Stochastic Gradient Coding for Flexible Straggler Mitigation in Distributed Learning

Rawad Bitar, Mary Wootters, and Salim El Rouayheb
rawad.bitar@rutgers.edu, marykw@stanford.edu, salim.elrouayheb@rutgers.edu.

***Abstract*—We consider distributed gradient descent in the presence of stragglers. Recent work on *gradient coding* and *approximate gradient coding* have shown how to add redundancy in distributed gradient descent to guarantee convergence even if some workers are slow or non-responsive. In this work we propose a new type of approximate gradient coding which we call *Stochastic Gradient Coding* (SGC). The idea of SGC is very simple: we distribute data points redundantly to workers according to a good combinatorial design. We prove that the convergence rate of SGC mirrors that of batched Stochastic Gradient Descent (SGD) for the ℓ_2 loss function, and show how the convergence rate can improve with the redundancy. We show empirically that SGC requires a small amount of redundancy to handle a large number of stragglers and that it can outperform existing approximate gradient codes when the number of stragglers is large.**

I. Introduction

This paper studies distributed variant of gradient descent. Let $X \in \mathbb{R}^{m\times \ell}$ be a data matrix and let $\boldsymbol{y} \in \mathbb{R}^m$ be a vector representing labels for the rows of X. Define $A = [X|\boldsymbol{y}]$ to be the concatenation of X and $\boldsymbol{y}$ and let $\mathbf{a}_i$ denote the i'th row of A. For a loss function $\mathcal{L}$, the goal is to find a vector $\boldsymbol{\beta}^* \in \mathbb{R}^\ell$ that best represents the data X as a function of the labels $\boldsymbol{y}$, i.e., $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \mathcal{L}(A, \boldsymbol{\beta})$. The example we will focus on is linear regression: that is, the loss function is $\mathcal{L}([X|\boldsymbol{y}], \boldsymbol{\beta}) = \|X\boldsymbol{\beta} - \boldsymbol{y}\|_2^2$. In gradient descent, the optimum $\boldsymbol{\beta}^*$ is computed by iteratively performing updates like:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma_t \nabla \mathcal{L}(A, \boldsymbol{\beta}_t). \tag{1}$$

When the loss function can be written as a sum over the rows $\boldsymbol{a}_i$ of A (that is, $\mathcal{L}(A, \boldsymbol{\beta}) = \sum_{i=1}^m \mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta})$, as is the case with the ℓ_2 loss function), gradient descent is easily parallelizable. In a distributed setting, the master partitions the data matrix A into rows $\mathbf{a}_i$ which are distributed between the workers. Each worker returns some linear combination(s) of the gradients $\nabla\mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta}_t)$ that it can compute, and the master aggregates these together to compute or approximate the update step (1).

We focus on the setting where some of the workers may be *stragglers,* i.e., slow or unresponsive [1]–[9]. A typical approach is to introduce some redundancy: the same piece of data $\mathbf{a}_i$ might be held by several workers. We focus on the following four desiderata:

(A) **Convergence speed.** We would like the error $\|\boldsymbol{\beta}_t - \boldsymbol{\beta}^*\|_2$ to shrink as quickly as possible.
(B) **Redundancy.** We would like to minimize the amount of storage overhead needed between the workers.
(C) **Communication.** We would like to minimize the amount of communication between the master and the workers.
(D) **Flexibility.** In practice, there is a great deal of variability in the number of stragglers over time. We would like an algorithm that degrades gracefully if more stragglers than expected occur.

Much existing work, referred to as *gradient coding* has focused on simulating gradient descent *exactly,* even in the presence of worst-case stragglers, for example [3], [4], [10]. The model is that at each round, an arbitrary set of s workers (for a fixed s) may not respond to the master. The goal is for the master to obtain the same update $\boldsymbol{\beta}_t$ at round t that gradient descent would obtain. However, such schemes require every data vector to be replicated on $s+1$ different workers.

