Journal of Machine Learning Research 20 (2019) 1-47 Submitted 3/18; Revised 4/19; Published 4/19

# Redundancy Techniques for Straggler Mitigation in Distributed Optimization and Learning

**Can Karakus** CAKARAK@AMAZON.COM
*Amazon Web Services*
*East Palo Alto, CA 94303, USA*

**Yifan Sun** YSUN13@CS.UBC.CA
*Department of Computer Science*
*University of British Columbia*
*Vancouver, BC, Canada*

**Suhas Diggavi** SUHAS@EE.UCLA.EDU
*Department of Electrical and Computer Engineering*
*University of California, Los Angeles*
*Los Angeles, CA 90095, USA*

**Wotao Yin** WOTAOYIN@MATH.UCLA.EDU
*Department of Mathematics*
*University of California, Los Angeles*
*Los Angeles, CA 90095, USA*

**Editor:** Tong Zhang

## Abstract

Performance of distributed optimization and learning systems is bottlenecked by "straggler" nodes and slow communication links, which significantly delay computation. We propose a distributed optimization framework where the dataset is "encoded" to have an overcomplete representation with built-in redundancy, and the straggling nodes in the system are dynamically treated as missing, or as "erasures" at every iteration, whose loss is compensated by the embedded redundancy. For quadratic loss functions, we show that under a simple encoding scheme, many optimization algorithms (gradient descent, L-BFGS, and proximal gradient) operating under data parallelism converge to an approximate solution even when stragglers are ignored. Furthermore, we show a similar result for a wider class of convex loss functions when operating under model parallelism. The applicable classes of objectives covers several popular learning problems such as linear regression, LASSO, support vector machine, collaborative filtering, and generalized linear models including logistic regression. These convergence results are deterministic, *i.e.*, they establish sample path convergence for arbitrary sequences of delay patterns or distributions on the nodes, and are independent of the tail behavior of the delay distribution. We demonstrate that equiangular tight frames have desirable properties as encoding matrices, and propose efficient mechanisms for encoding large-scale data. We implement the proposed technique on Amazon EC2 clusters, and demonstrate its performance over several learning problems, including matrix factorization, LASSO, ridge regression and logistic regression, and compare the proposed method with uncoded, asynchronous, and data replication strategies.

∗. Work done while at University of California, Los Angeles.

**Keywords:** Distributed optimization, straggler mitigation, proximal gradient, coordinate descent, restricted isometry property

## 1. Introduction

Solving learning and optimization problems at present scale often requires parallel and distributed implementations to deal with otherwise infeasible computational and memory requirements. However, such distributed implementations often suffer from system-level issues such as slow communication and unbalanced computational nodes. The runtime of many distributed implementations are therefore throttled by that of a few slow nodes, called stragglers, or a few slow communication links, whose delays significantly encumber the overall learning task Dean and Barroso (2013). In this paper, we propose a distributed optimization framework based on proceeding with each iteration without waiting for the stragglers, and encoding the dataset across nodes to add redundancy in the system in order to mitigate the resulting potential performance degradation due to lost updates.

We consider the master-worker architecture, where the dataset is distributed across a set of worker nodes which directly communicate to a master node to optimize a global objective. The encoding framework consists of an efficient linear transformation (coding) of the dataset that results in an overcomplete representation, which is then partitioned and distributed across the worker nodes. The distributed optimization algorithm is then performed directly on the encoded data, with all worker nodes oblivious to the encoding scheme, *i.e.*, no explicit decoding of the data is performed, and nodes simply solve the effective optimization problem on the encoded data. In order to mitigate the effect of stragglers, in each iteration, the master node only waits for the first $k$ updates to arrive from the $m$ worker nodes (where $k \leq m$ is a design parameter) before moving on; the remaining $m - k$ node results are treated as missing (erasures), whose loss is compensated by the data encoding.

The framework is applicable to both the data parallelism and model parallelism paradigms of distributed learning, and can be applied to distributed implementations of several popular optimization algorithms, including gradient descent, limited-memory-BFGS, proximal gradient, and block coordinate descent. Under data parallelism, we focus on quadratic loss functions with arbitrary convex regularization terms or arbitrary convex constraint sets, whereas for model parallelism, we consider general convex objectives where the vector of parameters act linearly on input data, such as logistic regression. We show that if the linear transformation is designed to satisfy a spectral condition resembling the restricted isometry property, the iterates resulting from the encoded version of these algorithms deterministically converge to an exact solution for the case of model paralellism, and an approximate one under data parallelism, where the approximation quality only depends on the properties of encoding and the parameter $k$. These convergence guarantees are deterministic in the sense that they hold for any pattern of node delays, *i.e.*, even if an adversary chooses which nodes to delay at every iteration. In addition, the convergence behavior is independent of the tail behavior of the node delay distribution, and can tolerate unbounded delay. Such a worst-case guarantee is not possible for the asynchronous versions of these algorithms, whose convergence rates deteriorate with increasing node delays. We point out that our approach is particularly suited to computing networks with a high degree of variability and

unpredictability, where a large number of nodes can delay their computations for arbitrarily long periods of time.

Our contributions are as follows: (i) we propose the encoded distributed optimization framework, and prove deterministic convergence guarantees under this framework for gradient descent, L-BFGS, proximal gradient and block coordinate descent algorithms; (ii) we provide three classes of encoding matrices, and discuss their properties, and describe how to efficiently encode with such matrices on large-scale data; (iii) we implement the proposed technique on Amazon EC2 clusters and compare their performance to uncoded, replication, and asynchronous strategies for problems such as ridge regression, collaborative filtering, logistic regression, and LASSO. In these tasks we show that in the presence of stragglers, the technique can result in significant speed-ups (specific amounts depend on the underlying system, and examples are provided in Section 5) compared to the uncoded case when we wait for all workers in each iteration, to achieve the same test error.

**Related work.** The approaches to mitigating the effect of stragglers can be broadly classified into three categories: replication-based techniques, asynchronous optimization, and coding-based techniques.

Replication-based techniques consist of either re-launching a certain task if it is delayed, or pre-emptively assigning each task to multiple nodes and moving on with the copy that completes first. Such techniques have been proposed and analyzed in Gardner et al. (2015); Ananthanarayanan et al. (2013); Shah et al. (2016); Wang et al. (2015); Yadwadkar et al. (2016), among others. Our framework does not preclude the use of such system-level strategies, which can still be built on top of our encoded framework to add another layer of robustness against stragglers. However, it is not possible to achieve the worst-case guarantees provided by encoding with such schemes, since it is still possible for both replicas to be delayed.

Perhaps the most popular approach in distributed learning to address the straggler problem is asynchronous optimization, where each worker node asynchronously pushes updates to and fetches iterates from a parameter server independently of other workers, hence the stragglers do not hold up the entire computation. This approach was studied in Recht et al. (2011); Agarwal and Duchi (2011); Dean et al. (2012); Li et al. (2014) (among many others) for the case of data parallelism, and Liu et al. (2015); You et al. (2016); Peng et al. (2016); Sun et al. (2017) for coordinate descent methods (model parallelism). Although this approach has been largely successful, all asynchronous convergence results depend on either a hard bound on the allowable delays on the updates, or a bound on the moments of the delay distribution, and the resulting convergence rates explicitly depend on such bounds. In contrast, our framework allows for completely unbounded delays. Further, as in the case of replication, one can still consider asynchronous strategies on top of the encoding, although we do not focus on such techniques within the scope of this paper.

A more recent line of work that address the straggler problem is based on coding-theory-inspired techniques Tandon et al. (2017); Lee et al. (2018); Dutta et al. (2016); Karakus et al. (2017a,b); Yang et al. (2017); Halbawi et al. (2017); Reisizadeh et al. (2017). Some of these works focus exclusively on coding for distributed linear operations, which are considerably simpler to handle. The works in Tandon et al. (2017); Halbawi et al. (2017) propose coding techniques for distributed gradient descent that can be applied more generally. However,

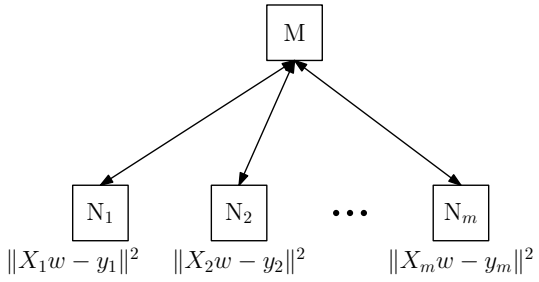


Figure 1: Uncoded distributed optimization with data parallelism, where $X$ and $y$ are partitioned as $X = [X_i]_{i\in[m]}$ and $y = [y_i]_{i\in[m]}$.

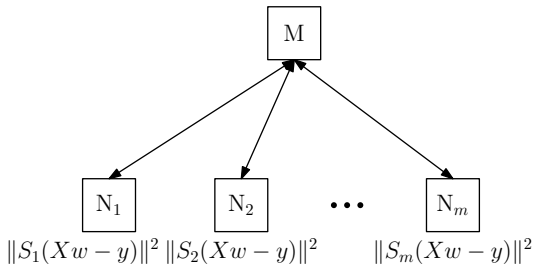


Figure 2: Encoded setup with data parallelism, where node $i$ stores $(S_iX, S_iy)$, instead of $(X_i, y_i)$. The uncoded case corresponds to $S = I$.

the approach proposed in these works require a redundancy factor of $r + 1$ in the code, to mitigate $r$ stragglers. Our approach relaxes the exact gradient recovery requirement of these works, consequently reducing the amount of redundancy required by the code.

The proposed technique, especially under data parallelism, is also closely related to randomized linear algebra and sketching techniques in Mahoney et al. (2011); Drineas et al. (2011); Pilanci and Wainwright (2015), used for dimensionality reduction of large convex optimization problems. The main difference between this literature and the proposed coding technique is that the former focuses on reducing the problem dimensions to lighten the computational load, whereas encoding *increases* the dimensionality of the problem to provide robustness. As a result of the increased dimensions, coding can provide a much closer approximation to the original solution compared to sketching techniques. In addition, unlike these works, our model allows for an arbitrary convex regularizer in addition to the encoded loss term.

## 2. Encoded Distributed Optimization

We will use the notation $[j] = \{i \in \mathbb{Z} : 1 \leq i \leq j\}$. All vector norms refer to 2-norm, and all matrix norms refer to spectral norm, unless otherwise noted. The superscript $^c$ will refer to complement of a subset, *i.e.*, for $A \subseteq [m]$, $A^c = [m]\backslash A$. For a sequence of matrices $\{M_i\}$ and a set $A$ of indices, we will denote $[M_i]_{i\in A}$ to mean the matrix formed by stacking the matrices $M_i$ vertically. The main notation used throughout the paper is provided in Table 1.

We consider a distributed computing network where the dataset $\{(x_i, y_i)\}_{i=1}^n$ is stored across a set of $m$ worker nodes, which directly communicate with a single master node. In practice the master node can be implemented using a fully-connected set of nodes, but this can still be abstracted as a single master node.

It is useful to distinguish between two paradigms of distributed learning and optimization; namely, data parallelism, where the dataset is partitioned across data samples, and model parallelism, where it is partitioned across features (see Figures 1 and 3). We will describe these two models in detail next.

| Notation | Explanation |
|---|---|
| $[j]$ | The set $\{i \in \mathbb{Z} : 1 \leq i \leq j\}$ |
| $m$ | Number of worker nodes |
| $n, p$ | The dimensions of the data matrix $X \in \mathbb{R}^{n \times p}$, vector $y \in \mathbb{R}^{n \times 1}$ |
| $k_t$ | Number of updates the master node waits for in iteration $t$, before moving on |
| $\eta_t$ | Fraction of nodes waited for in iteration, *i.e.*, $\eta_t = \frac{k_t}{m}$ |
| $A_t$ | The subset of nodes $[m]$ which send the fastest $k_t$ updates at iteration $t$ |
| $f(w), \widetilde{f}(w)$ | The original and encoded objectives, respectively, under data parallelism |
| $g(w) = \phi(Xw)$ | The original objective under model parallelism |
| $\widetilde{g}(v) = \phi(XS^\top v)$ | The encoded objective under model parallelism |
| $h(w)$ | Regularization function (potentially non-smooth) |
| $\nu$ | Strong convexity parameter |
| $L$ | Smoothness parameter for $h(w)$ (if smooth), and $g(w)$ |
| $\lambda$ | Regularization parameter |
| $\Psi_t$ | Mapping from gradient updates to step $\{\nabla f_i(t)\}_{i \in A_t} \mapsto d_t$ |
| $d_t$ | Descent direction chosen by the algorithm |
| $\alpha_t$, $\alpha$ | Step size |
| $M, \mu$ | Largest and smallest eigenvalues of $X^\top X$, respectively |
| $\beta$ | Redundancy factor ($\beta \geq 1$) |
| $S$ | Encoding matrix with dimensions $\beta n \times n$ |
| $S_i$ | $i$th row-block of $S$, corresponding to worker $i$ |
| $S_A$ | Submatrix of $S$ formed by $\{S_i\}_{i \in A \subseteq [m]}$ stacked vertically |

Table 1: Notation used in the paper.

### 2.1. Data parallelism

We focus on objectives of the form

$$f(w) = \frac{1}{2n}\|Xw - y\|^2 + \lambda h(w), \tag{1}$$

where $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$ are the data matrix and data vector, respectively. We assume each row of $X$ corresponds to a data sample, and the data samples and response variables can be horizontally partitioned as $X = \begin{bmatrix} X_1^\top & X_2^\top & \cdots & X_m^\top \end{bmatrix}^\top$ and $y = \begin{bmatrix} y_1^\top & y_2^\top & \cdots & y_m^\top \end{bmatrix}^\top$. In the uncoded setting, machine $i$ stores the row-block $X_i$ (Figure 1). We denote the largest and smallest eigenvalues of $X^\top X$ with $M > 0$, and $\mu \geq 0$, respectively. We assume $\lambda \geq 0$, and $h(w) \geq 0$ is a convex, extended real-valued function of $w$ that does not depend on data. Since $h(w)$ can take the value $h(w) = \infty$, this model covers arbitrary convex constraints on the optimization. Several important learning problems, such as ridge regression, LASSO, collaborative filtering (solved via alternating minimization), and support vector machine[1] fall within this formulaton.

