WEIGHTED GRADIENT CODING WITH LEVERAGE SCORE SAMPLING

Neophytos Charalambides[†], *Mert Pilanci*[‡], *and Alfred O. Hero III*[†],

[†]EECS Department University of Michigan, [‡]EE Department Stanford University

ABSTRACT

A major hurdle in machine learning is scalability to massive datasets. Approaches to overcome this hurdle include compression of the data matrix and distributing the computations. *Leverage score sampling* provides a compressed approximation of a data matrix using an importance weighted subset. *Gradient coding* has been recently proposed in distributed optimization to compute the gradient using multiple unreliable worker nodes. By designing coding matrices, gradient coded computations can be made resilient to stragglers, which are nodes in a distributed network that degrade system performance. We present a novel *weighted leverage score* approach, that achieves improved performance for distributed gradient coding by utilizing an importance sampling.

Index Terms— Reed-Solomon codes, distributed gradient descent, linear regression, sketching, straggler mitigation.

1. INTRODUCTION

In modern day machine learning the *curse of dimensionality* has been a major impediment to solving large scale optimization problems. Gradient methods are widely used in solving such problems, but computing the gradient is often cumbersome. A method for speeding up the gradient computation is by performing the necessary computations in a distributed manner, where a network of workers perform certain subtasks in parallel. A common issue which arises are stragglers: workers whose task may never be received, due to delay or outage. These straggler failures translate to erasures in coding theory. The authors of [1] proposed *gradient coding*, a scheme for exact recovery of the gradient when the objective loss function is additively separable. The exact recovery of the gradient is considered in several prior works, e.g., [1–6], while gradient coding for approximate recovery of the gradient is studied in [6–13]. Gradient coding requires the central server to receive the subtasks of a fixed fraction of any of the workers. We obtain an extension based on balanced Reed-Solomon codes [2, 14], introducing *weighted gradient coding*, where the central server recovers a weighted sum of the partial gradients of the loss function.

Our extension also includes dimensionality reduction via *sketching*, in which a matrix is sampled by row selection based on leverage scores. We adapt the leverage scores subsampling algorithm from [15, 16] to incorporate weights, which we refer to as *weighted leverage score sampling*.

The proposed approach merges weighted gradient coding with weighted leverage score sampling, and significantly benefits from both techniques. The introduction of weights allows for further "compression" in our data matrix when the leverage scores are non-uniform. Our experiments show that the proposed approach succeeds in reducing the number of iterations of gradient descent; without significant sacrifice in performance.

The paper is organized as follows. In section 2 we describe the "straggler problem" in gradient coding [1]. In section 3 we introduce leverage score sampling, and modify existing dimensionality reduction techniques by introducing weights. In 4 we present the weighted gradient coding scheme, which we combine with weighted leverage score sampling. In 5 we show equivalence of the gradients computed using our proposed scheme and those obtained by applying the leverage score sketching matrix [16]. Finally, in 6 we present experimental results. The contributions of this paper are:

- Introduction of a *weighted* gradient coding scheme, that is robust to stragglers.
- Incorporation of leverage score sampling into weighted gradient coding.
- Theory showing that perfect gradient recovery of *leverage score sketching* occurs, under mild assumptions.
- Presentation of experiments that corroborate our theoretical results.

2. STRAGGLERS AND GRADIENT CODING

2.1. Straggler Problem

Consider a single central server node that has at its disposal a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \subsetneq \mathbb{R}^p \times \mathbb{R}$ of N samples, where $\mathbf{x}_i$ represents the features and y_i the label of the i^{th} sample. The central server can distribute the dataset $\mathcal{D}$ among n workers in order to solve the optimization problem

$$\theta^\star = \arg\min_{\theta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \theta) \right\} \tag{1}$$

in an accelerated manner, where $L(\mathcal{D}; \theta) = \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i; \theta)$ is a predetermined loss-function. The objective function in (1)

can also include a regularizer $\lambda R(\theta)$ if necessary. A common approach to solving (1), is to employ gradient descent. Even if closed form solutions exist for (1), gradient descent can still be advantageous for large N.