On the other hand, there has also been work on *approximately* simulating gradient descent in order to reduce the redundancy in the system. In this model the number of stragglers is assumed to be random, rather than worst-case. If no redundancy is employed and the stragglers are independent at each round (similar in spirit to the method in [2]), then this becomes a close approximation to Batch SGD, see e.g. [11], [12]. (We will later refer to this algorithm as "Ignore-Stragglers SGD.") However, for convex loss functions it is well known that while Batch SGD does converge to $\boldsymbol{\beta}^*$ the

R. Bitar and S. El Rouayheb are with the ECE department of Rutgers University. M. Wootters is with the CS and EE departments at Stanford University. MW is partially funded by NSF grant CCF-1657049 and by NSF CAREER grant CCF-1844628.

978-1-5386-6900-6/19/$31.00 ©2019 IEEE

convergence is not as fast as that of classical gradient descent [11], [13], [14]. Thus, this approach maintains the good communication cost (C) and improves on (B) and (D), but sacrifices (A), the convergence speed.

This basic idea has been exploited in a line of work known as *approximate gradient coding* [10], [15]–[19]. This line of work studies the data redundancy d needed to tolerate s stragglers and allow the master to compute an approximation of the gradient if more than s workers are stragglers and propose code constructions that achieve this redundancy level.

In this work, we follow the approximate gradient coding framework and we focus on the stochastic straggler model. We assume that each worker is a straggler with probability p.[1] We propose a new construction which we call *Stochastic Gradient Coding* (SGC), and we prove convergence results. The closest code construction to our work is the *Bernoulli Gradient Codes* (BGC) proposed in [15], although that work does not provide a convergence analysis for the whole algorithm. The work of [16] analyzes the convergence speed of another approach of [15] which uses fractional repetition codes. For small redundancy d, the analysis of [16] gives a result where the error decays exponentially with T (the number of iterations) until some noise floor is hit. We describe both of these algorithms in more detail when we compare them empirically to SGC in Section V.

Contributions. We introduce an approach that we call *Stochastic Gradient Coding* (SGC). The master distributes data to the workers, with some redundancy that can vary from data point to data point, so that distribution pattern is *pair-wise balanced* in a sense defined below.[2] Then the algorithm proceeds similarly to the Ignore–Stragglers–SGD algorithm described above.

We provide a rigorous convergence analysis of SGC, which shows that SGC with redundancy $d > 1$, can obtain error bounds where $\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2$ decreases at first exponentially in the number of steps t and then proportionally to $\frac{1}{td}$. This mirrors existing results on SGD (which corresponds to the case $d = 1$), and quantifies the trade-off between replication and error.

We provide numerical simulations comparing SGC to Ignore–Stragglers–SGD (as well as a few other versions of SGD for comparison) and to other approximate gradient coding methods. Our simulations show that indeed SGC improves the accuracy of Ignore–Stragglers–SGD, with far less redundancy than would be required to implement exact gradient descent using coding. In addition, we show empirically that SGC outperforms other approximate gradient coding methods when the straggler rate is high.

Organization. We give a more precise definition of our set-up in Section II. We describe the SGC algorithm in Section III. In Section IV and V, we give a detailed discussion of both our theoretical and empirical results, respectively.

II. Setup

A. Probabilistic model of stragglers

In this paper, we adopt a probabilistic model of stragglers. More precisely, we assume that at every iteration each worker may be a straggler with some probability p, and this is independent between workers and between iterations. This is a strong assumption, but it is a natural starting place.Our probabilistic model is similar to the model in [10], [15]–[19] and is in contrast to the worst-case model assumed by much of the literature on coded computation.

B. Computational model

Our computational model has two stages, a distribution stage and a computation stage.

In the *distribution stage,* the master encodes the data using unequal data repetition code. More precisely, the master can decide to send each row $\mathbf{a}_i$ of A to d_i different workers. We refer to the parameter $d = \frac{1}{m}\sum_{i=1}^{m} d_i$ as the *average redundancy* of the scheme.