The encoding consists of solving the proxy problem

$$\widetilde{f}(w) = \frac{1}{2n}\|S\left(Xw - y\right)\|^2 + \lambda h(w) = \frac{1}{2n}\sum_{i=1}^{m} \underbrace{\|S_i\left(Xw - y\right)\|^2}_{f_i(w)} + \lambda h(w), \tag{2}$$

1. The dual formulation of SVM with squared hinge loss can be solved as a quadratic minimization over the simplex (Jaggi (2013); Li et al. (2009))

instead, where $S \in \mathbb{R}^{\beta n \times n}$ is a designed encoding matrix with redundancy factor $\beta \geq 1$, partitioned as $S = \begin{bmatrix} S_1^\top & S_2^\top & \cdots & S_m^\top \end{bmatrix}^\top$ across $m$ machines. Based on this partition, worker node $i$ stores $(S_i X, S_i y)$, and operates to solve the problem (2) in place of (1) (Figure 2). We will denote $\hat{w} \in \arg\min \widetilde{f}(w)$, and $w^* \in \arg\min f(w)$.

In general, the regularizer $h(w)$ can be non-smooth. We will say that $h(w)$ is $L$-smooth if $\nabla h(w)$ exists everywhere and satisfies

$$h(w') \leq h(w) + \langle \nabla h(w), w' - w \rangle + \frac{L}{2}\|w' - w\|^2$$

for some $L > 0$, for all $w, w'$. The objective $f$ is $\nu$-strongly convex if, for all $x, y$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\nu}{2}\|x - y\|^2.$$

Once the encoding is done and appropriate data is stored in the nodes, the optimization process works in iterations. At iteration $t$, the master node broadcasts the current iterate $w_t$ to the worker nodes, and wait for $k_t$ gradient updates $\nabla f_i(w)$ to arrive, corresponding to that iteration, and then chooses a step direction $d_t$ and a step size $\alpha_t$ (based on algorithm $\Psi_t$ that maps the set of gradients updates to a step) to update the parameters. We will denote $\eta_t = \frac{k_t}{m}$. We will also drop the time dependence of $k$ and $\eta$ whenever it is kept constant.

The set of fastest $k_t$ nodes to send gradients for iteration $t$ will be denoted as $A_t$. Once $k_t$ updates have been collected, the remaining nodes, denoted $A_t^c$, are interrupted by the master node[2]. Algorithms 1 and 2 describe the generic mechanism of the proposed distributed optimization scheme at the master node and a generic worker node, respectively.

The intuition behind the encoding idea is that waiting for only $k_t < m$ workers prevents the stragglers from holding up the computation, while the redundancy provided by using a tall matrix $S$ compensates for the information lost by proceeding without the updates from stragglers (the nodes in the subset $A_t^c$).

We next describe the three specific algorithms that we consider under data parallelism, to compute $d_t$.

**Gradient descent.** In this case, we assume that $h(w)$ is $L$-smooth. Then we simply set the descent direction

$$d_t = -\left(\frac{1}{2n\eta}\sum_{i \in A_t} \nabla f_i(w_t) + \lambda \nabla h(w_t)\right).$$

We keep $k_t = k$ constant, chosen based on the number of stragglers in the network, or based on the desired operating regime.

2. If the communication is already in progress at the time when $k_t$ faster gradient updates arrive, the communication can be finished without interruption, and the late update can be dropped upon arrival. Otherwise, such interruption can be implemented by having the master node send an interrupt signal, and having one thread at each worker node keep listening for such a signal.

**Limited-memory-BFGS.** We assume that $h(w) = \|w\|^2$, and assume $\mu + \lambda > 0$. Although L-BFGS is traditionally a batch method, requiring updates from all nodes, its stochastic variants have also been proposed by Mokhtari and Ribeiro (2015); Berahas et al. (2016). The key modification to ensure convergence in this case is that the Hessian estimate must be computed via gradient components that are common in two consecutive iterations, *i.e.*, from the nodes in $A_t \cap A_{t-1}$. We adapt this technique to our scenario. For $t > 0$, define $u_t := w_t - w_{t-1}$, and[3]

$$r_t := \frac{m}{2n\left|A_t \cap A_{t-1}\right|} \sum_{i \in A_t \cap A_{t-1}} \left(\nabla f_i(w_t) - \nabla f_i(w_{t-1})\right).$$

Then once the gradient terms $\{\nabla f_i(w_t)\}_{i \in A_t}$ are collected, the descent direction is computed by $d_t = -B_t \widetilde{g}_t$, where $\widetilde{g}_t = \frac{1}{2\eta n} \sum_{i \in A_t} \nabla f_i(w_t)$, and $B_t$ is the inverse Hessian estimate for iteration $t$, which is computed by

$$B_t^{(\ell+1)} = V_{j_{\ell,t}}^\top B_t^{(\ell)} V_{j_{\ell,t}} + \rho_{j_{\ell,t}} u_{j_{\ell,t}} u_{j_{\ell,t}}^\top, \quad \rho_j = \frac{1}{r_j^\top u_j}, \quad V_j = I - \rho_j r_j u_j^\top$$

with $j_{\ell,t} = t - \widetilde{\sigma} + \ell$, $B_t^{(0)} = \frac{r_t^\top r_t}{r_t^\top u_t} I$, and $B_t := B_t^{(\widetilde{\sigma})}$ with $\widetilde{\sigma} := \min\{t, \sigma\}$, where $\sigma$ is the L-BFGS memory length. Once the descent direction $d_t$ is computed, the step size is determined through exact line search[4]. To do this, each worker node computes $S_i X d_t$, and sends it to the master node. Once again, the master node only waits for the fastest $k_t$ nodes, denoted by $D_t \subseteq [m]$ (where in general $D_t \neq A_t$), to compute the step size that minimizes the function along $d_t$, given by

$$\alpha_t = -\rho \frac{d_t^\top \widetilde{g}_t}{d_t^\top \widetilde{X}_D^\top \widetilde{X}_D d_t}, \tag{3}$$

where $\widetilde{X}_D = [S_i X]_{i \in D_t}$, and $0 < \rho < 1$ is a back-off factor of choice.

**Proximal gradient.** Here, we consider the general case of non-smooth $h(w) \geq 0, \lambda \geq 0$. The descent direction $d_t$ is given by

$$d_t = \arg\min_w \widetilde{F}_t(w) - w_t,$$

where

$$\widetilde{F}_t(w) := \frac{1}{2\eta n} \sum_{i \in A_t} f_i(w_t) + \left\langle \frac{1}{2\eta n} \sum_{i \in A_t} \nabla f_i(w_t), w - w_t \right\rangle + \lambda h(w) + \frac{1}{2\alpha} \|w - w_t\|^2.$$

We keep the step size $\alpha_t = \alpha$ and $k_t = k$ constant.

3. Note that we assume here that the intersection $A_t \cap A_{t-1}$ is non-empty. This can be ensured by adaptively choosing $k_t$ (waiting for sufficiently many workers) so that $A_t \cap A_{t-1} \neq \varnothing$. This is further discussed in Section 3. However, our experiments indicate that convergence is still possible even when this condition does not hold.
4. Note that exact line search is not more expensive than backtracking line search for a quadratic loss, since it only requires a single matrix-vector multiplication.

**Algorithm 1** Generic encoded distributed optimization procedure under data parallelism, at the master node.

Given: $\Psi_t$, a sequence of functions that map gradients $\{\nabla f_i(w_t)\}_{i\in A_t}$ to a descent direction $d_t$
Initialize $w_0$, $\alpha_0$
**for** $t = 1, \dots, T$ **do**
  broadcast $w_t$ to all worker nodes
  wait to receive $k_t$ gradient updates $\{\nabla f_i(w_t)\}_{i\in A_t}$
  send interrupt signal to the nodes in $A_t^c$
  compute the descent direction $d_t = \Psi_t\left(\{\nabla f_i(w_t)\}_{i\in A_t}\right)$
  determine step size $\alpha_t$
  take the step $w_{t+1} = w_t + \alpha_t d_t$
**end for**

**Algorithm 2** Generic encoded distributed optimization procedure under data parallelism, at worker node $i$.

Given: $f_i(w) = \|S_i(Xw - y)\|^2$
**for** $t = 1, \dots, T$ **do**
  wait to receive $w_t$
  **while** not interrupted by master **do**
    compute $\nabla f_i(w_t)$
  **end while**
  **if** computation was interrupted **then**
    continue
  **else**
    send $\nabla f_i(w_t)$
  **end if**
**end for**

### 2.2. Model parallelism

Under the model parallelism paradigm, we focus on objectives of the form

$$\min g(w) := \min_w \phi(Xw) = \min_w \phi\left(\sum_{i=1}^m X_i w_i\right), \tag{4}$$

where the data matrix is partitioned as $X = [X_1\ X_2\ \cdots\ X_m]$, the parameter vector is partitioned as $w = \left[w_1^\top\ w_2^\top\ \cdots\ w_m^\top\right]^\top$, $\phi$ is convex, and $g(w)$ is $L$-smooth. Note that the data matrix $X$ is partitioned horizontally, meaning that the dataset is split across features, instead of data samples (see Figure 3). This class of objectives is applicable to any classification or regression problem with generalized linear models, such as logistic regression, softmax regression, and multinomial regression.

We encode the problem (4) by setting $w = S^\top v$, and solving the problem

$$\min_v \widetilde{g}(v) := \phi\left(XS^\top v\right) = \min_v \phi\left(\sum_{i=1}^m XS_i^\top v_i\right), \tag{5}$$

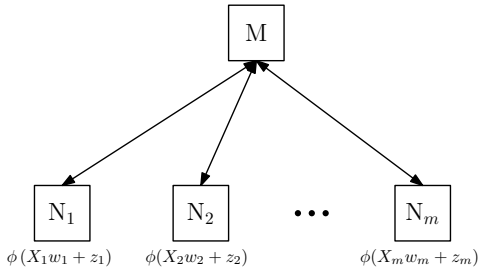


Figure 3: Uncoded distributed optimization with model parallelism, where $i$th node stores the $i$th partition of the model $w_i$. For $i = 1, \ldots, m$, $z_i = \sum_{j \neq i} X_j w_j$.

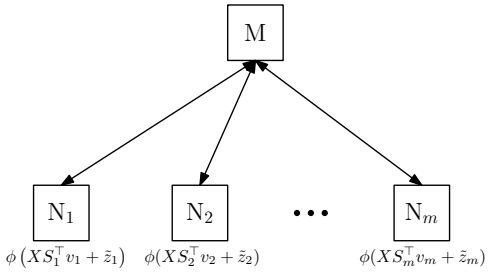


Figure 4: Encoded setup with model parallelism, where $i$th node stores the partition $v_i$ of the model in the "lifted" space. For $i = 1, \ldots, m$, $\widetilde{z}_i = \sum_{j \neq i} u_j = \sum_{j \neq i} XS_j^\top v_j$.

where $w \in \mathbb{R}^p$ and $S^\top = \begin{bmatrix} S_1^\top & S_2^\top & \cdots & S_m^\top \end{bmatrix} \in \mathbb{R}^{p \times \beta p}$ (see Figure 4). As a result, worker $i$ stores the column-block $XS_i^\top$, as well as the iterate partition $v_i$. Note that we increase the dimensions of the parameter vector by multiplying the dataset $X$ with a wide encoding matrix $S^\top$ from the right, and as a result we have redundant coordinates in the system. As in the case of data parallelism, such redundant coordinates provide robustness against erasures arising due to stragglers. Such increase in coordinates means that the problem is simply lifted onto a larger dimensional space, while preserving the original geometry of the problem. We will denote $u_{i,t} = XS_i^\top v_{i,t}$, where $v_{i,t}$ is the parameter iterates of worker $i$ at iteration $t$. In order to compute updates to its parameters $v_i$, worker $i$ needs the up-to-date value of $\widetilde{z}_i := \sum_{j \neq i} u_j$, which is provided by the master node at every iteration.

Let $\mathcal{S} = \arg\min_w g(w)$, and given $w$, let $w^*$ be the projection of $w$ onto $\mathcal{S}$. We will say that $g(w)$ satisfies $\nu$-restricted-strong convexity (Lai and Yin (2013)) if

$$\langle \nabla g(w), w - w^* \rangle \geq \nu \|w - w^*\|^2$$

for all $w$. Note that this is weaker than (implied by) strong convexity since $w^*$ is restricted to be the projection of $w$, but unlike strong convexity, it is satisfied under the case where $\phi$ is strongly convex, but $X$ has a non-trivial null space, *e.g.*, when it has more columns than rows.