The central server is capable of distributing the dataset appropriately, with a certain level of redundancy, in order to recover the gradient based on the full $\mathcal{D}$. As a first step we partition the $\mathcal{D}$ into k disjoint parts $\{\mathcal{D}_j\}_{j=1}^k$ each of size N/k. The quantity

$$g = \nabla_\theta L(\mathcal{D};\theta) = \sum_{j=1}^{k} \nabla_\theta \ell(\mathcal{D}_j;\theta) = \sum_{j=1}^{k} g_j$$

is the gradient. We call the summands $g_j \coloneqq \nabla_\theta \ell(\mathcal{D}_j;\theta)$ *partial gradients*.

In the distributed setting each worker node completes its task of sending back a linear combination of its assigned partial gradients. There can be different types of failures that can occur during the computation or the communication process. These failures are what we refer to as *stragglers*, that are ignored by the main server: specifically, the server only receives $f \coloneqq n - s$ completed tasks (s is the number of stragglers our scheme can tolerate). We denote by $\mathcal{I} \subsetneq [n] \coloneqq \{1, \cdots, n\}$ the index set of the f fastest workers who complete their task in time. Once *any* set of f tasks is received, the central server should be able to decode the received encoded partial gradients, and therefore recover the full gradient g.

2.2. Gradient Coding

Gradient coding is a procedure comprised of an encoding matrix $\mathbf{B} \in \mathbb{C}^{n\times k}$, and a decoding vector $\mathbf{a}_\mathcal{I} \in \mathbb{C}^f$; determined by $\mathcal{I}$. Each row of $\mathbf{B}$ corresponds to an encoding vector with support w, and each column corresponds to a data partition $\mathcal{D}_j$. The columns are with support d, so every partition is sent to d workers. By $\mathbf{B}_\mathcal{I} \in \mathbb{C}^{f\times k}$ we denote the submatrix of $\mathbf{B}$ consisting only of the rows indexed by $\mathcal{I}$. The entry $\mathbf{B}_{ij}$ is nonzero if and only if, $\mathcal{D}_j$ was assigned to the i^{th} worker. Furthermore, we have $n \cdot w = k \cdot d$; where $s = d - 1$.

Each worker node is assigned a number of gradients from the partition, indexed by $\mathcal{J}_i \subsetneq [k]$. The worker is tasked to compute an encoded version of the partial gradients $g_j \in \mathbb{R}^p$ corresponding to its assignments. Denote by

$$\mathbf{g} \coloneqq \begin{pmatrix} | & | & & | \\ g_1 & g_2 & \dots & g_k \\ | & | & & | \end{pmatrix}^T \in \mathbb{R}^{k\times p}$$

the matrix whose rows constitute the transposes of the partial gradients, and the received encoded gradients are the rows of $\mathbf{B}_\mathcal{I}\mathbf{g}$. The full gradient of the objective (1) on $\mathcal{D}$ should be recoverable by applying $\mathbf{a}_\mathcal{I}$

$$g^T = \mathbf{a}_\mathcal{I}^T(\mathbf{B}_\mathcal{I}\mathbf{g}) = \mathbf{1}_{1\times k}\mathbf{g} = \sum_{j=1}^{k} g_j^T.$$

Hence the encoding matrix $\mathbf{B}_\mathcal{I}$ must satisfy $\mathbf{a}_\mathcal{I}^T\mathbf{B}_\mathcal{I} = \mathbf{1}_{1\times k}$ for *all* of the $\binom{n}{s}$ possible index sets $\mathcal{I}$.

Every partition will be sent to an equal number of servers d, and each server will receive a total of w distinct partitions. In section 4, we introduce a procedure for recovering a weighted sum of partial gradients, i.e. $\tilde{g} = \sum_{i=1}^{k} \mathbf{w}_i g_i$. Once the gradient has been recovered, the central server can perform an iteration of its gradient based algorithm.

