ON DISTRIBUTED STOCHASTIC GRADIENT DESCENT FOR NONCONVEX FUNCTIONS IN THE PRESENCE OF BYZANTINES

Saikiran Bulusu, Prashant Khanduri, Pranay Sharma, Pramod K. Varshney

EECS Department, Syracuse University
{sabulusu, pkhandur, psharm04, varshney}@syr.edu

ABSTRACT

We consider the distributed stochastic optimization problem of minimizing a nonconvex function f in an adversarial setting. All the w worker nodes in the network are expected to send their stochastic gradient vectors to the fusion center (or server). However, some (at most α-fraction) of the nodes may be Byzantines, which may send arbitrary vectors instead. Vanilla implementation of distributed stochastic gradient descent (SGD) cannot handle such misbehavior from the nodes. We propose a robust variant of distributed SGD which is resilient to the presence of Byzantines. The fusion center employs a novel filtering rule that identifies and removes the Byzantine nodes. We show that $T = \tilde{O}\left(\frac{1}{w\epsilon^2} + \frac{\alpha^2}{\epsilon^2}\right)$ iterations are needed to achieve an ϵ-approximate stationary point (x such that $\|\nabla f(x)\|^2 \leq \epsilon$) for the nonconvex learning problem. Unlike other existing approaches, the proposed algorithm is independent of the problem dimension.

***Index Terms*—** Byzantines, Stochastic Gradient Descent, Distributed optimization, Adversarial machine learning

1. INTRODUCTION

For most successful machine learning methods, the stochastic gradient descent (SGD) algorithm is a critical component [1]. However, the data that is used in the current machine learning methods is becoming increasingly distributed. This may be due to data being naturally distributed as in the case of federated learning [2] or because data is partitioned across machines to parallelize computations [3]. Therefore, it is crucial to consider the distributed versions of these machine learning methods along with the parallel implementation of SGD. Typically, in a distributed setup, the data is distributed across many worker machines or nodes with the objective of optimizing a global function [4, 5, 6, 7, 8, 9]. Also, at each node, a single (or a minibatch) stochastic gradient is computed per iteration. SGD randomly selects a single sample function and only computes the gradient for that function unlike gradient descent (where the gradient is computed over the entire batch). This results in faster computations over large datasets.

This work was supported in part by National Science Foundation under Grant ENG 60064237.

Due to the presence of many nodes in the distributed setup, robustness of the machine learning methods is a major concern. For instance, consider a network in which data is collected from the nodes and sent to a fusion center (FC) where some of the nodes exhibit adversarial behavior [10]. These *Byzantine* nodes can intentionally send arbitrary data. This behavior can gravely impact the performance of the learning task. Therefore, designing Byzantine-resilient distributed methods to solve the optimization problems which can endure adversarial attacks is of vital importance.

Due to the popularity and practical advantages of SGD, it has been used in several works on Byzantine machine learning [11, 12, 13, 14, 15, 16, 17]. A major challenge is the adversarial nature of the nodes that can send arbitrary vectors to the FC. These arbitrary vectors are likely to degrade the performance of the algorithm. However, except for a few of the works in this direction, the rest of the papers assume that the objective function is either strongly convex or convex. In this paper, we consider the robustness of the distributed SGD in the presence of Byzantine nodes when the objective function is nonconvex.

Related Work: Most of the recent works on distributed optimization [4, 5, 6, 7, 8, 9] assume that all the nodes are *normal*. Here, we shall only provide an overview of the work done in the recent years in the direction of Byzantine-resilient machine learning. Various aggregation rules have been proposed to enhance the robustness of distributed SGD [11, 12, 13, 14, 15, 16, 17]. In [11], distributed gradient descent is considered for a nonconvex objective function in the presence of Byzantine nodes. The coordinate-wise trimmed mean and coordinate-wise median filtering methods are considered. However, the rate of convergence is shown to be a function of the problem dimension. In [12], a nonconvex objective is considered. For the update step, the gradient that has the smallest distance to its closest gradients is selected. Furthermore, in [13], in addition to the filtering method described above, the geometric median of the gradients is also incorporated to improve robustness. In [15], the robustness of a variant of SGD for a strongly-convex objective function is analyzed. In [16], distributed SGD for convex objective function in the presence of Byzantines is considered. Filtering is based on statistics which are the functions of the gradient vectors sent

by the nodes.