The *computation stage* is made up of rounds, each of which contains two repeating steps. In the first step, the master does some local computation and then sends a message to each worker. In the second step, each worker does some local computation and tries to send a message back to the master; however, with probability p the message may not reach the master. Then the round is over and the master repeats the first step to begin the next round. We refer to the total amount of communication per round as the *communication* of the scheme. We allow each worker to send only one message to the master to reduce the communication.

III. Stochastic gradient coding

In this section, we describe our solution, which we call *Stochastic Gradient Coding* (SGC). The idea behind SGC is extremely simple. It is very much like

[1] We note that rather than thinking of workers as being unresponsive with probability p, we could equivalently think about a setting where the master waits for the $1 - p$ fastest fraction of the workers before proceeding to the next iteration. This setting motivates the case of interest to us, where p is relatively large.

[2] We note that a randomized distribution scheme from a suitable distribution also seems to work, and in fact this is what we study in our empirical results.

the Ignore–Stragglers–SGD algorithm described above, except we introduce a small amount of redundancy. We describe the distribution stage and the computation stage of our algorithm below. Our scheme has parameters $d_1, \ldots, d_m$, which control the redundancy of each row, and a parameter γ_t which controls the step size. We will see in the theoretical and numerical analyses how to set these parameters.

Definition 1. *Let n be the total number of workers. We say that a distribution scheme that sends $\mathbf{a}_i$ to d_i different workers is* pair-wise balanced *if for all $i \neq i'$, the number of workers that receives $\mathbf{a}_i$ and $\mathbf{a}_{i'}$ is $\frac{d_i d_{i'}}{n}$.*

Notice that with a completely random distribution scheme, the expected number of workers who receive both $\mathbf{a}_i$ and $\mathbf{a}_{i'}$ for $i \neq i'$ is equal to $\frac{d_i d_{i'}}{n}$. In our analysis, it is convenient to deal with schemes that are exactly pair-wise balanced. However, for small d_i it is clear that no such schemes exist (indeed, we may have $\frac{d_i d_{i'}}{n} < 1$). In our simulations, we choose a uniformly random scheme which seems to work well (see Section V). We believe that our analysis should extend to a random assignment as well, although for simplicity we focus on pair-wise balanced schemes in our theoretical results.

The way SGC works is as follows: **Distribution Stage**. The master creates d_i copies of each row $\mathbf{a}_i$, $i = 1, \ldots, m$, and sends them to d_i distinct workers according to a pair-wise balanced scheme.[3] We denote by S_j, $j = 1, \ldots, n$, the set of indices of the data vectors given to worker $\mathtt{W}_j$, i.e., $S_j = \{i; \mathbf{a}_i \text{ is given to } \mathtt{W}_j\}$. **Computation Stage.** At each iteration t, the master sends $\boldsymbol{\beta}_t$ to all the workers. Each worker $\mathtt{W}_j$ computes

$$f_j(\boldsymbol{\beta}_t) \triangleq \gamma_t \sum_{i \in S_j} \frac{1}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta}_t)$$

and sends the result to the master. The master aggregates all the received answers from non straggler workers, sums them and updates $\boldsymbol{\beta}$ as follows:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma_t \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta}_t),$$

where $\mathcal{I}_i^j$ is the indicator function for worker j being non straggler and having obtained point $\mathbf{a}_i$ during the data distribution. For use below, we define

$$\hat{\mathbf{g}}_t \triangleq \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta}_t). \tag{2}$$

We call $\hat{\mathbf{g}}_t$ the estimate of the gradient at iteration t which estimates the average gradient

$$\mathbf{g}_t \triangleq \sum_{i=1}^{m} \nabla \mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta}_t).$$

IV. Theoretical Results

In this section we precisely state our theoretical results. Inspired by the approach of [20] for SGD, our approach is to consider a *weighted* distribution scheme; that is, we choose d_i proportionally to $\|\mathbf{x}_i\|_2^2$. While the statement below is only for the ℓ_2 loss function, we conjecture that it holds for more general loss functions.