3. WEIGHTED LEVERAGE SCORE SAMPLING

3.1. Leverage Score Sketching

Consider a data matrix $\mathbf{X} \in \mathbb{R}^{N\times p}$ for $N > p$, which is to be compressed through an operation which preserves the objective function in (1). Consider also

$$\theta_{ols}^\star = \arg\min_{\theta\in\mathbb{R}^p} \left\{\|\mathbf{X}\theta - \mathbf{y}\|_2^2\right\}.$$

for $\mathbf{X} \in \mathbb{R}^{N\times p}$ and $\mathbf{y} \in \mathbb{R}^N$. In the case of the over-constrained least squares problem; i.e. $N \gg p$, the dimension N is to be reduced to $r > p$, which corresponds to the number of constraints. Define the *sketching matrix* $\mathbf{S} \in \mathbb{R}^{r\times N}$ and consider the modified problem

$$\tilde{\theta}_{ols} = \arg\min_{\theta\in\mathbb{R}^p} \left\{\|\mathbf{S}(\mathbf{X}\theta - \mathbf{y})\|_2^2\right\}. \tag{2}$$

The leverage scores $\{\ell_i\}_{i=1}^N$ of matrix $\mathbf{X}$ are defined as the diagonal entries of the projection $P_\mathbf{X} = \mathbf{X}\mathbf{X}^\dagger$. Specifically $\ell_i = (P_\mathbf{X})_{ii}$. An equivalent definition of $\{\ell_i\}_{i=1}^N$ is via the reduced left singular vectors matrix $U \in \mathbb{R}^{N\times p}$, yielding $\ell_i = \|U_{(i)}\|_2^2 = \|U_{(i)}\|_F^2$, for $U_{(i)}$ the i^{th} row of U. A distribution is defined over the rows of $\mathbf{X}$ by normalizing these scores, where each row of $\mathbf{X}$ has respective probability of being sampled $\pi_i = \ell_i / \sum_{j=1}^N \ell_j = \ell_i / \|U\|_F^2$. Other methods for approximating the leverage scores are available [17–19], that do not require directly computing the singular vectors. We also note that a multitude of other efficient constructions of sketching matrices and iterative sketching algorithms have been studied in the literature [20–23].

The *leverage score sketching matrix* [15] is comprised of two matrices, a sampling matrix $S_\mathbf{X} \in \{0,1\}^{N\times r}$ and a rescaling matrix $D \in \mathbb{R}^{r\times r}$. The sketching matrix $\tilde{\mathbf{S}}$ is constructed in two steps:

1. randomly sample with replacement $r > p$ rows from $\mathbf{X}$, based on $\{\pi_i\}_{i=1}^N$
2. rescale each sampled row by $\frac{1}{\sqrt{r\pi_i}}$

for which $\tilde{\mathbf{S}} = D \cdot S_\mathbf{X}^T$. The modified least squares problem is then solved to obtain an approximate solution $\tilde{\theta}_{ols}$.

3.2. Weighted Leverage Score Sampling

The sampling procedure described in 3 reduces the size of the dataset. We define the *compression factor* as $\rho \coloneqq \frac{N}{r} \in \mathbb{Z}_+$,

i.e. $r|N$, and we require that $k|r$. Different from section 2.2 where k partitions were considered, here and in the sequel we consider $K = \rho \cdot k$ equipotent partitions $\{\mathcal{D}_i\}_{i=1}^K$ of size N/K, out of which we will draw k *distinct* data partitions. The sampling distribution is defined next.

The leverage score of each sampled partition is defined as the sum of the normalized leverage scores of the samples comprising it. That is; $\Pi_i \coloneqq \sum_{j:x_j \in \mathcal{D}_i} \pi_j$. In a similar manner to the definition of the leverage scores in 3.1 $\Pi_i = \|U_{(\mathcal{K}_i)}\|_F^2 / \|U\|_F^2$, where $U_{(\mathcal{K}_i)}$ denotes the submatrix consisting only of the rows corresponding to the elements in $\mathcal{D}_i$. The proposed compression matrix $\mathbf{S}_\text{p}$ is determined as follows:

1. randomly sample with replacement k parts in the partition of $\{\mathcal{D}_i\}_{i=1}^K$, based on $\{\Pi_i\}_{i=1}^K$
2. retain each sampled part in the partition only once, and count how many times each part was drawn — these counts correspond to entries in the *weight vector* $\mathbf{w}$
3. rescale each part in the partition by $\frac{1}{\sqrt{r\Pi_i}}$.