Also, in [17], the generalized Byzantine model is considered where a subset of the elements of the gradient vector of the nodes are corrupted. They consider three median-based filtering rules which are Byzantine-resilient under certain conditions. Although in [12, 13, 17], the authors have considered distributed SGD for minimizing the nonconvex objective function in the presence of Byzantine nodes, they only provide empirical analysis of the proposed algorithms.

Furthermore, in [14], distributed SGD algorithm is considered with a nonconvex objective function. An element-wise majority vote on the signs of the gradient vectors of the nodes is used to filter out the Byzantines. Although convergence guarantees are provided, a very simple Byzantine model is considered where the Byzantine nodes cannot collude amongst themselves and can only change the magnitude of the gradient vector.

Contributions: The contributions in the paper are as follows:

- We propose a Byzantine-resilient variant of distributed SGD for minimizing a nonconvex objective function. We show that even though α-fraction ($\alpha < 1/2$) of the nodes are Byzantines, convergence is ensured for our proposed algorithm. The Byzantines can collude among themselves.
- We propose a novel filtering rule which is independent of the problem dimension, unlike [11]. Specifically, the proposed aggregation rule does not perform coordinate-wise operations. Hence, the convergence guarantees are independent of the problem dimension.
- We show that as the number of Byzantine nodes reduces to zero ($\alpha = 0$), the proposed algorithm guarantees best known rate of convergence for distributed nonconvex learning (Section 4) [18], [19].

1.1. Outline and Notations

The system model and problem statement are presented in Section 2. The proposed algorithm and the novel Byzantine filtering rule are presented in Section 3. Next, the main result of the rate of the paper is stated in Section 4. We conclude the paper in Section 5.

Notation: The expectation with respect to the stochasticity of the proposed algorithm is denoted by $\mathbb{E}[\cdot]$. The cardinality of set B is represented by $|B|$. We define the notation, $[C] = \{1, 2, \ldots, C\}$. Moreover, in this paper, we represent the ℓ_2-norm of a vector as $\|\cdot\|$. We define ∇ as the gradient.

2. SYSTEM MODEL

We assume that w nodes and a FC comprise a network as illustrated in Fig. 1. Note that at most α-fraction ($\alpha < 1/2$) of the workers are adversarial in nature. All the nodes communicate with the FC synchronously. We assume that the Byzantine nodes can communicate among themselves.

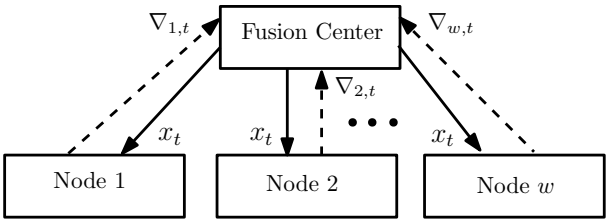

Fig. 1: System Model

Problem Formulation: Our goal is to minimize the following nonconvex $f : \mathbb{R}^d \to \mathbb{R}$

$$\min_{x \in \mathbb{R}^d} \{f(x) \triangleq \mathbb{E}_{s \sim \mathfrak{D}}[f_s(x)]\}, \tag{1}$$

where $\mathfrak{D}$ is the distribution over functions $f_s : \mathbb{R}^d \to \mathbb{R}$. We assume that the gradient of the function f has bounded norm, $||\nabla f(\cdot)|| \leq M$. We also assume that the function f is L-smooth,

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2, \forall x, y \in \mathbb{R}^d, \tag{2}$$

where $L > 0$ is the Lipschitz constant.

We also assume that each node has access to T sample functions from the distribution $\mathfrak{D}$. At each iteration $t = 0, 1, \ldots, T-1$, the FC broadcasts the point x_t to all the nodes in Fig. 1. Each node $i \in [w]$ sends back a vector $\nabla_{i,t} \in \mathbb{R}^d$

$$\nabla_{i,t} = \begin{cases} \nabla f_s(x_t), & s \sim \mathfrak{D}, i \in g, \\ *, & i \notin g, \end{cases} \tag{3}$$

where g is the set of normal nodes, $|g| \geq (1 - \alpha)w$. The Byzantine nodes may send an arbitrary vector denoted by $*$. A normal node returns a stochastic gradient $\nabla f_s(x_t)$ for a uniformly randomly chosen $s \sim \mathfrak{D}$. Therefore, $\mathbb{E}[\nabla_{i,t}] = \nabla f(x_t)$, for $i \in g$. Also, $\nabla_{i,t}$, for $i \in g$ has bounded norm and variance,

$$\|\nabla_{i,t}\| \leq M \text{ and } \|\nabla_{i,t} - \nabla f(x_t)\| \leq v. \tag{4}$$

The Byzantine nodes may collude among themselves in an attempt to degrade the optimization performance further.