Define a parameter

$$\mu = \frac{\frac{1}{m}\|X\|_F^2}{\|X^T X\|}.$$

This parameter measures how incoherent X is. If X is orthogonal, $\mu = 1$, while if, for example, X is the all-ones matrix, then $\mu = 1/m$. It is not hard to check that $\mu \in [0, 1]$.

Suppose that $\mathcal{D}$ is a pair-wise balanced distribution scheme which sends $\mathbf{a}_i$ to d_i different workers, where

$$d_i = \sigma \cdot \|x_i\|_2^2$$

and where

$$\sigma = \frac{md}{\|X\|_F^2} = \frac{d}{\mu \|X^T X\|} \qquad \text{and} \qquad d = \frac{1}{m} \sum_{i \in [m]} d_i.$$

The parameter d is the average redundancy of the scheme that will control σ and the d_i's. Notice that, as stated, it is possible that the d_i end up being non-integers; in the following, we will assume for simplicity below that $d_i \in \mathbb{N}$ for all i.

Theorem 1. *Consider an SGC algorithm run on a matrix $A \triangleq [X|\mathbf{y}]$ of dimension $m \times (\ell + 1)$ distributed to n workers according to a pair-wise balanced distribution scheme with d_i as described above, with loss function*

$$\mathcal{L}([X|\mathbf{y}], \boldsymbol{\beta}) = \|X\boldsymbol{\beta} - \mathbf{y}\|_2^2,$$

and assume that the degrees $d_i \leq n$ are all integral.

Suppose the stragglers follow the stochastic model of Section II, and that each worker is a straggler independently with probability p. Choose $\varepsilon > 0$ and choose $T \geq 2\log(1/\varepsilon^2)$.

[3] In our simulations, we assign rows to d_i workers uniformly at random, which as discussed above approximates a pair-wise balanced scheme. Similarly, the BGC construction of [15] approximates a pair-wise balanced scheme where each row is assigned to d workers uniformly at random, i.e., $d_i = d$ for $i = 1, \ldots, m$.

Suppose that the number of workers n satisfies $n \geq 8\left(\frac{p}{1-p}\right)$, and that

$$8\mu\left(\frac{p}{1-p}\right) \leq d.$$

Choose a step size

$$\gamma_t = \frac{1}{\|X^T X\|} \cdot \min\left\{\frac{1}{2}, \frac{\log(1/\varepsilon^2)}{t}\right\}.$$

Then, after T iterations of SGC, we have

$$\mathbb{E}\left[\|\boldsymbol{\beta}_T - \boldsymbol{\beta}^*\|_2^2\right] \leq \varepsilon^2 \|\boldsymbol{\beta}_0 - \boldsymbol{\beta}^*\|_2^2 + \frac{1}{dT}\psi,$$

where

$$\psi \triangleq \left(\log^2(1/\varepsilon^2)\left(\frac{p}{1-p}\right)\|\tilde{\mathbf{r}}\|^2 \mu\right),$$

the expectation is over the stragglers in each of the T iterations of SGC, μ is as defined above, and

$$\tilde{\mathbf{r}} = \frac{\|X\boldsymbol{\beta}^* - \boldsymbol{y}\|_2^2}{\|X^T X\|_2}.$$

Proof. The proof can be found in [21]. □

Corollary 2. *Suppose that $X\boldsymbol{\beta}^* = \mathbf{y}$ (that is, we are solving an overdetermined system for which there is a solution) and that $n \geq 8p/(1-p)$. Then the algorithm described in Theorem 1 converges with*

$$\mathbb{E}\left[\|\boldsymbol{\beta}_T - \boldsymbol{\beta}^*\|_2^2\right] \leq \varepsilon^2 \|\boldsymbol{\beta}_0 - \boldsymbol{\beta}^*\|^2$$

provided that $T \geq 2\log(1/\varepsilon^2)$ and $d \geq 8\mu p/(1-p)$.