For the sampling matrix first construct $S_\text{part} \in \{0,1\}^{K \times k}$, which has a single 1 entry in every column, corresponding to the distinct data partitions drawn, and then assign $S_{\mathbf{X}_\text{p}} = S_\text{part} \otimes \mathbf{I}_{N/K} \in \{0,1\}^{N \times r}$. For the rescaling matrix define $D_\text{part} \in \mathbb{R}^{k \times k}$ with diagonal entries $1/\sqrt{r\Pi_j}$ based on the k drawn parts of the partition, and then $D_\text{p} = D_\text{part} \otimes \mathbf{I}_{N/K} \in \mathbb{R}^{r \times r}$. The final compression matrix is $\mathbf{S}_\text{p} \coloneqq D_\text{p} \cdot S_{\mathbf{X}_\text{p}}^T$.

Note that S_part has no repeated columns. Note also that in the proposed distributed computing framework, we do not directly multiply by $\mathbf{S}_\text{p}$, but simply retain the sampled parts of the partition and rescale them before distributing the data.

4. WEIGHTED GRADIENT CODING

The objective is to recover the weighted sum of the partial gradients $\tilde{g}$, as depicted in Figure 1, subject to at most s erasures. This may be achieved by extending the construction proposed in [2]. The main idea in [2] is to use balanced Reed-Solomon codes [14], which are evaluation polynomial codes. Each column of the encoding matrix $\mathbf{B}$ corresponds to a partition $\mathcal{D}_i$ and is associated with a polynomial that evaluates to zero at the respective workers who have not been assigned that partition part. For more details the reader is referred to [2].

Theorem 4.1. *Let $\mathbf{B}$ and $\mathbf{a}_\mathcal{I}$ be an encoding matrix and decoding vector from [2], satisfying $\mathbf{a}_\mathcal{I}^T \mathbf{B}_\mathcal{I} = \mathbf{1}_{1\times k}$ for any $\mathcal{I}$. Let $\tilde{\mathbf{B}} \coloneqq \mathbf{B} \cdot \mathit{diag}(\mathbf{w})$ for any $\mathbf{w} \in \mathbb{C}^{1\times k}$. Then $\mathbf{a}_\mathcal{I}^T \tilde{\mathbf{B}}_\mathcal{I} = \mathbf{w}$.*

Proof. The properties of balanced Reed-Solomon codes imply the decomposition $\mathbf{B}_\mathcal{I} = \mathbf{G}_\mathcal{I}\mathbf{T}$, where $\mathbf{G}_\mathcal{I}$ is a Vandermonde matrix over the subgroup $U_n = \{a \in \mathbb{C} : a^n = 1\}$ of the circle group, and the entries of $\mathbf{T}$ corresponds to the coefficients of polynomials; constructed such that their constant term is 1, i.e. $\mathbf{T}_{(1)} = \mathbf{1}_{1\times k}$. The vector $\mathbf{a}_\mathcal{I}^T$ is the first row of $\mathbf{G}_\mathcal{I}^{-1}$, for which $\mathbf{a}_\mathcal{I}^T \mathbf{G}_\mathcal{I} = \mathbf{e}_1^T$; the first standard basis vector.

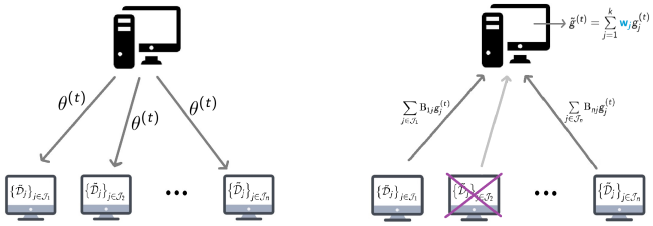

Fig. 1. Schematic representation of communication, with the recovery of $\tilde{g}$ at iteration t of gradient descent.

The matrix $\tilde{\mathbf{T}} = \mathbf{T} \cdot \mathit{diag}(\mathbf{w})$ is equal to $\mathbf{T}$ with its columns each scaled by the respective entry of $\mathbf{w}$, thus $\tilde{\mathbf{T}}_{(1)} = \mathbf{w}$. A direct consequence of this is that $\mathbf{a}_\mathcal{I}^T \tilde{\mathbf{B}}_\mathcal{I} = \mathbf{e}_1^T \tilde{\mathbf{T}} = \tilde{\mathbf{T}}_{(1)} = \mathbf{w}$, which completes the proof. □

When combined with the leverage score sampling scheme, the expected time for computing the weighted gradient per iteration reduces by a factor of ρ.