For a given $w \in \mathbb{R}^p$, we define the level set of $g$ at $w$ as $D_g(w) := \{w' : g(w') \leq g(w)\}$. We will say that the level set at $w_0$ has diameter $R$ if

$$\sup \left\{ \|w - w'\| : w, w' \in D_g(w_0) \right\} \leq R.$$

As in the case of data parallelism, we assume that the master node waits for $k$ updates at every iteration, and then moves onto the next iteration (see Algorithms 3 and 4). We similarly define $A_t$ as the set of $k$ fastest nodes in iteration $t$, and also define

$$I_{i,t} = \begin{cases} 1 & i \in A_t \\ 0 & i \notin A_t. \end{cases}$$

**Algorithm 3** Encoded block coordinate descent at worker node $i$.

1: Given: $X_i$, $v_i$.
2: **for** $t = 1, \dots, T$ **do**
3: wait to receive $(I_{i,t-1}, \widetilde{z}_{i,t})$
4: **if** $I_{i,t} == 1$ **then**
5: take step $v_{i,t} = v_{i,t-1} + d_{i,t-1}$
6: **else**
7: set $v_{i,t} = v_{i,t-1}$
8: **end if**
9: **while** not interrupted by master **do**
10: compute next step $d_{i,t} = \alpha S_i X^\top \nabla \phi \left( X S_i^\top v_{i,t} + \widetilde{z}_{i,t} \right)$
11: compute $u_{i,t} = X S_i^\top v_{i,t}$
12: **end while**
13: **if** computation was interrupted **then**
14: continue
15: **else**
16: send $u_{i,t}$ to master node
17: **end if**
18: **end for**

**Algorithm 4** Encoded block coordinate descent at the master node.

1: **for** $t = 1, \dots, T$ **do**
2: **for** $i = 1, \dots, m$ **do**
3: send $(I_{i,t-1}, \widetilde{z}_{i,t})$ to worker $i$
4: **end for**
5: wait to receive $k$ updated parameters $\{u_{i,t}\}_{i \in A_t}$
6: send interrupt signal to the nodes in $A_t^c$
7: set $u_{i,t} = u_{i,t-1}$ for $i \in A_t^c$
8: compute $\widetilde{z}_{i,t} = \sum_{j \neq i} u_{j,t}$ for all $i$
9: **end for**

Under model parallelism, we consider block coordinate descent, described in Algorithm 3, where worker $i$ stores the current values of the partition $v_i$, and performs updates on it, given the latest values of the rest of the parameters. The parameter estimate at time $t$ is denoted by $v_{i,t}$, and we also define $\widetilde{z}_{i,t} = \sum_{j \neq i} u_{i,t} = \sum_{j \neq i} X S_j^\top v_j$. The iterates are updated by

$$v_{i,t} - v_{i,t-1} = \Delta_{i,t} := \begin{cases} -\alpha \nabla_i \widetilde{g}(v_{t-1}), & \text{if } i \in A_t \\ 0, & \text{otherwise,} \end{cases}$$

for a step size parameter $\alpha > 0$, where $\nabla_i$ refers to gradient only with respect to the variables $v_i$, *i.e.*, $\nabla \widetilde{g} = [\nabla_i \widetilde{g}]_{i \in [m]}$. Note that if $i \notin A_t$ then $v_i$ does not get updated in worker $i$, which ensures the consistency of parameter values across machines. This is achieved by lines 4–8 in Algorithm 3. Worker $i$ learns about this in the next iteration, when $I_{i,t-1}$ is sent by the master node.

## 3. Main Theoretical Results: Convergence Analysis

In this section, we prove convergence results for the algorithms described in Section 2. Note that since we modify the original optimization problem and solve it obliviously to this change, it is not obvious that the solution has any optimality guarantees with respect to the original problem. We show that, it is indeed possible to provide convergence guarantees in terms of the *original* objective under the encoded setup.

### 3.1. A spectral condition

In order to show convergence under the proposed framework, we require the encoding matrix $S$ to satisfy a certain spectral criterion on $S$. Let $S_A$ denote the submatrix of $S$ associated with the subset of machines $A$, *i.e.*, $S_A = [S_i]_{i\in A}$. Then the criterion in essence requires that for any sufficiently large subset $A$, $S_A$ behaves approximately like a matrix with orthogonal columns. We make this precise in the following statement.

**Definition 1** *Let $\beta \geq 1$, and $\frac{1}{\beta} \leq \eta \leq 1$ be given. A matrix $S \in \mathbb{R}^{\beta n \times n}$ is said to satisfy the $(m, \eta, \epsilon)$-block-restricted isometry property ($(m, \eta, \epsilon)$-BRIP) if for any $A \subseteq [m]$ with $|A| \geq \eta m$,*

$$(1-\epsilon)I_n \preceq \frac{1}{\eta} S_A^\top S_A \preceq (1+\epsilon)I_n. \tag{6}$$

Note that this is similar to the restricted isometry property used in compressed sensing (Candes and Tao (2005)), except that we do not require (6) to hold for every submatrix of $S$ of size $\mathbb{R}^{\eta m \times n}$. Instead, (6) needs to hold only for the submatrices of the form $S_A = [S_i]_{i\in A}$, which is a less restrictive condition. In general, it is known to be difficult to analytically prove that a structured, deterministic matrix satisfies the general RIP condition. Such difficulty extends to the BRIP condition as well. However, it is known that i.i.d. sub-Gaussian ensembles and randomized Fourier ensembles satisfy this property (Candes and Tao (2006)). In addition, numerical evidence suggests that there are several families of constructions for $S$ whose submatrices have eigenvalues that mostly tend to concentrate around 1. We point out that although the strict BRIP condition is required for the theoretical analysis, in practice the algorithms perform well as long as the bulk of the eigenvalues of $S_A$ lie within a small interval $(1-\epsilon, 1+\epsilon)$, even though the extreme eigenvalues may lie outside of it (in the non-adversarial setting). In Section 4, we explore several classes of matrices and discuss their relation to this condition.

### 3.2. Convergence of encoded gradient descent

We first consider the algorithms described under data parallelism architecture. The following theorem summarizes our results on the convergence of gradient descent for the encoded problem.

**Theorem 2** *Let $w_t$ be computed using encoded gradient descent with an encoding matrix that satisfies $(m, \eta, \epsilon)$-BRIP, with step size $\alpha_t = \frac{2\zeta}{M(1+\epsilon)+\lambda L}$ for some $0 < \zeta \leq 1$, for all $t$. Let $\{A_t\}$ be an arbitrary sequence of subsets of $[m]$ with cardinality $|A_t| \geq \eta m$ for all $t$. Then, for $f$ as given in (1),*

*1.*

$$\frac{1}{t}\sum_{\tau=1}^{t} f(w_\tau) - \kappa_1 f(w^*) \leq \frac{4\epsilon f(w_0) + \frac{1}{2\alpha}\|w_0 - w^*\|^2}{(1-7\epsilon)\,t}$$

*2. If $f$ is in addition $\nu$-strongly convex, then*

$$f(w_t) - \frac{\kappa_2^2(\kappa_2-\gamma)}{1-\kappa_2\gamma} f\left(w^*\right) \leq \left(\kappa_2\gamma\right)^t f(w_0), \quad t = 1, 2, \ldots,$$

*where $\kappa_1 = \frac{1+3\epsilon}{1-7\epsilon}$, $\kappa_2 = \frac{1+\epsilon}{1-\epsilon}$, and $\gamma = \left(1 - \frac{4\nu\zeta(1-\zeta)}{M(1+\epsilon)+\lambda L}\right)$, where $\epsilon$ is assumed to be small enough so that $\kappa_2\gamma < 1$.*

The proof is provided in Appendix A, which relies on the fact that the solution to the effective "instantaneous" problem corresponding to the subset $A_t$ lies in a bounded set $\{w : f(w) \leq \kappa f(w^*)\}$ (where $\kappa$ depends on the encoding matrix and strong convexity assumption on $f$), and therefore each gradient descent step attracts the iterate towards a point in this set, which must eventually converge to this set. Theorem 2 shows that encoded gradient descent can achieve the standard $O\left(\frac{1}{t}\right)$ convergence rate for the general case[5], and linear convergence rate for the strongly convex case, up to an approximate minimum. For the convex case, the convergence is shown on the running mean of past function values, whereas for the strongly convex case we can bound the function value at every step. Note that although the nodes actually minimize the encoded objective $\widetilde{f}(w)$, the convergence guarantees are given in terms of the original objective $f(w)$. Note that under the strongly convex case, linear convergence requires condition $\kappa_2\gamma < 1$. Since $\gamma$, which is slightly smaller than 1, is a decreasing function of the fraction of the strong convexity parameter to the smoothness parameter, the practical implication of this is that the less the curvature of the function (the smaller the strong convexity parameter $\nu$), the more workers we need to wait in each iteration, so that $\kappa_2$ is small enough to satisfy $\kappa_2\gamma < 1$, which ensures the linear convergence guarantee of the theorem. Note that even if this condition is not satisfied, the algorithm can still converge, but not necessarily at the linear rate promised by the theorem.

Theorem 2 provides deterministic, sample path convergence guarantees under any (adversarial) sequence of active nodes $\{A_t\}$, which is in contrast to the stochastic methods, which show convergence typically in expectation. Further, the convergence rate is not affected by the tail behavior of the delay distribution, since the delayed updates of stragglers are not applied to the iterates.

Note that since we do not seek exact solutions under data parallelism, we can keep the redundancy factor $\beta$ fixed regardless of the number of stragglers. Increasing number of stragglers in the network simply results in a looser approximation of the solution, allowing for a graceful degradation. This is in contrast to existing work Tandon et al. (2017) seeking exact convergence under coding, which shows that the redundancy factor must grow linearly with the number of stragglers.

5. Since the convergence result is deterministic, the variance reduction factor $\frac{1}{m}$ present in stochastic algorithms does not appear in our convergence rate.

### 3.3. Convergence of encoded L-BFGS

We consider the variant of L-BFGS described in Section 2. For our convergence result for L-BFGS, we need another assumption on the matrix $S$, in addition to (6). Defining $\breve{S}_t = [S_i]_{i\in A_t\cap A_{t-1}}$ for $t > 0$, we assume that for some $\delta > 0$,

$$\delta I \preceq \breve{S}_t^\top \breve{S}_t \tag{7}$$

for all $t > 0$. Note that this requires that one should wait for sufficiently many nodes to send updates so that the overlap set $A_t \cap A_{t-1}$ has more than $\frac{1}{\beta}$ nodes, and thus the matrix $\breve{S}_t$ can be full rank. When the columns of $X$ are linearly independent, this is satisfied if $\eta \geq \frac{1}{2} + \frac{1}{2\beta}$ in the worst-case, and in the case where node delays are i.i.d. across machines, it is satisfied in expectation if $\eta \geq \frac{1}{\sqrt{\beta}}$. One can also choose $k_t$ adaptively so that $k_t = \min\left\{k : |A_t(k) \cap A_{t-1}| > \frac{1}{\beta}\right\}$. We note that although this condition is required for the theoretical analysis, the algorithm may perform well in practice even when this condition is not satisfied.

We first show that this algorithm results in stable inverse Hessian estimates under the proposed model, under arbitrary realizations of $\{A_t\}$ (of sufficiently large cardinality), which is done in the following lemma.

**Lemma 3** *Let $\mu + \lambda > 0$. Then there exist constants $c_1, c_2 > 0$ such that for all $t$, the inverse Hessian estimate $B_t$ satisfies $c_1 I \preceq B_t \preceq c_2 I$.*

The proof, provided in Appendix A, is based on the well-known trace-determinant method. Using Lemma 3, we can show the following convergence result.

**Theorem 4** *Let $\mu + \lambda > 0$, and let $w_t$ be computed using the L-BFGS method described in Section 2, with an encoding matrix that satisfies $(m, \eta, \epsilon)$-BRIP. Let $\{A_t\}, \{D_t\}$ be arbitrary sequences of subsets of $[m]$ with cardinality $|A_t|, |D_t| \geq \eta m$ for all $t$. Then, for $f$ as described in Section 2,*

$$f(w_t) - \frac{\kappa^2(\kappa - \gamma)}{1 - \kappa\gamma} f(w^*) \leq (\kappa\gamma)^t f(w_0),$$

*where $\kappa = \frac{1+\epsilon}{1-\epsilon}$, and $\gamma = \left(1 - \frac{4(\mu+\lambda)c_1c_2}{(M+\lambda)(1+\epsilon)(c_1+c_2)^2}\right)$, where $c_1$ and $c_2$ are the constants in Lemma 3.*

Similar to Theorem 2, the proof is based on the observation that the solution of the effective problem at time $t$ lies in a bounded set around the true solution $w^*$. As in gradient descent, coding enables linear convergence deterministically, unlike the stochastic and multi-batch variants of L-BFGS, *e.g.*, Mokhtari and Ribeiro (2015); Berahas et al. (2016).

### 3.4. Convergence of encoded proximal gradient

Next we consider the encoded proximal gradient algorithm, described in Section 2, for objectives with potentially non-smooth regularizers $h(w)$. The following theorem characterizes our convergence results under this setup.

**Theorem 5** *Let $w_t$ be computed using encoded proximal gradient with an encoding matrix that satisfies $(m, \eta, \epsilon)$-BRIP, with step size $\alpha_t = \alpha < \frac{1}{M}$, and where $\epsilon < \frac{1}{7}$. Let $\{A_t\}$ be an arbitrary sequence of subsets of $[m]$ with cardinality $|A_t| \geq \eta m$ for all $t$. Then, for $f$ as described in Section 2,*

1. *For all $t$,*

$$\frac{1}{t}\sum_{\tau=1}^{t} f(w_\tau) - \kappa f(w^*) \leq \frac{4\epsilon f(w_0) + \frac{1}{2\alpha}\|w_0 - w^*\|^2}{(1-7\epsilon)\, t},$$

2. *For all $t$,*

$$f(w_{t+1}) \leq \kappa f(w_t),$$

*where $\kappa = \frac{1+7\epsilon}{1-3\epsilon}$.*

As in the previous algorithms, the convergence guarantees hold for arbitrary sequences of active nodes $\{A_t\}$. Note that as in the gradient descent case, the convergence is shown on the mean of past function values. Since this does not prevent the iterates from having a sudden jump at a given iterate, we include the second part of the theorem to complement the main convergence result, which implies that the function value cannot increase by more than a small factor of its current value.

### 3.5. Convergence of encoded block coordinate descent

Finally, we consider the convergence of encoded block coordinate descent algorithm. The following theorem characterizes our main convergence result for this case.

**Theorem 6** *Let $w_t = S^\top v_t$, where $v_t$ is computed using encoded block coordinate descent as described in Section 2. Let $S$ satisfy $(m, \eta, \epsilon)$-BRIP, and the step size satisfy $\alpha < \frac{1}{L(1+\epsilon)}$. Let $\{A_t\}$ be an arbitrary sequence of subsets of $[m]$ with cardinality $|A_t| \geq \eta m$ for all $t$. Let the level set of $g$ at the first iterate $D_g(w_0)$ have diameter $R$. Then, for $g(w) = \phi(Xw)$ as described in Section 2, the following hold.*

1. *If $\phi$ is convex, then*

$$g(w_t) - g(w^*) \leq \frac{1}{\frac{1}{\pi_0} + Ct},$$

*where $\pi_0 = g(w_0) - g(w^*)$, and $C = \frac{(1-\epsilon)\alpha}{R^2}\left(1 - \frac{\alpha L'}{2}\right)$.*

2. *If $g$ is $\nu$-restricted-strongly convex, then*

$$g(w_t) - g(w^*) \leq \left(1 - \frac{1}{\xi}\right)^t \left(g(w_0) - g(w^*)\right),$$

*where $\xi = \frac{2}{\nu(1-\epsilon)\alpha}\left(1 - \frac{L(1+\epsilon)\alpha}{2}\right)^{-1}$.*

Theorem 6 demonstrates that the standard $O\left(\frac{1}{t}\right)$ rate for the general convex, and linear rate for the strongly convex case can be obtained under the encoded setup. Similar to the previous cases, encoding allows for deterministic convergence guarantees under adversarial failure patterns.