In particular, since $\mu \leq 1$, this says that we need to take $d \gtrsim p/(1-p)$ and the algorithm converges extremely quickly.

V. Simulation Results

A. Simulation setup

We simulate the performance of SGC on synthetic data X of dimension 1000×100. The data is generated as follows, each row vector $\mathbf{x}_i$ is generated using a Gaussian distribution $\mathcal{N}(0,1)$. We pick a random vector $\bar{\boldsymbol{\beta}}$ with components being integers between 1 and 10 and generate $y_i \sim \mathcal{N}(\langle \mathbf{x}_i, \bar{\boldsymbol{\beta}} \rangle, 1)$. We run linear regression using the ℓ_2 loss function, i.e.,

$$\mathcal{L}(\mathbf{a}_i, \boldsymbol{\beta}_t) = \frac{1}{2}\left(\langle \mathbf{x}_i, \boldsymbol{\beta}_t \rangle - y_i\right)^2.$$

We show simulations for $n = 100$ workers. For each simulation we vary the probability of a worker being a straggler from $p = 0$ to $p = 0.9$ with a step of 0.1. We run the algorithm for 5000 iterations with a variable step size given by $\gamma_t = 7\frac{\ln(10^{100})}{t^{0.7}}$. For all simulations we run each experiment 10 different times and average the results. We fix the average redundancy to be $d = 2$ for all algorithms. For SGC, the d_i's are computed as stated in Section IV.

TABLE I: Summary of the algorithms we compare to SGC.

Algorithm	Brief description
Bernoulli Gradient Code (BGC) [15]	Similar to SGC but all data vectors are replicated d times, i.e., $d_i = 2$ for all $i \in [m]$.
ErasureHead [15], [16]	Partitions the data set equally and sends each partition to d workers. Workers send the sum of the partial gradients to the master who computes the gradient estimate as the sum of distinct received partial gradients divided by total number of data vectors.
Ignore–Stragglers–SGD [2]	Partitions the data among the workers with no redundancy. Workers send the sum of the partial gradients to the master who computes the gradient estimate as the sum of distinct partial gradients divided by the average number of data vectors received per iteration.
SGC–Send–All	Same as SGC with one difference: at each iteration the workers send all the partial gradients to the master. The master computes the gradient estimate as the sum of *distinct* partial gradients divided by the average number of data vectors received per iteration.

We compare SGC to four stochastic gradient descent types of algorithms summarized in Table I. We omit the constructions presented in [10] and [18] because they do not match our setting; the former requires a high redundancy factor d and the latter requires the master to run a decoding algorithm at each iteration.

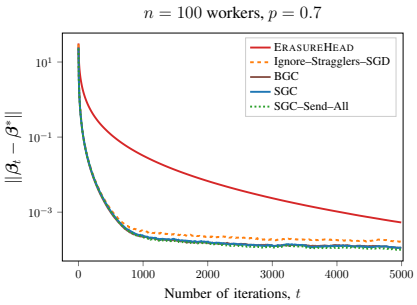

Fig. 1: Evaluation of the considered algorithms as function of the number of iterations when the probability of a worker being straggler is $p = 0.7$.

B. Empirical results

From Figure 1 and similar plots for different values of p, we notice that for all p the convergence rate of SGC exhibits two phases: an exponential decay followed by an error floor. To see the benefit of replication, we compare SGC to Ignore–Stragglers–SGD. Both have the same performance in the exponential phase, but SGC has

a lower error floor due to redundancy. A lower bound on the performance of SGC is SGC–Send–All which has a lower error floor because it computes a better estimate of the gradient at the expense of a higher communication cost. However, as p increases the gap between the two error floors of SGC and SGC–Send–All decreases. In our simulations, we notice the error floor of both algorithms almost match for $p \geq 0.6$ as can be seen in Figure 2.