It is worth noting that weighted gradient coding has applications in other settings, e.g. if the partitions are drawn from noisy sources of different variance; one could select $\mathbf{w}$ based on the estimated noise variances, obtaining improved gradient resiliency. This also directly relates to scenarios where heteroskedastic data is considered.

5. EQUIVALENCE OF GRADIENTS

In this section we show that the proposed weighted scheme will satisfy the same properties as the matrix $\tilde{\mathbf{S}}$ from section 3, when gradient based methods are used to solve (2). This is a consequence of theorem 5.1, applied to the least squares problem. The main reason for this equivalence property is that the weighted gradient $\tilde{g}$ matches the gradient when the leverage score sketching matrix is applied.

The pre-processing in section 3.2 which takes place on the data matrix can be accomplished by using another compression matrix $\hat{\mathbf{S}}$. This matrix is defined as

$$\hat{\mathbf{S}} \coloneqq \sqrt{\mathbf{W}} \cdot \left(D_\text{p} \cdot S_{\mathbf{X}_\text{p}}^T \right) = \sqrt{\mathbf{W}} \cdot \mathbf{S}_\text{p} \in \mathbb{R}^{r \times N}$$

for $\mathbf{W} = \mathit{diag}(\mathbf{w}) \otimes \mathbf{I}_{N/K} \in \mathbb{R}^{r\times r}$. For the objective function of the optimization problem (2)

$$L_\text{S}(\mathbf{S}, \mathbf{X}, \mathbf{y}; \theta) \coloneqq \sum_{i=1}^{r} \left((\mathbf{S}\mathbf{X}\theta)_i - (\mathbf{S}\mathbf{y})_i \right)^2 \tag{3}$$

we have the following.

Theorem 5.1. *Let $\mathcal{D}_i = \{\mathbf{x}_i\}$ for all $i \in [N]$ and $\sum_{i=1}^k \mathbf{w}_i$ be the total number of random draws used to construct $\tilde{\mathbf{S}}$ and $\hat{\mathbf{S}}$. For L_S as specified by (3):*

$$\nabla_\theta L_\text{S}(\tilde{\mathbf{S}}, \mathcal{D}; \theta) = \nabla_\theta L_\text{S}(\hat{\mathbf{S}}, \mathcal{D}; \theta) \tag{4}$$

under any permutation of the rows of $\tilde{\mathbf{S}}$ or $\hat{\mathbf{S}}$.

Proof. Denote the index list of sampled parts of the partition by $\mathcal{S}$, and their index set by $\bar{\mathcal{S}} \subseteq [N]$, i.e. $\mathcal{S}$ has elements with multiplicity equal to $\mathbf{w}_i$ and every element of $\bar{\mathcal{S}}$ is distinct. By assumption $r = \sum_{i=1}^{k} \mathbf{w}_i$, and $|\bar{\mathcal{S}}| = k \leq |\mathcal{S}| = r$. Further note that $\hat{\mathbf{S}}^T \hat{\mathbf{S}} = \mathbf{S}_{\mathrm{p}}^T \mathbf{W} \mathbf{S}_{\mathrm{p}}$, and for the loss function (4)

$$\begin{aligned}
\nabla_\theta L_{\mathrm{S}}(\tilde{\mathbf{S}}, \mathcal{D}; \theta) &= 2\mathbf{X}^T \left(\tilde{\mathbf{S}}^T \tilde{\mathbf{S}}\right) (\mathbf{X}\theta - \mathbf{y}) \\
&= 2 \sum_{l \in \mathcal{S}} \mathbf{x}_l \cdot D_{ll}^2 \cdot \left(\mathbf{x}_l^T \theta - y_l\right) \\
&= 2 \sum_{j \in \bar{\mathcal{S}}} \mathbf{w}_j \mathbf{x}_j \cdot (D_{\mathrm{p}})_{jj}^2 \cdot \left(\mathbf{x}_j^T \theta - y_j\right) \\
&= 2 \sum_{j \in \bar{\mathcal{S}}} \mathbf{x}_j \cdot \left(\sqrt{\mathbf{w}_j} \cdot (D_{\mathrm{p}})_{jj}\right)^2 \cdot \left(\mathbf{x}_j^T \theta - y_j\right) \\
&= 2 \sum_{j \in \bar{\mathcal{S}}} \mathbf{x}_j \cdot \left(\sqrt{\mathbf{W}} \cdot D_{\mathrm{p}}\right)_{jj}^2 \cdot \left(\mathbf{x}_j^T \theta - y_j\right) \\
&= 2\mathbf{X}^T \left(\hat{\mathbf{S}}^T \hat{\mathbf{S}}\right) (\mathbf{X}\theta - \mathbf{y}) \\
&= \nabla_\theta L_{\mathrm{S}}(\hat{\mathbf{S}}, \mathcal{D}; \theta)
\end{aligned}$$