Note that unlike the data parallelism setup, we can achieve exact minimum under model parallelism, since the underlying geometry of the problem does not change under encoding; the same objective is simply mapped onto a higher-dimensional space, which has redundant coordinates. In every iteration, such redundancy serves as a "soft" error correction mechanism, by allowing each worker to receive sufficient information about the up-to-date values of the rest of the parameters hosted in the other nodes, even though a subset of them are not able to send their updates sufficiently fast. The smaller the $\epsilon$ is in the BRIP condition, the more representative the redundant coordinates are for the missing ones, which is reflected in the convergence rate, as a non-zero $\epsilon$ slightly weakens the constants in the convergence expressions. Still, note that this penalty in convergence rate only depends on the encoding matrix and not on the delay profile in the system. This is in contrast to the asynchronous coordinate descent methods; for instance, in Liu et al. (2015), the step size is required to shrink *exponentially* in the maximum allowable delay, and thus the guaranteed convergence rate can exponentially degrade with increasing worst-case delay in the system. The same is true for the linear convergence guarantee in Peng et al. (2016).

## 4. Code Design

### 4.1. Block RIP condition and code design

We first discuss two classes of encoding matrices with regard to the BRIP condition; namely equiangular tight frames, and random matrices.

**Tight frames.** A unit-norm *frame* for $\mathbb{R}^n$ is a set of vectors $F = \{a_i\}_{i=1}^{n\beta}$ with $\|a_i\| = 1$, where $\beta \geq 1$, such that there exist constants $\xi_2 \geq \xi_1 > 0$ such that, for any $u \in \mathbb{R}^n$,

$$\xi_1 \|u\|^2 \leq \sum_{i=1}^{n\beta} |\langle u, a_i \rangle|^2 \leq \xi_2 \|u\|^2.$$

The frame is *tight* if the above is satisfied with $\xi_1 = \xi_2$. In this case, it can be shown that the constants are equal to the redundancy factor of the frame, *i.e.*, $\xi_1 = \xi_2 = \beta$. If we form $S \in \mathbb{R}^{(\beta n)\times n}$ by rows that form a *tight frame*, then we have $S^\top S = \beta I$, which ensures $\|Xw - y\|^2 = \frac{1}{\beta}\|SXw - Sy\|^2$. Then for any solution $\hat{w}$ to the encoded problem (with $k = m$),

$$\nabla \widetilde{f}(\hat{w}) = X^\top S^\top S(X\hat{w} - y) = \beta X^\top (X\hat{w} - y) = \beta \nabla f(\hat{w}).$$

Therefore, the solution to the encoded problem satisfies the optimality condition for the original problem as well:

$$-\nabla \widetilde{f}(\hat{w}) \in \partial h(\hat{w}), \quad \Leftrightarrow \quad -\nabla f(\hat{w}) \in \partial h(\hat{w}),$$

and if $f$ is also strongly convex, then $\hat{w} = w^*$ is the unique solution. This means that for $k = m$, obliviously solving the encoded problem results in the same objective value as in the original problem.

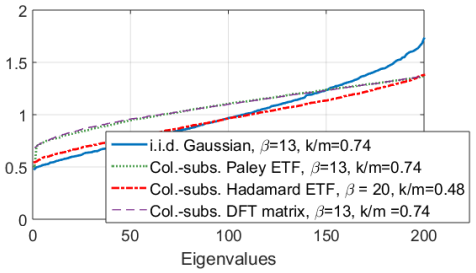


Figure 5: Sample spectrum of $S_A^\top S_A$ for various constructions with high redundancy, and small $k$ (normalized).

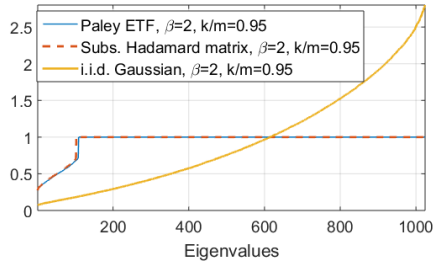


Figure 6: Sample spectrum of $S_A^\top S_A$ for various constructions with moderate redundancy, and large $k$ (normalized).

Define the maximal inner product of a unit-norm tight frame $F = \{a_i\}_{i=1}^{n\beta}$, where $a_i \in \mathbb{R}^n, \forall i$, by

$$\omega(F) := \max_{\substack{a_i, a_j \in F \\ i \neq j}} |\langle a_i, a_j \rangle| .$$

**Definition 7 (Equiangular tight frame (ETF))** *A tight frame is called an* equiangular tight frame *(ETF) if* $|\langle a_i, a_j \rangle| = \omega(F)$ *for every* $i \neq j$.

**Proposition 8 (Welch (1974))** *Let* $F = \{a_i\}_{i=1}^{n\beta}$ *be a tight frame, where* $a_i \in \mathbb{R}^n$ *for* $i = 1, \ldots, n\beta$. *Then* $\omega(F) \geq \sqrt{\frac{\beta - 1}{n\beta - 1}}$. *Moreover, equality is satisfied if and only if* $F$ *is an equiangular tight frame.*

Therefore, an ETF minimizes the correlation between its individual elements, making each submatrix $S_A^\top S_A$ as close to orthogonal as possible. This, combined with the property that tight frames preserve the optimality condition when all nodes are waited for ($k = m$), make ETFs good candidates for encoding, in light of the required property (6). We specifically evaluate the Paley ETF from Paley (1933) and Goethals and Seidel (1967); Hadamard ETF from Szöllősi (2013) (not to be confused with Hadamard matrix); and Steiner ETF from Fickus et al. (2012) in our experiments.

Although the derivation of tight eigenvalue bounds for subsampled ETFs is a long-standing problem, numerical evidence (see Figures 5, 6) suggests that they tend to have their eigenvalues more tightly concentrated around 1 than random matrices (also supported by the fact that they satisfy Welch bound, Proposition 8 with equality).

**Random matrices.** Another natural choice of encoding could be to use i.i.d. random matrices. Although encoding with such random matrices can be computationally expensive and may not have the desirable properties of encoding with tight frames, their eigenvalue behavior can be characterized analytically. In particular, using the existing results on the eigenvalue scaling of large i.i.d. Gaussian matrices from Geman (1980); Silverstein (1985) and union bound, it can be shown that

$$\mathbb{P}\left( \max_{A:|A|=k} \lambda_{\max}\left( \frac{1}{\beta\eta n} S_A^\top S_A \right) > \left( 1 + \sqrt{\frac{1}{\beta\eta}} \right)^2 \right) \to 0 \tag{8}$$

$$\mathbb{P}\left(\min_{A:|A|=k} \lambda_{\min}\left(\frac{1}{\beta\eta n} S_A^\top S_A\right) < \left(1-\sqrt{\frac{1}{\beta\eta}}\right)^2\right) \to 0, \tag{9}$$

as $n \to \infty$, if the elements of $S_A$ are drawn i.i.d. from $N(0,1)$. Hence, for sufficiently large redundancy and problem dimension, i.i.d. random matrices are good candidates for encoding as well. However, for finite $\beta$, even if $k = m$, in general the optimum of the original problem is not recovered exactly, for such matrices.

### 4.2. Discussion on required redundancy

A common problem with worst-case bounds is that they are often pessimistic when compared against actual performance on real instances. In our case, directly computing the redundancy for small enough $\epsilon < 1$ (as required by Theorem 5, for instance) suggests a required redundancy $\beta$ of more than 200. However, as we also show in Section 5, in practice much lower values values of redundancy (such as $\beta = 2$) results in good performance. The reason for this apparent discrepancy is that in the proofs of the theorems, we take a conservative approach and bound quadratic terms of the form

$$\xi(w) = \frac{\left|(Xw-y)^\top (S_A^\top S_A - I)(Xw-y)\right|}{\|Xw-y\|^2},$$

which is at worst

$$\xi(w) \leq \epsilon = \max\left\{\lambda_{\max}\left(S_A^\top S_A - I\right), -\lambda_{\min}\left(S_A^\top S_A - I\right)\right\}.$$

Note that the inequality is tight if $Xw - y$ is an extremal eigenvector of $S_A^T S_A - I$, which, although possible, is highly unlikely. We illustrate this point through an example experiment shown in Figure 7, which plots $\xi(w_t)$ for a run of gradient descent on linear regression with random data, where $w_t$ is the $t^{th}$ iterate. In the entire optimization process, it is clear that $\epsilon$ is a very pessimistic upper bound on $\xi(w_t)$.

In particular, most of the energy of the vector $Xw_t - y$ lies in the eigenspace associated with the bulk of the eigenvalues of $S_A^\top S_A$. From another perspective, if we were to use the "empirical" maximum of the value $\epsilon$ for this particular run (instead of the worst-case bound, the largest eigenvalue), we would achieve $\epsilon = 0.074$ at redundancy $\beta = 2$, a dramatic improvement compared to the above requirement of $\beta > 200$ for $\epsilon = \frac{1}{7}$.

This observation implies that for good practical performance, what is more important is to ensure that most eigenvalues lie close to, or is exactly 1. The following proposition shows that for ETFs, the bulk of the eigenvalues can be identically 1.

**Proposition 9** *If the rows of $S$ are chosen to form an ETF with redundancy $\beta$, then for $\eta \geq 1 - \frac{1}{\beta}$, $\frac{1}{\beta} S_A^\top S_A$ has $n(1-\beta(1-\eta))$ eigenvalues equal to 1.*

This follows immediately from Cauchy interlacing theorem, using the fact that $S_A S_A^\top$ and $S_A^\top S_A$ have the same spectra except zeros. Therefore for sufficiently large $\eta$, ETFs have a mostly flat spectrum even for low redundancy, and thus in practice one would expect ETFs to perform well even for small amounts of redundancy. This is also confirmed by Figure 6, as well as our numerical results.

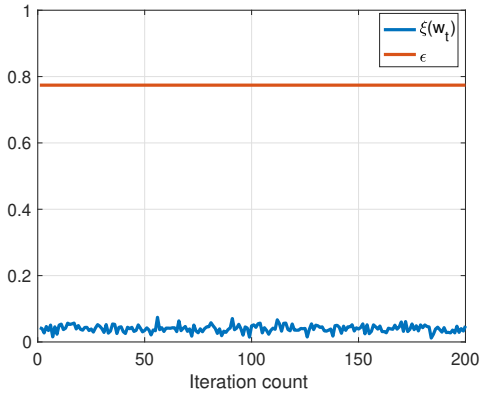


Figure 7: The evolution of $\xi(w_t)$ and $\epsilon$ against iteration count. The parameters are $(n, d, \beta) = (2001, 1400, 2)$, where we encode the data variables with a Paley ETF. We initialize the elements of $w_0$ with a standard normal distribution. We set $\eta = 0.9$, *i.e.*, we wait for 90% of the machines at every step.

### 4.3. Efficient encoding

In this section we discuss some of the possible practical approaches to encoding. Some of the practical issues involving encoding include the computational complexity of encoding, as well as the loss of sparsity in the data due to the multiplication with $S$, and the resulting increase in time and space complexity. We address these issues in this section.

#### 4.3.1. Efficient distributed encoding with sparse matrices

Let the dataset $(X, y)$ lie in a database, accessible to each worker node, where each node is responsible for computing their own encoded partitions $S_i X$ and $S_i y$. We assume that $S$ has a sparse structure. Given $S$, define $B_i(S) = \{j : S_{ij} \neq 0\}$ as the set of indices of the non-zero elements of the $i$th row of $S$. For a set $\mathcal{I}$ of rows, we define $B_{\mathcal{I}}(S) = \cup_{i \in \mathcal{I}} B_i(S)$.

Let us partition the set of rows of $S$, $[\beta n]$, into $m$ machines, and denote the partition of machine $k$ as $\mathcal{I}_k$, *i.e.*, $\bigsqcup_{k=1}^{m} \mathcal{I}_k = [\beta n]$, where $\sqcup$ denotes disjoint union. Then the set of non-zero columns of $S_k$ is given by $B_{\mathcal{I}_k}(S)$. Note that in order to compute $S_k X$, machine $k$ only requires the rows of $X$ in the set $B_{\mathcal{I}_k}(S)$. In what follows, we will denote this submatrix of $X$ by $\widetilde{X}_k$, *i.e.*, if $x_i^\top$ is the $i$th row of $X$, $\widetilde{X}_k := \left[x_i^\top\right]_{i \in B_{\mathcal{I}_k}(S)}$. Similarly $\widetilde{y}_k = [y_i]_{i \in B_{\mathcal{I}_k}(S)}$, where $y_i$ is the $i$th element of $y$.

Consider the specific computation that needs to be done by worker $k$ during the iterations, for each algorithm. Under the data parallelism setting, worker $k$ computes the following gradient:

$$\nabla f_k(w) = X^\top S_k^\top S_k (Xw - y) \overset{(a)}{=} \widetilde{X}_k^\top S_k^\top S_k (\widetilde{X}_k w - \widetilde{y}_k) \tag{10}$$

where (a) follows since the rows of $X$ that are not in $B_{\mathcal{I}_k}$ get multiplied by zero vector. Note that the last expression can be computed without any matrix-matrix multiplication. This gives a natural storage and computation scheme for the workers. Instead of computing $S_k X$ offline and storing it, which can result in a loss of sparsity in the data, worker $k$

can store $\widetilde{X}_k$ in uncoded form, and compute the gradient through (10) whenever needed, using only matrix-vector multiplications. Since $S_k$ is sparse, the overhead associated with multiplications of the form $S_k v$ and $S_k^\top v$ is small.