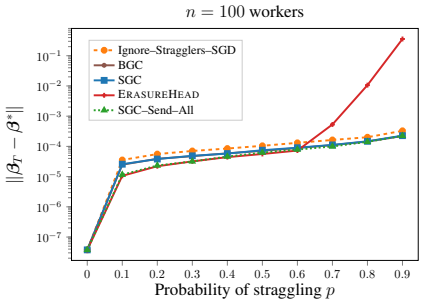

Fig. 2: Final convergence result achieved after running 5000 iterations as function of p the probability of a worker being a straggler.

The interesting phenomena here is that ERASUREHEAD has the best convergence rate for small probability of straggling workers, i.e., the setting it was designed for. However, the convergence rate of ERASUREHEAD deteriorates with the increase of p. In other words, when using SGC the master can decide to wait for the fastest few workers and move to the next iteration which may not be possible for ERASUREHEAD.

VI. CONCLUSION

We present a new code construction that we call *Stochastic Gradient Coding* (SGC). SGC is an approximate gradient code that allows a master to run a distributed gradient descent–based machine learning algorithm distributively on n workers while tolerating stragglers. With small data redundancy, SGC allows the master to wait for the few fastest workers to finish their assigned computations and move to the next iteration. We provide theoretical guarantees on the convergence of SGC for the case of the ℓ_2 loss function and show that SGC has an exponential convergence rate which mirrors the behavior of the well-known stochastic gradient descent. In the extended version, we provide theoretical guarantees for SGC for all strongly convex loss functions and show that SGC has a convergence rate inversely proportional to the number of iterations.

ACKNOWLEDGEMENTS

We thank Deanna Needell for helpful pointers to the literature.

REFERENCES

[1] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.
[2] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," *arXiv preprint arXiv:1604.00981*, 2016.
[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.
[4] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning (ICML)*, pp. 3368–3376, 2017.
[5] M. Amiri and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8177–8181, 2019.
[6] N. Ferdinand and S. Draper, "Anytime stochastic gradient descent: A time to hear from all the workers," in *56th Annual Allerton Conference on Communication, Control, and Computing*, pp. 552–559, 2018.
[7] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 5434–5442, 2017.
[8] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv preprint arXiv:1806.00939*, 2018.
[9] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure distributed computing," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2900–2904, 2017.
[10] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic MDS codes and expander graphs," 2017.
[11] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
[12] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for ℓ_1-regularized loss minimization," *Journal of Machine Learning Research*, vol. 12, no. Jun, pp. 1865–1892, 2011.
[13] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," *arXiv preprint arXiv:1109.5647*, 2011.
[14] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[15] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," *arXiv preprint arXiv:1711.06771*, 2017.
[16] H. Wang, Z. Charles, and D. Papailiopoulos, "ErasureHead: Distributed gradient descent without delays using approximate gradient coding," *arXiv preprint arXiv:1901.09671*, 2019.
[17] S. Wang, J. Liu, and N. Shroff, "Fundamental limits of approximate gradient coding," *arXiv preprint arXiv:1901.08166*, 2019.
[18] S. Horii, T. Yoshida, M. Kobayashi, and T. Matsushima, "Distributed stochastic gradient descent using LDGM codes," *arXiv preprint arXiv:1901.04668*, 2019.
[19] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Robust gradient descent via moment encoding with LDPC codes," *arXiv preprint arXiv:1805.08327*, 2018.
[20] D. Needell and R. Ward, "Batched stochastic gradient descent with weighted sampling," in *International Conference Approximation Theory*, pp. 279–306, 2016.
[21] R. Bitar, M. Wootters, and S. El Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *arXiv preprint arXiv:1905.05383*, 2019.