completing the proof. □

Corollary 5.2. *If $\tilde{\mathbf{S}}$ and $\hat{\mathbf{S}}$ were to be constructed by sampling partitions of more than one elements based on $\{\Pi_i\}_{i=1}^K$, conclusion (4) of theorem 5.1 remains valid.*

The benefit of the proposed weighted procedure, is that the weights allow further compression of the data matrix $\mathbf{X}$; without affecting the recovery of the gradient. If $\{\pi_i\}_{i=1}^N$ and $\{\Pi_i\}_{i=1}^K$ are not close to being uniform, $\hat{\mathbf{S}}(\mathbf{X}-\mathbf{y})$ could have significantly fewer rows than $\tilde{\mathbf{S}}(\mathbf{X}-\mathbf{y})$. Under the conditions of theorem 5.1, the proposed matrix $\hat{\mathbf{S}}$ is applicable to the sketching procedures in [15, 16, 24, 25].

A convergence result can also be established using [24] theorem 10. In particular, under appropriate assumptions, after $O(1/\varepsilon)$ iterations with termination criterion $\|\tilde{g}^{(t)}\|_2 \leq \varepsilon$, the proposed weighted gradient coding procedure applied to (3) produces an ε-approximation to $\theta_{ols}^\star$.

6. EXPERIMENTS

6.1. Binary Classification of MNIST

For dataset partitions $k = 20$, number of workers $n = 50$, parts per worker $w = 12$, and allocation of each part to distinct worker $d = 30$, we trained a logistic regression model by applying gradient descent with the proposed method. The number of training samples was $N = 10000$ and the procedure was tested on 1791 samples, for classifying images of four and nine from MNIST, of dimension $p = 784$. The table below shows averaged results over six runs while varying ρ, when the weights were introduced and when they were not.

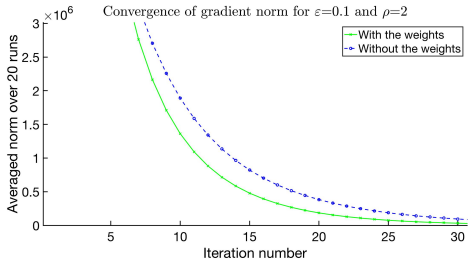

Fig. 2. Convergence of the gradient norm in 6.2 for $\rho = 2$, with and without the weights, where the norm at each iteration is an average of 20 different runs.

With weights		
ρ	Error	Iter.
4	7.09%	25.5
10	7.78%	14.67
20	8.17%	12.5

Without weights		
ρ	Error	Iter.
4	9.08%	24.67
10	8.32%	14.67
20	8.5%	12.67

Without any compression there was an average error of 4.37%. For $\rho = 4, 10$; $\mathbf{w}$ becomes non-uniform and weighting results in better classification accuracy. As ρ decreases, the distribution $\{\Pi_i\}_{i=1}^K$ becomes closer to uniform, leading to reduced advantage in using weighted gradient coding.

6.2. Linear Regression

We retain the same setting of $k = 20, n = 50, w = 12$ and $d = 30$ from 6.1, and generate random data matrices with varying leverage scores as follows for the proposed weighted coding procedure. For all $i \in [20]$ we generate $\mathbf{X}_i \in \mathbb{Z}^{50 \times 20}$ random matrices with entries from $\mathrm{Uni}(-15i, 15i)$, concatenate them and shuffle the rows to form $\mathbf{X} \in \mathbb{Z}^{1000 \times 20}$. We then select an arbitrary $\mathbf{y} \in \mathrm{im}(\mathbf{X})$, and add standard Gaussian noise $\vec{\epsilon}$. We ran experiments to compare the proposed weighted procedure for solving (2), for a fixed gradient descent step-size of $\alpha_t = 10^{-7}$, with the same termination criterion $\|\nabla_\theta L_{\mathrm{S}}(\hat{\mathbf{S}}, \mathcal{D}; \theta^{(t)})\|_2 < 0.1$ over 20 runs and error measure $\left\|\tilde{\theta}_{ols} - \mathbf{X}^\dagger (\mathbf{y} + \vec{\epsilon})\right\|_2^2$.