Similarly, under model parallelism, the computation required by worker $k$ is

$$\nabla_k \widetilde{g}(v) = S_k X^\top \nabla_k \phi\left(X S_k^\top v_k + \widetilde{z}_k\right) = S_k \widetilde{X}_k^\top \nabla_k \phi\left(\widetilde{X}_k S_k^\top v_k + \widetilde{z}_k\right), \tag{11}$$

and as in the data parallelism case, the worker can store $\widetilde{X}_k$ uncoded, and compute (11) online through matrix-vector multiplications.

**Example: Steiner ETF.** We illustrate the described technique through Steiner ETF, based on the construction proposed in Fickus et al. (2012), using $(2, 2, v)$-Steiner systems. Let $v$ be a power of 2, let $H \in \mathbb{R}^{v \times v}$ be a real Hadamard matrix, and let $h_i$ be the $i$th column of $H$, for $i = 1, \ldots, v$. Consider the matrix $V \in \{0, 1\}^{v \times v(v-1)/2}$, where each column is the incidence vector of a distinct two-element subset of $\{1, \ldots, v\}$. For instance, for $v = 4$,

$$V = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Note that each of the $v$ rows have exactly $v - 1$ non-zero elements. We construct Steiner ETF $S$ as a $v^2 \times \frac{v(v-1)}{2}$ matrix by replacing each 1 in a row with a distinct column of $H$, and normalizing by $\sqrt{v-1}$. For instance, for the above example, we have

$$S = \frac{1}{\sqrt{3}} \begin{bmatrix} h_2 & h_3 & h_4 & 0 & 0 & 0 \\ h_2 & 0 & 0 & h_3 & h_4 & 0 \\ 0 & h_2 & 0 & h_3 & 0 & h_4 \\ 0 & 0 & h_2 & 0 & h_3 & h_4 \end{bmatrix}.$$

We will call a set of rows of $S$ that arises from the same row of $V$ a block. In general, this procedure results in a matrix $S$ with redundancy factor $\beta = \frac{2v}{v-1}$. In full generality, Steiner ETFs can be constructed for larger redundancy levels; we refer the reader to Fickus et al. (2012) for a full discussion of these constructions.

We partition the rows of the $V$ matrix into $m$ machines, so that each machine gets assigned $\frac{v}{m}$ rows of $V$, and thus the corresponding $\frac{v}{m}$ blocks of $S$.

This construction and partitioning scheme is particularly attractive for our purposes for two reasons. First, it is easy to see that for any node $k$, $|B_{\mathcal{I}_k}|$ is upper bounded by $\frac{v(v-1)}{m} = \frac{2n}{m}$, which means the memory overhead compared to the uncoded case is limited to a factor[6] of $\beta$. Second, each block of $S_k$ consists of (almost) a Hadamard matrix, so the multiplication $S_k v$ can be efficiently implemented through Fast Walsh-Hadamard Transform.

6. In practice, we have observed that the convergence performance improves when the blocks are broken into multiple machines, so one can, for instance, assign half-blocks to each machine.

**Example: Haar matrix.** Another possible choice of sparse matrix is column-subsampled Haar matrix, which is defined recursively by

$$H_{2n} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_n \otimes [1\ 1] \\ I_n \otimes [1\ -1] \end{bmatrix}, \quad H_1 = 1,$$

where $\otimes$ denotes Kronecker product. Given a redundancy level $\beta$, one can obtain $S$ by randomly sampling $\frac{n}{\beta}$ columns of $H_n$. It can be shown that in this case, we have $|B_{\mathcal{I}_k}| \leq \frac{\beta n \log(n)}{m}$, and hence encoding with Haar matrix incurs a memory cost by logarithmic factor.

#### 4.3.2. Fast transforms

Another computationally efficient method for encoding is to use fast transforms: Fast Fourier Transform (FFT), if $S$ is chosen as a subsampled DFT matrix, and the Fast Walsh-Hadamard Transform (FWHT), if $S$ is chosen as a subsampled real Hadamard matrix. In particular, one can insert rows of zeroes at random locations into the data pair $(X, y)$, and then take the FFT or FWHT of each column of the augmented matrix. This is equivalent to a randomized Fourier or Hadamard ensemble, which is known to satisfy the RIP with high probability by Candes and Tao (2006). However, such transforms do not have the memory advantages of the sparse matrices, and thus they are more useful for the setting where the dataset is dense, and the encoding is done offline.

### 4.4. Cost of encoding

Since encoding increases the problem dimensions, it clearly comes with the cost of increased space complexity. The memory and storage requirement of the optimization still increases by a factor of 2, if the encoding is done offline (for dense datasets), or if the techniques described in the previous subsection are applied (for sparse datasets)[7]. Note that the added redundancy can come by increasing the amount of effective data points per machine, by increasing the number of machines while keeping the load per machine constant, or a combination of the two. In the first case, the computational load per machine increases by a factor of $\beta$. Although this can make a difference if the system is bottlenecked by the computation time, distributed computing systems are typically communication-limited, and thus we do not expect this additional cost to dominate the speed-up from the mitigation of stragglers.

## 5. Numerical Results

We implement the proposed technique on four problems: ridge regression, matrix factorization, logistic regression, and LASSO.

7. Note that the increase in space complexity is not higher for sparse matrices, since the sparsity loss can be avoided using the technique described in Section 4.3.1

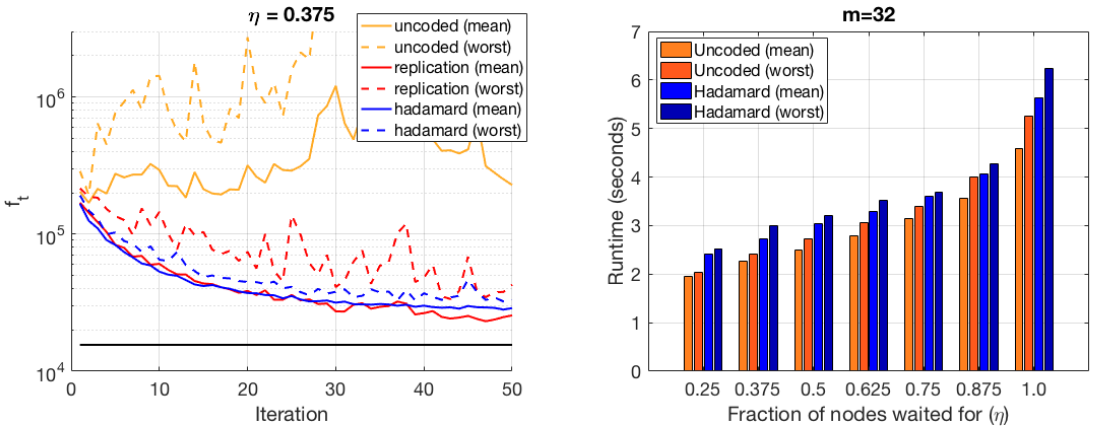


Figure 8: **Left:** Sample evolution of uncoded, replication, and Hadamard (FWHT)-coded cases, for $k = 12$, $m = 32$. **Right:** Runtimes of the schemes for different values of $\eta$, for the same number of iterations for each scheme. Note that this essentially captures the delay profile of the network, and does not reflect the relative convergence rates of different methods.

### 5.1. Data parallelism

#### 5.1.1. Ridge regression

We generate the elements of matrix $X$ i.i.d. $\sim N(0,1)$, and the elements of $y$ are generated from $X$ and an i.i.d. $N(0,1)$ parameter vector $w^*$, through a linear model with Gaussian noise, for dimensions $(n,p) = (4096, 6000)$. We solve the problem $\min_w \frac{1}{2n}\|S\left(Xw - y\right)\|^2 + \frac{\lambda}{2}\|w\|^2$, for regularization parameter $\lambda = 0.05$. We evaluate column-subsampled Hadamard matrix with redundancy $\beta = 2$ (encoded using FWHT), replication and uncoded schemes. We implement distributed L-BFGS as described in Section 3 on an Amazon EC2 cluster using `mpi4py` Python package, over $m = 32$ `m1.small` instances as worker nodes, and a single `c3.8xlarge` instance as the central server.

Figure 8 shows the result of our experiments, which are aggregated from 20 trials. In addition to uncoded scheme, we consider data replication, where each uncoded partition is replicated $\beta = 2$ times across nodes, and the server discards the duplicate copies of a partition, if received in an iteration. It can be seen that for low $\eta$, uncoded L-BFGS may not converge when a fixed number of nodes are waited for, whereas the Hadamard-coded case stably converges. We also observe that the data replication scheme converges on average, but its performance may deteriorate if both copies of a partition are delayed. Figure 8 suggests that this performance can be achieved with an approximately 40% reduction in the runtime, compared to waiting for all the nodes.

Note that in the left plot of Figure 8, iteration count does not equal time. The combined message of the two plots is that although encoding incurs a computational overhead due to increased dimensions (as can be seen by the fact that for the same $\eta$, Hadamard-encoded case takes longer than the uncoded case), the time savings due to waiting for fewer nodes per iteration (reduced $\eta$) dominates this overhead, especially for communication-limited scenarios. However, when $\eta$ is reduced, uncoded optimization can destabilize or lead to

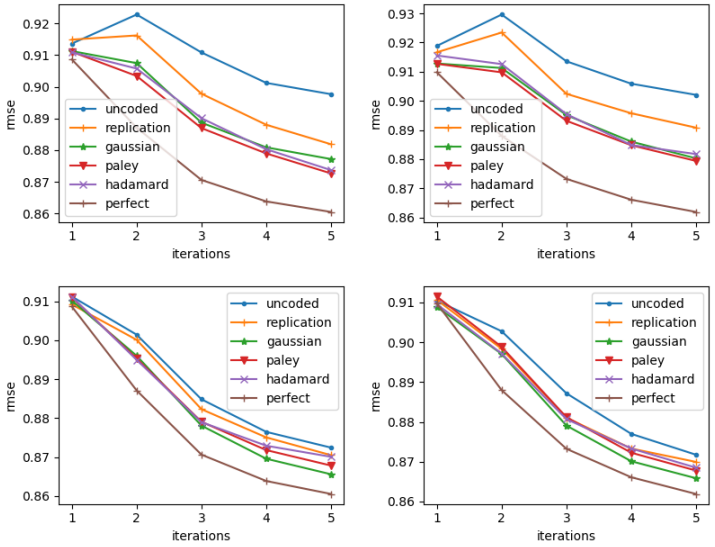


Figure 9: Test RMSE for $m = 8$ (left) and $m = 24$ (right) nodes, where the server waits for $k = m/8$ (top) and $k = m/2$ (bottom) responses. "Perfect" refers to the case where $k = m$.

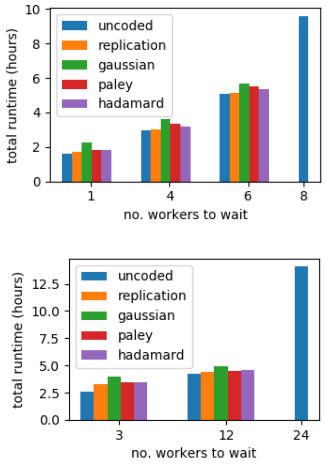


Figure 10: Total runtime with $m = 8$ and $m = 24$ nodes for different values of $k$, under fixed 100 iterations for each scheme.

less accurate solutions, as can be seen from the aggregated sample evolution on the left. The left plot shows that, encoding effectively counteracts this effect, maintaining smooth convergence due to the redundant data in the nodes.

### 5.1.2. Matrix factorization

We next apply matrix factorization on the MovieLens-1M dataset (Riedl and Konstan (1998)) for the movie recommendation task. We are given $R$, a sparse matrix of movie ratings 1–5, of dimension $\#users \times \#movies$, where $R_{ij}$ is specified if user $i$ has rated movie $j$. We withhold randomly 20% of these ratings to form an 80/20 train/test split. The goal is to recover user vectors $x_i \in \mathbb{R}^p$ and movie vectors $y_i \in \mathbb{R}^p$ (where $p$ is the embedding dimension) such that $R_{ij} \approx x_i^T y_j + u_i + v_j + b$, where $u_i$, $v_j$, and $b$ are user, movie, and global biases, respectively. The optimization problem is given by

$$\min_{x_i,y_j,u_i,v_j} \sum_{i,j:\ \text{observed}} (R_{ij} - u_i - v_j - x_i^T y_j - b)^2 + \lambda \left( \sum_i \|x_i\|_2^2 + \|u\|_2^2 + \sum_j \|y_j\|_2^2 + \|v\|_2^2 \right). \tag{12}$$

We choose $b = 3$, $p = 15$, and $\lambda = 10$, which achieves test RMSE 0.861, close to the current best test RMSE on this dataset using matrix factorization[8].

Problem (12) is often solved using alternating minimization, minimizing first over all $(x_i, u_i)$, and then all $(y_j, v_j)$, in repetition. Each such step further decomposes by row and column, made smaller by the sparsity of $R$. To solve for $(x_i, u_i)$, we first extract

8. http://www.mymedialite.net/examples/datasets.html

$I_i = \{j \mid r_{ij} \text{ is observed}\}$, and minimize

$$\left( \left[ y_{I_i}^T, \mathbf{1} \right] \begin{bmatrix} x_i \\ u_i \end{bmatrix} - (R_{i,I_i}^T - v_{I_i} - b\mathbf{1}) \right)^2 + \lambda \left( \sum_i \|x_i\|_2^2 + \|u\|_2^2 \right) \tag{13}$$

for each $i$, which gives a sequence of regularized least squares problems with variable $w = [x_i^T, u_i]^T$, which we solve distributedly using coded L-BFGS; and repeat for $w = [y_j^T, v_j]^T$, for all $j$.

The Movielens experiment is run on a single 32-core machine with Linux 4.4. In order to simulate network latency, an artificial delay of $\Delta \sim \exp(10 \text{ ms})$ is imposed each time the worker completes a task. Small problem instances ($n < 500$) are solved locally at the central server, using the built-in function `numpy.linalg.solve`. To reduce overhead, we create a bank of encoding matrices $\{S_n\}$ for Paley ETF and Hadamard ETF, for $n = 100, 200, \ldots, 3500$, and then given a problem instance, subsample the columns of the appropriate matrix $S_n$ to match the dimensions. Overall, we observe that encoding overhead is amortized by the speed-up of the distributed optimization.