In the absence of any Byzantine nodes ($\alpha = 0$), the FC can simply aggregate the $\{\nabla_{i,t}\}_{i \in [w]}$ vectors received from all the nodes, and use this aggregate to update the iterate value. However, since $\alpha > 0$, the FC needs to identify the *normal* nodes and filter out the *alleged* Byzantine nodes at each iteration. We assume that the FC also has access to some sample functions from the distribution $\mathfrak{D}$. At each time t, the FC computes an "extra" stochastic gradient $\nabla_{0,t}$, which is

Algorithm 1 Nonconvex Byzantine SGD

1: **Initialize** $x_0 \in \mathbb{R}^d$, step size $\eta > 0$, number of iterations T, $\mathfrak{T}_A = 2M^2\sqrt{2\log(\frac{2wT}{\delta})T}$ with $\delta \in (0,1)$ and $\delta = \frac{\mu}{\sqrt{T}}$ where $0 < \mu \leq 1$ and variance bound v, gradient norm bound M

2: $g_0 = [w]$

3: **for** $t = 0, 1, \ldots, T-1$ **do**

4: FC broadcasts x_t to all the nodes

5: **for** $i = 1, 2, \ldots, w$ **do**

6: Node i computes $\nabla_{i,t}$ according to Eq. (3)

7: **end for**

8: Receive $\nabla_{i,t} \in \mathbb{R}^d$ from all nodes $i \in [w]$

9: $x_{t+1} = x_t - \frac{\eta}{|g_t|}\sum_{i \in g_t} \nabla_{i,t}$

10: $A_i^{(t)} \leftarrow \sum_{t'=0}^{t} \langle \nabla_{0,t'}, \nabla_{i,t'} \rangle, \forall i \in [w]$

11: Compute $A_{med}^{(t)} = \text{med}\{A_1^{(t)}, \ldots, A_w^{(t)}\}$

12: $\nabla_{med}^t = \nabla_{i,t}$ where $i \in [w]$ is any node s.t $|\{j \in [w] : \|\nabla_{j,t} - \nabla_{i,t}\| \leq 2v\}| > \frac{w}{2}$

13: $g_{t+1} = \{i \in g_t : \left|A_i^{(t)} - A_{med}^{(t)}\right| \leq 2\mathfrak{T}_A \wedge \|\nabla_{i,t} - \nabla_{med}^t\| \leq 4v\}$

14: **if** $|g_{t+1}| < (1-\alpha)w$ **then**

15: $g_{t+1} = \{i \in g_t : \left|A_i^{(t)} - A_{med}^{(t)}\right| \leq 4M^2T \wedge \|\nabla_{i,t} - \nabla_{med}^t\| \leq 4v\}$

16: **end if**

17: **end for**

18: **Return** $\tilde{x}$ chosen uniformly randomly from $\{x_t\}_{t=1}^T$

used to update the set of *normal* nodes g_t over time. The FC only aggregates the vectors received from the nodes in g_t. This aggregate is used to update the iterate value which is then broadcast to all the nodes and the process repeats. Note that the nodes once declared as Byzantines are removed from consideration for all the future time instants.

Next, we propose a Byzantine-resilient variant of SGD to solve the minimization problem in (1). In particular, we propose a novel Byzantine filtering rule which guarantees state-of-the-art convergence rate.

3. NONCONVEX BYZANTINE SGD

We discuss the steps involved in the proposed Algorithm 1 that runs for T iterations. Each iteration $t \in [T]$ begins with the FC broadcasting the iterate x_t to all the worker nodes (Step 4). Also, at each iteration $t \in [T]$, the FC computes the set g_t consisting of the *normal* nodes.

The nodes in response, send $\nabla_{i,t}$ according to Eq. (3) to the FC in Step 6. Note that a normal node $i \in g$ sends a stochastic gradient, $\nabla_{i,t} = \nabla f_s(x_t), s \sim \mathfrak{D}$. However, a Byzantine node, $i \in g_t \setminus g$, may send any arbitrary vector, $*$.