Average Number of Iterations and Error			
ρ	Weighted	Unweighted	Error
2	107.55	145.6	$O(10^{-5})$
4	75.3	84.4	$O(10^{-4})$

As in 6.1, for lower ρ the proposed weighted gradient coding approach achieves the same order of error as the unweighted, in fewer gradient descent iterations (on average).

In Figure 2 we demonstrate the benefit of weighted versus unweighted. Even though the weights introduce a much higher gradient norm at first, it drops much faster and the termination criterion is met in fewer iterations.

7. REFERENCES

[1] Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *International Conference on Machine Learning*, pages 3368–3376, 2017.

[2] Wael Halbawi, Navid Azizan, Fariborz Salehi, and Babak Hassibi. Improving distributed gradient descent using Reed-Solomon codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2027–2031. IEEE, 2018.

[3] Emre Ozfatura, Deniz Gunduz, and Sennur Ulukus. Gradient coding with clustering and multi-message communication. *arXiv preprint arXiv:1903.01974*, 2019.

[4] Min Ye and Emmanuel Abbe. Communication-computation efficient gradient coding. *arXiv preprint arXiv:1802.03475*, 2018.

[5] Neophytos Charalambides, Hessam Mahdavifar, and Alfred O. Hero. Numerically stable binary gradient coding. *arXiv preprint arXiv:2001.11449*, 2020.

[6] Netanel Raviv, Itzhak Tamo, Rashish Tandon, and Alexandros G Dimakis. Gradient coding from cyclic MDS codes and expander graphs. *arXiv preprint arXiv:1707.03858*, 2017.

[7] Zachary Charles and Dimitris Papailiopoulos. Gradient coding via the stochastic block model. *arXiv preprint arXiv:1805.10378*, 2018.

[8] Zachary Charles, Dimitris Papailiopoulos, and Jordan Ellenberg. Approximate gradient coding via sparse random graphs. *arXiv preprint arXiv:1711.06771*, 2017.

[9] Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Erasurehead: Distributed gradient descent without delays using approximate gradient coding. *arXiv preprint arXiv:1901.09671*, 2019.

[10] Rawad Bitar, Mary Wootters, and Salim El Rouayheb. Stochastic gradient coding for flexible straggler mitigation in distributed learning.

[11] Sinong Wang, Jiashang Liu, and Ness Shroff. Fundamental limits of approximate gradient coding. *arXiv preprint arXiv:1901.08166*, 2019.

[12] Swanand Kadhe, O Ozan Koyluoglu, and Kannan Ramchandran. Gradient coding based on block designs for mitigating adversarial stragglers. *arXiv preprint arXiv:1904.13373*, 2019.

[13] Shunsuke Horii, Takahiro Yoshida, Manabu Kobayashi, and Toshiyasu Matsushima. Distributed stochastic gradient descent using LDGM codes. *arXiv preprint arXiv:1901.04668*, 2019.

[14] Wael Halbawi, Zihan Liu, and Babak Hassibi. Balanced Reed-Solomon codes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 935–939. IEEE, 2016.

[15] Ping Ma, Michael W Mahoney, and Bin Yu. A statistical perspective on algorithmic leveraging. *The Journal of Machine Learning Research*, 16(1):861–911, 2015.

[16] Shusen Wang. A practical guide to randomized matrix computations with matlab implementations. *arXiv preprint arXiv:1505.07570*, 2015.

[17] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

[18] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems*, pages 5672–5682, 2018.

[19] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[20] Mert Pilanci. *Fast Randomized Algorithms for Convex Optimization and Statistical Estimation*. PhD thesis, UC Berkeley, 2016.

[21] Mert Pilanci and Martin J. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Trans. Info. Theory*, 9(61):5096–5115, 2015.

[22] Mert Pilanci and Martin J Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

[23] Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

[24] Michael W Mahoney. Lecture notes on randomized linear algebra. *arXiv preprint arXiv:1608.04481*, 2016.

[25] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.