Figure 9 gives the final performance of our distributed L-BFGS for various encoding schemes, for each of the 5 epochs, which shows that coded schemes are most robust for small $k$. A full table of results is given in Appendix D.

#### 5.1.3. LASSO

We solve the LASSO problem, with the objective

$$\min_w \frac{1}{2n} \|Xw - y\|^2 + \lambda \|w\|_1,$$

where $X \in \mathbb{R}^{130{,}000 \times 100{,}000}$ is a matrix with i.i.d. $N(0,1)$ entries, and $y$ is generated from $X$ and a parameter vector $w^*$ through a linear model with Gaussian noise:

$$y = Xw^* + \sigma z,$$

where $\sigma = 40$, $z \sim N(0,1)$. The parameter vector $w^*$ has 7695 non-zero entries out of 100,000, where the non-zero entries are generated i.i.d. from $N(0,4)$. We choose $\lambda = 0.6$ and consider the sparsity recovery performance of the corresponding LASSO problem, solved using proximal gradient (iterative shrinkage/thresholding algorithm).

We implement the algorithm over 128 `t2.medium` worker nodes which collectively store the matrix $X$, and a `c3.4xlarge` master node. We measure the sparsity recovery performance of the solution using the F1 score, defined as the harmonic mean

$$F1 = \frac{2PR}{P+R},$$

where $P$ and $R$ are precision recall of the solution vector $\hat{w}$ respectively, defined as

$$P = \frac{|\{i : w_i^* \neq 0, \hat{w}_i \neq 0\}|}{|i : \hat{w}_i \neq 0|}, \quad R = \frac{|\{i : w_i^* \neq 0, \hat{w}_i \neq 0\}|}{|i : w_i^* \neq 0|}$$

.

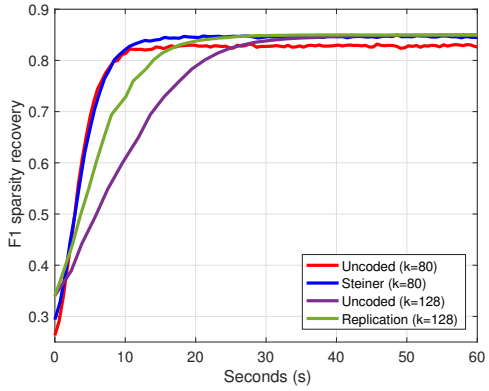


Figure 11: Evolution of F1 sparsity recovery performance for each scheme.

Figure 11 shows the sample evolution of the F1 score of the model under uncoded, replication, and Steiner encoded scenarios, with artificial multi-modal communication delay distribution $q_1\mathcal{N}(\mu_1, \sigma_1^2) + q_2\mathcal{N}(\mu_s, \sigma_2^2) + q_3\mathcal{N}(\mu_3, \sigma_3^2)$, where $q_1 = 0.8$, $q_2 = 0.1$, $q_3 = 0.1$; $\mu_1 = 0.2$s, $\mu_s = 0.6$s, $\mu_3 = 1$s; and $\sigma_1 = 0.1$s, $\sigma = 0.2$s, $\sigma_3 = 0.4$s, independently at each node. We observe that the uncoded case $k = 80$ results in a performance loss in sparsity recovery due to data dropped from delayed noes, and uncoded and replication with $k = 128$ converges slow due to stragglers, while Steiner coding with $k = 80$ is not delayed by stragglers, while maintaining almost the same sparsity recovery performance as the solution of the uncoded $k = 128$ case.

### 5.2. Model parallelism

#### 5.2.1. Logistic regression

In our next experiment, we apply logistic regression for document classification for Reuters Corpus Volume 1 (`rcv1.binary`) dataset from Lewis et al. (2004), where we consider the binary task of classifying the documents into corporate/industrial/economics vs. government/social/markets topics. The dataset has 697,641 documents, and 47,250 term frequency-inverse document frequency (tf-idf) features. We randomly select 32,500 features for the experiment, and reserve 100,000 documents for the test set. We use logistic regression with $\ell_2$-regularization for the classification task, with the objective

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left\{-z_i^\top w + b\right\}\right) + \lambda \|w\|^2,$$

where $z_i = y_i x_i$ is the data sample $x_i$ multiplied by the label $y_i \in \{-1, 1\}$, and $b$ is the bias variable. We solve this optimization using encoded distributed block coordinate descent as described in Section 2, and implement Steiner and Haar encoding as described in Section 4, with redundancy $\beta = 2$. In addition we implement the asynchronous coordinate descent, as well as replication, which represents the case where each partition $Z_i$ is replicated across two nodes, and the faster copy is used in each iteration. We use $m = 128$ `t2.medium` instances as worker nodes, and a single `c3.4xlarge` instance as the master node, which

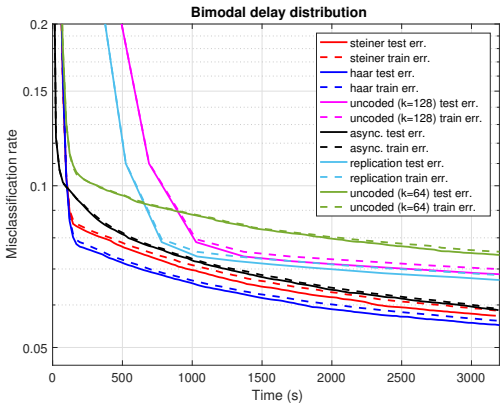


Figure 12: Test and train errors over time (in seconds) for each scheme, for the bimodal delay distribution. Steiner and Haar encoding is done with $k = 64$, $\beta = 2$.

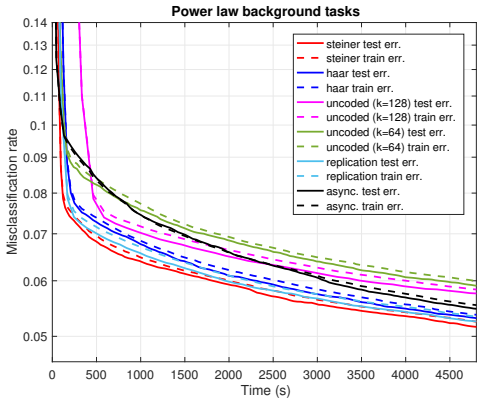


Figure 13: Test and train errors over time (in seconds) for each scheme. Number of background tasks follow a power law. Steiner and Haar encoding is done with $k = 80$, $\beta = 2$.

| **# Nodes** ($m$) | **Dimensions** ($n \times d$) |
|---|---|
| 8 | $8000 \times 3000$ |
| 32 | $32000 \times 12000$ |
| 128 | $128000 \times 48000$ |

Table 2: Problem dimensions for each experiment.

communicate using the `mpi4py` package. We consider two models for stragglers. In the first model, at each node, we add a random delay drawn from a Gaussian mixture distribution $q\mathcal{N}(\mu_1, \sigma_1^2) + (1-q)\mathcal{N}(\mu_s, \sigma_2^2)$, where $q = 0.5$, $\mu_1 = 0.5$s, $\mu_s = 20$s, $\sigma_1 = 0.2$s, $\sigma_2 = 5$s. In the second model, we do not directly add any delay, but at each machine we launch a number of dummy background tasks (matrix multiplication) that are executed throughout the computation. The number of background tasks across the nodes is distributed according to a power law with exponent $\alpha = 1.5$. The number of background tasks launched is capped at 50.

Figures 12 and 13 shows the evolution of training and test errors as a function of wall clock time. We observe that for each straggler model, either Steiner or Haar encoded optimization dominates all schemes. Figures 14 and 15 show the statistics of how frequent each node participates in an update, for the case with background tasks, for encoded and asynchronous cases, respectively. We observe that the stark difference in the relative speeds of different machines result in vastly different update frequencies for the asynchronous case, which results in updates with large delays, and a corresponding performance loss.

### 5.3. Speed tests

In order to quantify the speed-up achieved by the encoding scheme, we design a set of controlled distributed optimization experiments which compares it against other commonly

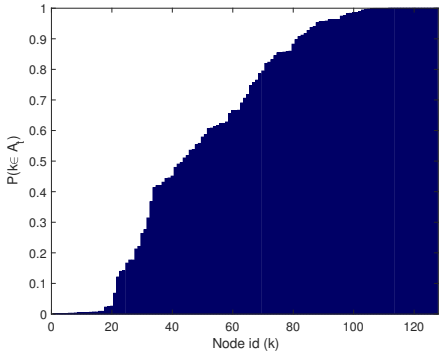


Figure 14: The fraction of iterations each worker node participates in (the empirical probability of the event $\{k \in A_t\}$), plotted for Steiner encoding with $k = 80$, $m = 128$. The number of background tasks are distributed by a power law with $\alpha = 1.5$ (capped at 50).

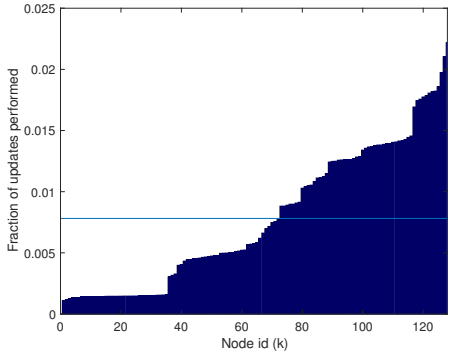


Figure 15: The fraction of updates performed by each node, for asynchronous block coordinate descent. The horizontal line represents the uniformly distributed case. The number of background tasks are distributed by a power law with $\alpha = 1.5$ (capped at 50).

| **Scheme** | **Runtime (s)** $\mu_s = 0$ | **Runtime (s)** $\mu_s = \gamma$ | **Runtime (s)** $\mu_s = 2\gamma$ |
|---|---|---|---|
| Steiner | 1.71 | 2.10 | 2.13 |
| Synchronous | 1.00 | 7.93 | 14.89 |
| Replication | 1.03 | 2.11 | 3.16 |
| Asynchronous | 0.61 | - | - |

Table 3: The average wall-clock time taken to achieve $1.05 \times$ MMSE for each scheme, for $m = 8$ workers.

| **Scheme** | **Runtime (s)** $\mu_s = 0$ | **Runtime (s)** $\mu_s = \gamma$ | **Runtime (s)** $\mu_s = 2\gamma$ |
|---|---|---|---|
| Steiner | 1.84 | 2.58 | 3.09 |
| Synchronous | 1.12 | 9.06 | 17.24 |
| Replication | 1.17 | 4.96 | 8.69 |
| Asynchronous | 0.66 | - | - |

Table 4: The average wall-clock time taken to achieve $1.05 \times$ MMSE for each scheme, for $m = 32$ workers.

| **Scheme** | **Runtime (s)** $\mu_s = 0$ | **Runtime (s)** $\mu_s = \gamma$ | **Runtime (s)** $\mu_s = 2\gamma$ |
|---|---|---|---|
| Steiner | 6.64 | 15.18 | 18.51 |
| Synchronous | 3.69 | 43.60 | 84.55 |
| Replication | 4.56 | 26.55 | 49.75 |
| Asynchronous | 8.77 | - | - |

Table 5: The average wall-clock time taken to achieve $1.05 \times$ MMSE for each scheme, for $m = 128$ workers.

straggler mitigation schemes, namely replication and asynchronous optimization, as well as the uncoded synchronous case. To do this, we focus on ridge regression, and use the mean-squared error (MSE) in predicting the test-set response variable $y$ as our metric. We compare the methods against each other by measuring the wall-clock time of optimization required to achieve a fixed mean-squared error.

Note that since the speed-up is due to the mitigation of stragglers, the performance of the proposed technique depends strongly on many cluster- and problem-specific factors, such as computational power of nodes, number of nodes, background processes running at nodes, networking configurations, latency and bandwidth available in the network, compute-to-communication ratio required by the problem, availability of nodes[9]. Since these experiments are intended purely as speed measurements, we make several design decisions to build a highly-controlled computing environment, in order to isolate the effects of the straggler mitigation technique and side-step problem- or network-specific artifacts that might affect the results.

First, to have direct control over the delays induced by the workers and measure the exact effect of the delay distribution on the runtime, we use compute-optimized, high-performance, high-bandwidth `c4.4xlarge` and `c4.large` instances available through Amazon EC2, set a node to be a straggler with probability $q = 0.25$, and for stragglers in each step we add an artificial delay drawn from Gaussian distribution $\mathcal{N}\left(\mu_s, \sigma_s^2\right)$, to simulate more unreliable clusters. In our tests, we vary $\mu_s$ to measure the effect of the amount of straggler delay (keeping $\sigma_s = 0.4\mu_s$). We sweep $\mu_s$ over the values $\{0, \gamma, 2\gamma\}$, where $\gamma$ is chosen as the amount of time it takes for a non-straggler node to perform 5-6 iterations for the given problem dimensions. Note that such delays can be observed in practice in unreliable/burstable clusters (see, for instance, the EC2 experiments in Tandon et al. (2016) conducted on `t2.micro` instances, where the stragglers are $8\times$ slower, or Ananthanarayanan et al. (2013) which similarly reports $8\times$ straggler delay for latency-limited tasks).

Second, as we scale the number of workers, we scale the problem proportionally as well, in order to maintain the same computation-communication ratio. To be able to control exact problem dimensions, we use synthetic data, where the data matrix $X \in \mathbb{R}^{n\times d}$ is generated i.i.d. $\mathcal{N}(0,1)$, and $y = X\bar{w} + \eta$ for some ground truth $\bar{w}$ and noise $\eta \in \mathcal{N}\left(0, \sigma_\eta^2\right)$, with $\sigma_\eta^2$ proportional to $n$.

Third, since under data-parallelism the proposed method does not converge to the exact minimum, we first find the minimum MSE ($\mathsf{MMSE}$) achieved for the given problem through a preliminary experiment using fully synchronous optimization with all nodes, as well as hyper-parameter optimization to determine a good value for the regularization parameter $\lambda$, which we determine to be $\lambda = 0.025$. Then, we set the MSE bar as $1.05 \times \mathsf{MMSE}$. Note that if the exact minimum is desired for encoded scenarios in a real application, one can always increase $k_t$ to $m$ (*i.e.*, start using all nodes) towards the end of the optimization, after this approximate target is reached.

