,r$.

Since workers' pushing process are modelled with r independent stationary poisson processes, the process of the master receiving gradients from r workers can be treated as the superposition of these poisson processes. According to the superposition theorem of poisson process, the superposition of different poisson processes $N_1(t),\dots,N_r(t)$ with rates $\lambda_1,\dots,\lambda_r$ will be a poisson process $N(t)$ with rate $\lambda = \sum_{i=1}^{r} \lambda_i$. Moreover, the probability of an event sampled from $N(t)$ belonging to jth poisson process $N_j(t)$ is given by λ_j/λ. In our scenario this is exactly the probability of samples of worker j being used by the server. Divide it by subset size n_i and we obtain the probability of each sample in $\mathcal{S}$ being used, which is precisely the value of $p'(\xi), \xi \in \mathcal{S}_i$:

$$p'(\xi) = \frac{\lambda_i}{\lambda n_i},\ \xi \in \mathcal{S}_i$$

Apparently $p'(\xi)$ in Equation (4) is an approximator of $p'(\xi)$ here. With an exact expression of $p'(\xi)$, we can derive a new formula for the unbiased gradient estimate which is more amenable to our analysis and we denote it by $G'(\mathbf{x}, \xi)$:

$$G'(\mathbf{x}; \xi) = \frac{p(\xi)}{p'(\xi)} \nabla F(\mathbf{x}; \xi) \tag{7}$$

And $G(\mathbf{x}; \xi)$ in Equation (5) is an approximator of $G'(\mathbf{x}; \xi)$ here. Based on the definition of $G'(\mathbf{x}; \xi)$, we make the following assumptions:

Assumption 2: We assume that the following holds:

1) *(Unbiased Gradient):* The stochastic gradient $\nabla F(\mathbf{x}; \xi)$ for ASGD over centralized datasets is unbiased:

$$\mathbb{E}_{\xi \in \mathcal{S}} [\nabla F(\mathbf{x}; \xi)] = \nabla f(\mathbf{x})$$

2) *(Bounded Variance):* $G'(\mathbf{x}; \xi)$ has a bounded variance:

$$\mathbb{E}_{\xi \in \mathcal{S}'} (\|G'(\mathbf{x}; \xi) - \nabla f(\mathbf{x})\|^2) \leq \sigma^2,\ \forall \mathbf{x}$$

3) *(Lipschitzian Gradient):* The gradient $\nabla f(\cdot)$ of the objective function $f(\mathbf{x})$ is L-Lipschitz continuous:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|,\ \forall \mathbf{x}, \mathbf{y}$$

Combining the condition of unbiased gradient in Assumption 2 and the definition of $G'(\mathbf{x}; \xi)$ in Equation (7), we have the unbiased property of $G'(\mathbf{x}; \xi)$ for ASGD over decentralized datasets:

$$\begin{aligned}\mathbb{E}_{\xi \in \mathcal{S}'} [G'(\mathbf{x}; \xi)] &= \mathbb{E}_{\xi \in \mathcal{S}'} \left[\frac{p(\xi)}{p'(\xi)} \nabla F(\mathbf{x}; \xi)\right] \\ &= \mathbb{E}_{\xi \in \mathcal{S}} [\nabla F(\mathbf{x}; \xi)] = \nabla f(\mathbf{x})\end{aligned}$$

Assumption 3: We assume that the following holds:

1) *(Sample Independence):* All samples $\xi_{k,m}$ in algorithm 1 are independent to each other.
2) *(Bounded Delay):* All delay variables τ_k in algorithm 2 are bounded by T: $\max_k \tau_k \leq T$

The following theorem establishes the convergence rate of the DASGD.

Theorem 1 (Convergence rate of the DASGD): Suppose that Assumption 1,2,3 hold. If the stepsize γ_k is a constant and satisfies that:

$$\gamma = \sqrt{(f(\mathbf{x}_1) - f(\mathbf{x}^*)) / (MLK\sigma^2)} c_2 / c_1$$

where c_1, c_2 are constants defined by:

$$c_1 = \max_{k \in \{1, \cdots, K\}} \left(\left(\frac{\lambda_{i_k}}{\lambda}\right)^2 \sigma_{i_k}(t_k) + 1\right)$$

$$c_2 = \min_{k \in \{1, \cdots, K\}} \left(1 - 3e^{\lambda_{i_k} t_k} \frac{\sum_{j \neq i_k} \lambda_j}{\lambda}\right)$$

And K is the total number of iterations and satisfies that:

$$4ML(f(\mathbf{x}_1) - f(\mathbf{x}^*))(T+1)^2 / \sigma^2 \leq K,$$

then the proposed DASGD algorithm has the following convergence rate:

$$\frac{\sum_{k=1}^{K} \mathbb{E}\left(\|\nabla f(\mathbf{x}_k)\|^2\right)}{K} \leq 4\sigma \sqrt{\frac{(f(\mathbf{x}_1) - f(\mathbf{x}^*)) L}{MK}}$$

V. Numerical Experiments

To evaluate the proposed DASGD algorithm, we perform numerical experiments on three typical machine learning tasks. In the heterogeneous decentralized setting, we compare the performance of the DASGD with that of the ASGD. The performance of the ASGD on centralized setting serves as a baseline and we hope the proposed algorithm would work efficiently on decentralized datasets as the ASGD does on centralized datasets.