Iterate Update Rule: The FC performs iterate update using the set g_t computed in the previous iteration (in Step 9)

$$x_{t+1} = x_t - \frac{\eta}{|g_t|}\sum_{i \in g_t} \nabla_{i,t}, \tag{5}$$

where η is the step size.

In each iteration, the FC computes an extra stochastic gradient $\nabla_{0,t}$ using the sample function f_s, drawn independent of the samples at the nodes. This is used to compute $\langle \nabla_{0,t}, \nabla_{i,t} \rangle$, for $i \in [w]$, to obtain $A_i^{(t)}$ in Step 10 given as

$$A_i^{(t)} = \sum_{t'=0}^{t} \langle \nabla_{0,t}, \nabla_{i,t} \rangle. \tag{6}$$

Since the statistic $A_i^{(t)}$ can be computed as a running sum, the memory consumption is not high. The FC computes the median of $A_1^{(t)}, \ldots, A_w^{(t)}$ in Step 11 given by

$$A_{med}^{(t)} = \text{med}\left\{A_1^{(t)}, \ldots, A_w^{(t)}\right\}. \tag{7}$$

The FC also computes the vector median of $\{\nabla_{1,t}, \ldots, \nabla_{w,t}\}$ as any vector $\nabla_{i,t}$, for $i \in [w]$ such that

$$|j \in [w] : \|\nabla_{j,t} - \nabla_{i,t}\| \leq 2v| > w/2$$

in Step 12. We denote this vector median as ∇_{med}^t. The two statistics defined above, $A_{med}^{(t)}$ and ∇_{med}^t, are used to design the filtering rule to prune the Byzantines, as discussed next.

Byzantine Filtering Rule: At the beginning of the algorithm, the FC assumes all the nodes in the network to be normal, hence $g_0 = [w]$. At each iteration, the FC computes the set g_t consisting of the *normal* nodes as the following:

$$g_{t+1} = \Big\{i \in g_t : |A_i^{(t)} - A_{med}^{(t)}| \leq 2\mathfrak{T}_A \wedge \|\nabla_{i,t} - \nabla_{med}^t\| \leq 4v\Big\}, \tag{8}$$

where the threshold $\mathfrak{T}_A$ is defined in Step 1 of the algorithm. The FC expects that the normal nodes satisfy the following criteria.

We present the Pinelis' inequality [20] below which states that the average of the sum of random variables converges to zero.

Lemma 3.1. *Let the sequence of random variables $X_1, X_2, \ldots, X_N \in \mathbb{R}^d$ represent a random process such that we have $\|X_n\| \leq M$ and $\mathbb{E}[X_n | X_1, \ldots, X_{n-1}] = 0$. Then,*

$$\mathbb{P}\left[\|X_1 + \ldots + X_N\|^2 \leq 2\log(2/\delta)M^2N\right] \geq 1 - \delta.$$

Firstly, for the normal nodes, the statistic $A_i^{(t)}$ approaches its true value ($\sum_{t'=0}^{t} \|\nabla f(x_t)\|^2$) as the number of iterations t increases, with high probability from the Pinelis' inequality stated above. Since all the normal nodes sample functions from the same distribution $\mathfrak{D}$, and the fraction of Byzantines $\alpha < 1/2$, the median of A_i^t will be close to the true value. Hence, the rule

$$|A_i^{(t)} - A_{med}^{(t)}| \leq 2\mathfrak{T}_A. \tag{9}$$

Next, for the normal nodes, the gradient $\nabla_{i,t}$ is close to the true gradient ($\nabla f(x_t)$). The vector median ∇_{med}^t will be close to the true gradient as the functions are sampled from the same distribution $\mathfrak{D}$ at the normal nodes and the fraction of Byzantines, $\alpha < 1/2$. Therefore, the rule

$$\|\nabla_{i,t} - \nabla_{med}^t\| \leq 4v. \tag{10}$$

However, note that after computing the set g_t using the update equation in (8), if the cardinality of g_t is less than $(1-\alpha)w$, the FC computes the set g_t as the following:

$$g_{t+1} = \Big\{ i \in g_t : |A_i^{(t)} - A_{med}^{(t)}| \leq 4M^2T \\ \wedge \|\nabla_{i,t} - \nabla_{med}^t\| \leq 4v \Big\}, \tag{11}$$

where the threshold $4M^2T$ is the worst-case bound. The process is repeated at each iteration. The filtering rule described above ensures that the effect of the Byzantine nodes left unfiltered by the FC is negligible. In particular, they do not impact the convergence of the algorithm. This observation is made concrete by the convergence result presented in the next section (Theorem 4.1).