Under this setup, we perform three experiments, sweeping the number of workers in the set $m \in \{8, 32, 128\}$. The problem dimensions for each case is given in Table 2. For encoding we use Steiner ETFs described in Section 4 with redundancy $\beta = 2$, and $k_t = 0.75m$. In

9. For instance, in Amazon EC2 clusters, burstable `t2` and `t3`-type instances can provide high performance for a limited period of time, before slowing down to a baseline level of performance; or low-cost spot instances, which can instantly become available or unavailable based on supply and demand.

all cases, we use gradient descent with step size $\alpha_t = 0.2$, which is run for 120 steps. $\gamma$ is determined to be 0.2s, 0.3s, and 1.2s for the three problem sizes, respectively.

The results are provided in Tables 3–5, where four methods are compared: Steiner encoding, uncoded synchronous optimization, uncoded asynchronous optimization, and replication, where each data shard is replicated across two nodes, and the faster copy is used in optimization. Note that Steiner encoding and replication schemes have double the data per worker. The results are averaged over 5 trials.

We observe that when there is no delay ($\mu_s = 0$), the additional computation overhead of encoding (as well as the perturbation of data) results in relatively high runtime to achieve the desired MSE. We see a similar overhead for replication, although replication enjoys the advantage of having access to raw, uncoded data. However, in the regime where the system is bottlenecked by stragglers ($\mu_s = \gamma$ and $\mu_s = 2\gamma$), the scheme results in significant speed up, ranging from approximately $3\times$ to $7\times$ against the uncoded synchronous; and up to $2\times$ against the replication baseline. Note that replication still gets impacted by straggler nodes if both copies of a shard happen to be delayed. When there are no stragglers, and for modest cluster sizes, asynchronous optimization achieves the best performance. However, since the gradient staleness scales linearly with the cluster size, asynchronous performance falls behind Steiner for $m = 128$ workers even in the absence of stragglers. We also observe that when stragglers are introduced, outdated gradients significantly degrade the asynchronous performance, as the desired MSE bar is not reached at the end of 120 steps. We have run the same experiment for the uncoded case with erasures as well; however, the method failed to achieve the MSE target in all the experiments, including the case with $\mu_s = 0$. This is because the straggler nodes tend to persist their slow computation throughout optimization. In other words, it is usually approximately the same subset of nodes that provide the updates, effectively removing the data stored in straggler nodes from the optimization. Finally, we note that the runtime for the encoded case also increases in the presence of stragglers (though less severely than the other schemes), due to various implementation overheads such as the interruption of the stragglers for the next iteration. However, we believe that part of this overhead could be mitigated in a more efficient implementation.

We conclude from these experiments that encoding can result in large speed-ups in highly unreliable, low-cost clusters which can induce large delays (such as EC2 `t2` instances), or clusters with intermittent node availability (such as EC2 spot instances). It is less suited to reliable clusters with high-capacity, high-bandwidth nodes due to the computational overhead associated with it.

## Acknowledgments

The work of Can Karakus and Suhas Diggavi was supported in part by NSF grants #1314937 and #1514531, and by UC-NL grant LFR-18-548554. The work of Wotao Yin was supported by ONR Grant N000141712162, and NSF Grant DMS-1720237.

## Appendix A. Proofs of Theorems 2 and 4

In the proofs, we will ignore the normalization constants on the objective functions for brevity. We will assume the normalization $\frac{1}{\sqrt{\eta}}$ is absorbed into the encoding matrix $S_A$. Let $f(w) = \|Xw - y\|^2 + \lambda h(w)$. Let $\widetilde{f}_t^A := \|S_{A_t}(Xw_t - y)\|^2 + \lambda h(w_t)$, and $\widetilde{f}^A(w) := \|S_{A_t}(Xw - y)\|^2 + \lambda h(w)$, where we set $A \equiv A_t$. Let $\widetilde{w}_t^*$ denote the solution to the effective "instantaneous" problem at iteration $t$, *i.e.*, $\widetilde{w}_t^* = \arg\min_w \widetilde{f}^A(w)$.

Unless otherwise stated, throughout this appendix, we will also denote

$$w^* = \arg\min_w \|Xw - y\|^2 + \lambda h(w)$$
$$\hat{w} = \arg\min_w \|S_A(Xw - y)\|^2 + \lambda h(w)$$

unless otherwise noted, where $A$ is a fixed subset of $[m]$.

### A.1. Lemmas

**Lemma 10 (Approximation error)** *If $S$ satisfies (6) (BRIP) for any $A \subseteq [m]$ with $|A| \geq k$, for any convex set $C$,*

$$\|X\hat{w} - y\|^2 \leq \kappa^2 \|Xw^* - y\|^2,$$

*where $\kappa = \frac{1+\epsilon}{1-\epsilon}$, $\hat{w} = \arg\min_{w \in C} \|S_A(Xw - y)\|^2$, and $w^* = \arg\min_{w \in C} \|Xw - y\|^2$.*

**Proof** Define $e = \hat{w} - w^*$ and note that

$$\|X\hat{w} - y\| = \|Xw^* - y + Xe\| \leq \|Xw^* - y\| + \|Xe\|$$

by triangle inequality, which implies

$$\|X\hat{w} - y\|^2 \leq \left(1 + \frac{\|Xe\|}{\|Xw^* - y\|}\right)^2 \|Xw^* - y\|^2. \tag{14}$$

Note that we have

$$\|S_A(X\hat{w} - y)\|^2 \leq \|S_A(Xw^* - y)\|^2$$

by the definition of $\hat{w}$. Expanding the quadratic terms and canceling the common term $y^\top S_A^\top S_A y$ from both sides, we get

$$\hat{w}^\top X^\top S_A^\top S_A X\hat{w} - 2y^\top S_A^\top S_A X\hat{w} \leq w^{*\top} X^\top S_A^\top S_A X w^* - 2y^\top S_A^\top S_A X w^*.$$

Adding $-2w^{*\top} X^\top S_A^\top S_A X\hat{w} + w^{*\top} X^\top S_A^\top S_A X w^*$ to both sides,

$$\begin{aligned}\hat{w}^\top X^\top S_A^\top S_A X\hat{w} - 2w^{*\top} X^\top S_A^\top S_A X\hat{w} + w^{*\top} X^\top S_A^\top S_A X w^* - 2y^\top S_A^\top S_A X\hat{w} \\ \leq 2w^{*\top} X^\top S_A^\top S_A X w^* - 2y^\top S_A^\top S_A X w^* - 2w^{*\top} X^\top S_A^\top S_A X\hat{w},\end{aligned}$$

which can be re-arranged into

$$\|S_A X(\hat{w} - w^*)\|^2 \leq 2y^\top S_A^\top S_A X(\hat{w} - w^*) - 2w^{*\top} X^\top S_A^\top S_A X(\hat{w} - w^*),$$

which, using the definition $e = \hat{w} - w^*$, is equivalent to

$$\|S_A X e\|^2 \leq -2(Xw^* - y)^\top S_A^\top S_A X e = -2e^\top X^\top S_A^\top S_A (Xw^* - y). \tag{15}$$

Now, for any $c > 0$, consider

$$\begin{aligned}
\|Xe\|^2 &\overset{(a)}{\leq} \frac{\|S_A X e\|^2}{1-\epsilon} \overset{(b)}{\leq} -2\frac{e^\top X^\top S_A^\top S_A (Xw^* - y)}{1-\epsilon} \\
&= -2\frac{e^\top X^\top \left(S_A^\top S_A - cI\right)(Xw^* - y)}{1-\epsilon} - \frac{2c}{1-\epsilon} e^\top X^\top (Xw^* - y) \\
&\overset{(c)}{\leq} -2\frac{e^\top X^\top \left(S_A^\top S_A - cI\right)(Xw^* - y)}{1-\epsilon} \\
&\overset{(d)}{\leq} 2\frac{\left\|e^\top X^\top \left(cI - S_A^\top S_A\right)\right\|}{1-\epsilon} \|Xw^* - y\| \\
&\overset{(e)}{\leq} 2\frac{\left\|cI - S_A^\top S_A\right\|}{1-\epsilon} \|Xw^* - y\| \|Xe\|,
\end{aligned}$$

where (a) follows from the BRIP property for $S_A$; (b) follows by (15); (c) follows by the fact that since $\hat{w} \in C$, $e$ represents a feasible direction of the constrained optimization, and thus the convex optimality condition implies $\langle \nabla p(w^*), \hat{w} - w^* \rangle = 2e^\top X^\top (Xw^* - y) \geq 0$, where $p(w) := \|Xw - y\|^2$; (d) follows by Cauchy-Schwarz inequality; and (e) follows by the definition of matrix norm.

The above chain of inequalities then imply that

$$\frac{\|Xe\|}{\|Xw^* - y\|} \leq \frac{2\left\|cI - S_A^\top S_A\right\|}{1-\epsilon}.$$

Since this is true for any $c > 0$, we make the minimizing choice $c = \frac{\lambda_{\max} + \lambda_{\min}}{2}$ (where $\lambda_{\max}$ and $\lambda_{\min}$ represent the largest and smallest eigenvalues of $S_A^\top S_A$, respectively), which gives

$$\frac{\|Xe\|}{\|Xw^* - y\|} \leq \frac{\lambda_{\max} - \lambda_{\min}}{1-\epsilon} \leq \frac{2\epsilon}{1-\epsilon}.$$

Plugging this back in (14), we get the desired result. ■

**Lemma 11 (Approximation error with regularization)** *If $S$ satisfies* (6) *(BRIP) for any $A \subseteq [m]$ with $|A| \geq k$,*

$$f(\hat{w}) \leq \kappa^2 f(w^*),$$

*where $\kappa = \frac{1+\epsilon}{1-\epsilon}$, $\hat{w} = \arg\min_w \|S_A(Xw - y)\|^2 + \lambda h(w)$, and $w^* = \arg\min_w \|Xw - y\|^2 + \lambda h(w)$.*

**Proof** Consider a fixed $A_t = A$, and a corresponding

$$\hat{w} = \widetilde{w}_t^* \in \arg\min_w \|S_A(Xw - y)\|^2 + \lambda h(w)$$

Define

$$\hat{w}(r) = \underset{w:\lambda h(w)\leq r}{\arg\min} \|S_A\left(Xw-y\right)\|^2$$
$$w^*(r) = \underset{w:\lambda h(w)\leq r}{\arg\min} \|Xw-y\|^2.$$

Finally, define

$$r^* = \underset{r}{\arg\min} \|Xw^*(r)-y\|^2 + r.$$

Now, consider

$$\begin{aligned} f(\hat{w}) = \|X\hat{w}-y\|^2 + \lambda h(\hat{w}) &= \min_r \left(\|X\hat{w}(r)-y\|^2 + r\right) \\ &\leq \|X\hat{w}(r^*)-y\|^2 + r^* \overset{(a)}{\leq} \kappa^2\|Xw^*(r^*)-y\|^2 + r^* \\ &\leq \kappa^2\left(\|Xw^*(r^*)-y\|^2 + r^*\right) = \kappa^2 f(w^*), \end{aligned}$$

which shows the desired result, where (a) follows by Lemma 10, and by the fact that the set $\{w : \lambda h(w) \leq r\}$ is a convex set. ■

**Lemma 12 (Error evolution)** *Let* $\kappa = \frac{1+\epsilon}{1-\epsilon}$. *If*

$$\widetilde{f}^A_{t+1} - \widetilde{f}^A\left(\widetilde{w}^*_t\right) \leq \gamma\left(\widetilde{f}^A_t - \widetilde{f}^A\left(\widetilde{w}^*_t\right)\right)$$

*for all* $t > 0$, *and for some* $0 < \gamma < 1$ *with* $\kappa\gamma < 1$, *where* $\widetilde{w}^*_t \in \arg\min_w \widetilde{f}^A_t$, *then*

$$f(w_t) \leq (\kappa\gamma)^t f(w_0) + \frac{\kappa^2(\kappa-\gamma)}{1-\kappa\gamma} f\left(w^*\right).$$

**Proof** Since for any $w$,

$$(1-\epsilon)\|Xw-y\|^2 \leq (Xw-y)^\top S_A^\top S_A\left(Xw-y\right),$$

we have

$$(1-\epsilon) f(w) \leq \widetilde{f}^A(w).$$

Similarly $\widetilde{f}^A(w) \leq (1+\epsilon) f(w)$, and therefore, using the assumption of the lemma

$$(1-\epsilon) f(w_{t+1}) - (1+\epsilon) f\left(\widetilde{w}^*_t\right) \leq \gamma\left((1+\epsilon) f(w_t) - (1-\epsilon) f\left(\widetilde{w}^*_t\right)\right),$$

which can be re-arranged into the linear recursive inequality

$$f(w_{t+1}) \leq \kappa\gamma f(w_t) + (\kappa-\gamma) f\left(\widetilde{w}^*_t\right) \overset{(a)}{\leq} \kappa\gamma f(w_t) + \kappa^2(\kappa-\gamma) f\left(w^*\right),$$

where $\kappa = \frac{1+\epsilon}{1-\epsilon}$ and (a) follows by Lemma 11. By considering such inequalities for $0 \leq \tau \leq t$, multiplying each by $(\kappa\gamma)^{t-\tau}$ and summing, we get

$$\begin{aligned} f(w_t) &\leq (\kappa\gamma)^t f(w_0) + \kappa^2(\kappa-\gamma) f(w^*) \sum_{\tau=0}^{t-1} (\kappa\gamma)^\tau \\ &\leq (\kappa\gamma)^t f(w_0) + \frac{\kappa^2(\kappa-\gamma)}{1-\kappa\gamma} f(w^*). \end{aligned}$$

■

**Lemma 13 (Strong convexity parameter)** *Under the assumptions of Theorem 4, $\widetilde{f}^A(w)$ is $(1-\epsilon)(\mu+\lambda)$-strongly convex.*

**Proof** It is sufficient to show that the minimum eigenvalue of $\widetilde{X}_A^\top \widetilde{X}_A$ is bounded away from zero. This can easily be shown by the fact that

$$u^\top \widetilde{X}_A^\top \widetilde{X}_A u = u^\top \hat{X}^\top S_A^\top S_A \hat{X} u \geq (1-\epsilon)\|\hat{X}u\|^2 \geq (1-\epsilon)(\mu+\lambda)\|u\|^2,$$

for any unit vector $u$. ■

**Lemma 14 (Rotation bound)** *Let $M \in \mathbb{R}^{p\times p}$ be a symmetric positive definite matrix, with the condition number (ratio of maximum eigenvalue to the minimum eigenvalue) given by $\kappa$. Then, for any unit vector $u$,*

$$\frac{u^\top M u}{\|Mu\|} \geq \frac{2\sqrt{\kappa}}{\kappa+1}.$$

**Proof** We point out that this is a special case of Kantorovich inequality, but provide a dedicated proof here for completeness.