A. Verify the performance

Logistic regression. Our experiments on logistic regression are conducted on the Reuters RCV1 dataset [11], which has 23,149 training and 781,265 test examples. Each example has 47,236 features and only about 0.16% of them are non-zero. We swap the training set and test set in our experiments to accentuate the impact of varying subset sizes. We use the C++ implementation of parameter server framework in [12]. The DASGD is implemented based on it. We run the ASGD over the centralized training set and evaluate model accuracy on the test set, which works as a baseline. Then we unevenly split the whole training set into several subsets and each worker can only access one of them. In this simulated decentralized setting, we run the ASGD and the DASGD, and evaluate the model accuracy respectively. In experiments on RCV1 dataset, the minibatch size is chosen as $M = 1000$ and the step size is chosen as $\gamma = 10^{-3}$. The parameter server runs in the local mode, where workers are implemented by multiple processes on one machine. We set 5 workers in this experiment.

SVM. The experiment setting of SVM is similar to that of logistic regression. To make sure the training process converges fast enough, we use Pegasos algorithm [13] to train SVM and parallelize it as a replacement of the ASGD. We replace the stochastic gradient in it by modified stochastic gradient used in the DASGD to correspond to the DASGD. The only difference between Parallel Pegasos and the ASGD is that Pegasos has an extra restriction of model parameter norm [13]. We set the punishment coefficient $\lambda = 1$ in SVM objective function. Like above, the minibatch size is chosen as $M = 1000$ and the step size is chosen as $\gamma = 10^{-3}$. We set 5 workers in this experiment.

CNN. We evaluate our algorithm by training LENET [14] on MNIST dataset [14]. LENET is a convolutional neural network designed for handwritten digit classification. We implement parallel training of neural network across multiple processes in PyTorch. We spawn 5 stochastic gradient workers and the experiment setting is identical to the preceding ones. The minibatch size M is equal to 64 and the step size is chosen as the default value $\gamma = 10^{-3}$.

Results of model accuracy in three experiments are summarized in table I. All the accuracy items were obtained by taking average among 5 runs. We treat the model accuracy in the first setting as a baseline. When the training set is unevenly distributed among workers and multiple processes naturally vary in operational speed, as shown in the second setting in table I, model accuracy can decrease up to 4.3 percent as anticipated, which is unacceptable in practical applications(The results would be much worse if we magnify the degree of heterogeneity of the system). The DASGD algorithm recovers model accuracy by more than 70%, which verifies its capability in the heterogeneous decentralized setting.

TABLE I

MODEL ACCURACY($\%$) IN THREE EXPERIMENTS

setting	Logistic Regression	SVM	CNN
centralized, ASGD	95.92	96.82	93.97
decentralized, ASGD	92.93	95.80	89.60
decentralized, DASGD	95.65	96.50	93.07

VI. CONCLUSION

We proposed the DASGD for stochastic optimization problems on decentralized datasets. To tackle the heterogeneous issues of ASGD on decentralized datasets we design a new gradient estimate for SGD update. An ergodic convergence rate of $O(1/\sqrt{K})$ is proved for non-convex and smooth objectives, which is analogous to the convergences result of ASGD on shared datasets. The assumption that running process of each worker is an independent stationary poisson process is not rarely seen in previous works on distributed optimization [15]–[18]. Generally speaking, totally asynchronous cases are intractable and there must be some constraints on the degree of asynchronous behaviors. Another concern is that before the execution of the DASGD the server needs to know the subset size of each worker, which may not be provided due to the privacy policy of data sources. So design of privacy preserved distributed algorithms can be a promising avenue for future work.

REFERENCES

[1] D. P. Bertsekas, "Distributed asynchronous computation of fixed points," *Mathematical Programming*, vol. 27, no. 1, pp. 107–120, 1983.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

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

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

[5] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Advances in Neural Information Processing Systems*, 2013, pp. 1223–1231.

[6] W. Zhang, S. Gupta, X. Lian, and J. Liu, "Staleness-aware async-sgd for distributed deep learning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, p. 2350–2356.

[7] J. Liu, S. J. Wright, C. Re, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 285–322, 2015.

[8] Y. Wang, V. Sadhanala, W. Dai, W. Neiswanger, S. Sra, and E. P. Xing, "Parallel and distributed block-coordinate frank-wolfe algorithms," *International Conference on Machine Learning*, pp. 1548–1557, 2016.

[9] K. Tran, S. Hosseini, L. Xiao, T. Finley, and M. Bilenko, "Scaling up stochastic dual coordinate ascent," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2015, p. 1185–1194.

[10] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 10–15 Jul 2018, pp. 4848–4856.

[11] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of machine learning research*, vol. 5, no. Apr, pp. 361–397, 2004.

[12] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*. USENIX Association, 2014, p. 583–598.

[13] S. Shalevshwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for svm," *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 2011.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] A. Nedic, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2010.

[16] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal processing*, vol. 57, no. 7, pp. 2748–2761, 2009.

[17] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Asynchronous gossip algorithms for stochastic optimization," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 3581–3586.

[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.