4. CONVERGENCE GUARANTEES

In this section, we present our main result. We state that the proposed algorithm converges even in the presence of Byzantine nodes ($\alpha < 1/2$). Hence, the proposed algorithm exhibits resilience against adversarial attacks. Also, we show that by choosing the step size $\eta = O(1/\sqrt{T})$, the rate of convergence is $\tilde{O}(\frac{1}{\sqrt{wT}} + \frac{\alpha}{\sqrt{T}})$ where the $\tilde{O}$ notation subsumes the logarithmic factors.

Theorem 4.1. *For nonconvex and L-smooth f, let w be the number of nodes. For $\alpha \in [0, 1/2)$, variance bound v and gradient norm bound M (4), and constant step size η, $0 < \mu \leq 1$. Also, $\frac{2wT}{\delta} \geq \exp\frac{T\delta^2}{(1-\delta)^2}$ and $\delta \leq \min\left\{\frac{1}{25w}, \frac{Lv}{3M^2}\sqrt{\frac{1+25\alpha^2 w}{w}}\right\}$, and x_* is an optimal point for* (1). *Then,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \leq \frac{2(f(x_0) - f(x_*))}{\eta T} \\ + \frac{32\alpha M^2\sqrt{2\log(\frac{2wT^{\frac{3}{2}}}{\mu})}}{(1-\alpha)\sqrt{T}} + \frac{8v^2L\eta}{(1-\alpha)^2}\left(\frac{2}{w} + 25\alpha^2\right). \tag{12}$$

Here, for $\eta = \xi/\sqrt{T}$ where $\xi = \sqrt{\frac{(f(x_0)-f(x_*))(1-\alpha)^2 w}{4v^2L\left(1+25\alpha^2 w\right)}}$, we have

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \leq \tilde{O}\left(\frac{1}{\sqrt{wT}} + \frac{\alpha}{\sqrt{T}}\right), \tag{13}$$

where $\tilde{O}(\cdot)$ notation subsumes all constants and logarithmic factors.

Proof. We omit the proof due to space constraints. □

Next, we show that in the absence of Byzantine nodes ($\alpha = 0$), the proposed algorithm achieves the best known rate of convergence (up to logarithmic factors) for distributed nonconvex learning, namely $\tilde{O}\left(\frac{1}{\sqrt{wT}}\right)$.

Corollary. *For the Byzantine-free case, where $\alpha = 0$, we have*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \leq \tilde{O}\left(\frac{1}{\sqrt{wT}}\right).$$

which is the best known result for the convergence of distributed stochastic nonconvex optimization [18], [19].

Proof. By substituting $\alpha = 0$ in Theorem 4.1, we obtain the result. □

5. CONCLUSION

We proposed a Byzantine-resilient variant of the distributed SGD algorithm with nonconvex objective function in an adversarial setting where α-fraction are Byzantine nodes. We proposed a novel aggregation rule and the effect of Byzantine nodes on the performance of the algorithm was analyzed. We analyzed the rate of convergence of the proposed algorithm. The avenues for future work for the Byzantine-resilient variant of the distributed SGD algorithm include analyzing the rates of convergence for the minibatch case, for variable step sizes, and extending to the generalized Byzantine model.

6. REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*, Cambridge University Press, 2011.

[4] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[5] X. Lian, C. Zhang, H. Zhang, C. J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[6] R. Gemulla, E. Nijkamp, P. J. Haas, and 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.

[7] 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.

[8] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[9] 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.

[10] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[11] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*, 2018, pp. 5636–5645.

[12] P. Blanchard, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[13] E. El-Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*, 2018.

[14] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signsgd with majority vote is communication efficient and byzantine fault tolerant," *arXiv preprint arXiv:1810.05291*, 2018.

[15] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 1544–1551.

[16] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2018, pp. 4613–4623.

[17] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, 2018.

[18] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *International Conference on Machine Learning*, 2019, pp. 7184–7193.

[19] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Advances in Neural Information Processing Systems*, 2018, pp. 2525–2536.

[20] I. Pinelis, "Optimum bounds for the distributions of martingales in banach spaces," *The Annals of Probability*, vol. 22, no. 4, pp. 1679–1706, 1994.