Let $M$ have the eigen-decomposition $M = Q^\top D Q$, where $Q$ has orthonormal columns, and $D$ is a diagonal matrix with positive, decreasing entries $d_1 \geq d_2 \geq \cdots \geq d_n$, with $\frac{d_1}{d_n} = \kappa$. Let $y = (Qu)^{\circ 2}$, where $^{\circ 2}$ denotes entry-wise square. Then the quantity we are interested in can be represented as

$$\hat{y} = \arg\inf_{y\in\Delta} \frac{\sum_{i=1}^n d_i y_i}{\sqrt{\sum_{i=1}^n d_i^2 y_i}},$$

and $\frac{u^\top M u}{\|Mu\|}$ is lower bounded by the optimal objective value of the above optimization problem. The set $\Delta = \{y \geq 0 : \mathbf{1}^\top y = 1\}$ is the simplex. Note that $\hat{y}$ is invariant to any positive scaling on the entire vector $d$, so we can consider a convexification, where the minimum is attained since $y$ is confined to a compact set, and where $\hat{y}$ is the solution to:

$$\begin{array}{ll} \textbf{min.} & \hat{d}^T y \\ \textbf{subj.to} & \sum_{i=1}^n \hat{d}_i^2 y_i = 1 \\ & y \geq 0,\ y^T \mathbf{1} = 1 \end{array}$$

and $\hat{d}$ is the scaled version of $d$ such that $\sum_{i=1}^n \hat{d}_i^2 y_i = 1$. The Lagrange dual is

$$\begin{array}{ll} \textbf{max.} & \alpha + \beta \\ \textbf{subj.to} & \hat{d} - \beta \hat{d}^{\circ 2} - \alpha \mathbf{1} \geq \mathbf{0} \end{array}$$

where by complementary slackness, the gap in the inequality corresponds to the nonzero pattern of $y$, e.g.

$$y_i(\hat{d}_i - \beta \hat{d}_i^2 - \alpha) = 0, \ \forall i \in [n].$$

Now we consider the quadratic function $d - \beta d^2 - \alpha$ in terms of a scalar $d$, with fixed $\alpha$ and $\beta$. It is clear that this concave function can have at most 2 zeros. In other words, there can be at most two indices $i, j$ where

$$\hat{d}_k - \beta \hat{d}_k^2 - \alpha = 0, \ k \in \{i, j\}, \quad y_k = 0 \ k \notin \{i, j\}$$

and it is possible that $i = j$. (At least one solution must exist, since $y^T \mathbf{1} = 1$, so $y$ cannot be all 0.) This reduces our primal optimization problem significantly, to two variables $y_i$ and $y_j$, where the two linear constraints can be inverted directly

$$y_i = \frac{1 - \hat{d}_j^2}{\hat{d}_i^2 - \hat{d}_j^2}, \ y_j = \frac{\hat{d}_i^2 - 1}{\hat{d}_i^2 - \hat{d}_j^2}$$

Plugging back into the primal objective,

$$y_i \hat{d}_i + y_j \hat{d}_j = \frac{\hat{d}_i - \hat{d}_i \hat{d}_j^2 + \hat{d}_j \hat{d}_i^2 - \hat{d}_j}{\hat{d}_i^2 - \hat{d}_j^2} = \frac{1 + (\hat{d}_i \hat{d}_j)}{\hat{d}_i + \hat{d}_j}$$

which is minimized when $\hat{d}_i$ and $\hat{d}_j$ are as far apart as possible; e.g., $i = 1$ and $j = n$.

We can equivalently reparametrize $y_1$ and $y_n$ in terms of $d_1$ and $d_n$ as

$$y_1(c) = \frac{c^2 - d_n^2}{d_1^2 - d_n^2}, \ y_n(c) = \frac{d_1^2 - c^2}{d_1^2 - d_n^2}$$

where $c^2 = d_1^2 y_1 + d_n^2 y_n$. Now we consider our original optimization problem, significantly simplified

$$\inf_{y \in \Delta} \frac{\sum_{i=1}^n d_i y_i}{\sqrt{\sum_{i=1}^n d_i^2 y_i}} = \inf_{c \geq 0} \frac{d_1 y_1(c) + d_n y_n(c)}{c} = \inf_{c \geq 0} \frac{c^2 + d_1 d_n}{c(d_1 + d_n)}.$$

This is now a scalar nonconvex optimization problem, with minimum at either the boundaries ($c = 0$ or $c \to +\infty$) or the one stationary point ($c = d_1 d_n$). The stationary point is the only one in which this quantity is not positive infinity, and is thus the solution. Therefore, simplifying with $\kappa = d_1/d_n$, we arrive at

$$\hat{y}_1 = \frac{1}{\kappa + 1}, \ \hat{y}_n = \frac{\kappa}{\kappa + 1}.$$

and for any unit-normed $u$,

$$\frac{u^\top M u}{\|Mu\|} \geq \frac{c^2 + d_1 d_n}{c(d_1 + d_n)} = \frac{(1 + d_1 d_n)}{(d_1 + d_n)} = \frac{(1/d_n + d_1)}{(\kappa + 1)} \geq \frac{2\sqrt{d_1/d_n})}{(\kappa + 1)} = \frac{2\sqrt{\kappa})}{(\kappa + 1)}.$$

where the inequality comes from the arithmetic-geometric mean inequality. ■

### A.2. Proof of Theorem 2

The proof of the first part of the theorem is a special case of the proof of Theorem 5 (with $\lambda = 0$, and the smooth regularizer incorporated into $p(w)$) and thus we omit this proof and refer the reader to Appendix B. We prove the second part here.

Note that because of the condition in (6) (BRIP), we have

$$(1-\epsilon)I \preceq S_A^\top S_A \preceq (1+\epsilon)I.$$

Using smoothness of the objective, and the choices $d_t = -\nabla \widetilde{f}^A(w_t)$ and $\alpha_t = \alpha$, we have

$$\begin{aligned}
&\widetilde{f}^A\left(w_{t+1}\right) - \widetilde{f}^A(w_t) \leq \alpha \nabla \widetilde{f}^A(w_t)(w_t)^\top d_t + \frac{1}{2}\alpha^2 d_t^\top X^\top S_A^\top S_A X d_t + \lambda \frac{L}{2}\alpha^2 \|d_t\|^2 \\
&\leq -\alpha\left(1 - \frac{(1+\epsilon)M + \lambda L}{2}\alpha\right)\left\|\nabla \widetilde{f}^A(w_t)\right\|^2 = -\frac{2\zeta\left(1-\zeta\right)}{(1+\epsilon)M + \lambda L}\|\nabla \widetilde{f}^A(w_t)\|^2 \\
&\overset{\text{(a)}}{\leq} -\frac{4\nu\zeta\left(1-\zeta\right)}{M\left(1+\epsilon\right) + \lambda L}\left(\widetilde{f}^A\left(w_t\right) - \widetilde{f}^A\left(\widetilde{w}_t^*\right)\right),
\end{aligned}$$

where (a) follows by strong convexity. Re-arranging this inequality, and using the definition of $\gamma$, we get

$$\widetilde{f}_{t+1}^A - \widetilde{f}^A\left(\widetilde{w}_t^*\right) \leq \gamma\left(\widetilde{f}_t^A - \widetilde{f}^A\left(\widetilde{w}_t^*\right)\right),$$

which, using Lemma 12, implies the result.

### A.3. Proof of Theorem 4

We first present the proof of Lemma 3, then move on to the main proof.

**Proof** [Proof of Lemma 3] The reader is referred to Section 2.1 for notation. Define $\breve{S}_t := S_{A_t \cap A_{t-1}}$. First note that

$$\begin{aligned}
r_t^\top u_t &= \left(X^\top \breve{S}_t^\top \breve{S}_t\left[\left(Xw_t - y\right) - \left(Xw_{t-1} - y\right)\right]\right)^\top \left(w_t - w_{t-1}\right) \\
&= \left(w_t - w_{t-1}\right)^\top X^\top \breve{S}_t^\top \breve{S}_t X\left(w - w_{t-1}\right) \\
&\geq \delta\mu\|u_t\|^2, \qquad (16)
\end{aligned}$$

by (7), where recall $\delta \geq 0$ is the smallest eigenvalue of $\breve{S}_t^\top \breve{S}_t$. Also consider

$$\frac{\|r_t\|^2}{r_t^\top u_t} = \frac{\left(w_t - w_{t-1}\right)^\top \left(X^\top \breve{S}_t^\top \breve{S}_t X\right)\left(X^\top \breve{S}_t^\top \breve{S}_t X\right)\left(w_t - w_{t-1}\right)}{\left(w_t - w_{t-1}\right)^\top X^\top \breve{S}_t^\top \breve{S}_t X\left(w_t - w_{t-1}\right)},$$

which implies

$$\delta\mu \leq \frac{\|r_t\|^2}{r_t^\top u_t} \leq (1+\epsilon)M,$$

again by (6). Now, setting $j_\ell = t - \widetilde{\sigma} + \ell$, consider the trace

$$\mathbf{tr}\left(B_t^{(\ell+1)}\right) = \mathbf{tr}\left(B_t^{(\ell)}\right) - \mathbf{tr}\left(\frac{B_t^{(\ell)} u_{j_\ell} u_{j_\ell}^\top B_t^{(\ell)}}{u_{j_\ell}^\top B_t^{(\ell)} u_{j_\ell}}\right) + \mathbf{tr}\left(\frac{r_{j_\ell} r_{j_\ell}^\top}{r_{j_\ell}^\top u_{j_\ell}}\right)$$

$$\begin{aligned}
&\overset{\text{(a)}}{\leq} \mathbf{tr}\left(B_t^{(\ell)}\right) + \mathbf{tr}\left(\frac{r_{j_\ell} r_{j_\ell}^\top}{r_{j_\ell}^\top u_{j_\ell}}\right) \\
&= \mathbf{tr}\left(B_t^{(\ell)}\right) + \frac{\|r_{j_\ell}\|^2}{r_{j_\ell}^\top u_{j_\ell}} \\
&\leq \mathbf{tr}\left(B_t^{(\ell)}\right) + (1+\epsilon)M,
\end{aligned}$$

where (a) follows since the trace of a positive semidefinite matrix is always non-negative. This implies $\mathbf{tr}\left(B_t\right) \leq (1+\epsilon)M\left(\widetilde{\sigma}+d\right)$. We now do a similar exercise for the determinant. Note first that for two arbitrary vectors $a$ and $b$,

$$\begin{aligned}
\det(I - aa^T + bb^T) &= \det\left(I + \begin{bmatrix}-a & b\end{bmatrix}\begin{bmatrix}a^T \\ b^T\end{bmatrix}\right) \overset{\text{(a)}}{=} \det\left(I + \begin{bmatrix}a^T \\ b^T\end{bmatrix}\begin{bmatrix}-a & b\end{bmatrix}\right) \\
&= (1 - a^T a)(1 + b^T b) + (a^T b)^2
\end{aligned}$$

where (a) is by Sylvester's identity. Since $B_t^{(\ell)} \succ 0$, we can take the Cholesky factorization $B_t^{(\ell)} = LL^T$ and write the recursion for $B_t^{(\ell)}$ as a product:

$$B_t^{(\ell+1)} \;=\; L\left(I - L^T \frac{u_{j_\ell} u_{j_\ell}^T}{u_{j_\ell}^T B_t^{(\ell)} u_{j_\ell}} L + L^{-1}\frac{r_{j_\ell} r_{j_\ell}^T}{r_{j_\ell}^T u_{j_\ell}} L^{-T}\right) L^T.$$

Using the above identity for

$$a = \frac{1}{u_{j_\ell}^T B_t^{(\ell)} u_{j_\ell}} L^T u_{j_\ell}, \qquad b = \frac{1}{\sqrt{r_{j_\ell}}^T u_{j_\ell}} L^T r_{j_\ell}$$

we can see that $a^T a = 1$, and $(a^T b)^2 = \frac{r_{j_\ell}^T u_{j_\ell}}{u_{j_\ell}^T B_t^{(\ell)} u_{j_\ell}}$. Therefore

$$\begin{aligned}
\det\left(B_t^{(\ell+1)}\right) &= \det\left(B_t^{(\ell)}\right) \cdot \frac{r_{j_\ell}^\top u_{j_\ell}}{u_{j_\ell}^\top B_t^{(\ell)} u_{j_\ell}} \\
&= \det\left(B_t^{(\ell)}\right) \cdot \frac{r_{j_\ell}^\top u_{j_\ell}}{\|u_{j_\ell}\|^2} \cdot \frac{\|u_{j_\ell}\|^2}{u_{j_\ell}^\top B_t^{(\ell)} u_{j_\ell}} \\
&\geq \det\left(B_t^{(\ell)}\right) \cdot \delta\mu \cdot \frac{\|u_{j_\ell}\|^2}{u_{j_\ell}^\top B_t^{(\ell)} u_{j_\ell}} \\
&\geq \det\left(B_t^{(\ell)}\right) \cdot \delta\mu \cdot \frac{1}{\lambda_{\min}(B_t^{(\ell)})} \\
&\geq \det\left(B_t^{(\ell)}\right) \cdot \delta\mu \cdot \frac{1}{\mathbf{tr}\left(B_t^{(\ell)}\right)} \\
&\geq \det\left(B_t^{(\ell)}\right) \frac{\delta\mu}{(1+\epsilon)M\left(\widetilde{\sigma}+d\right)},
\end{aligned}$$

which implies $\det(B_t) \geq \det\left(B_t^{(0)}\right)\left(\frac{\delta\mu}{(1+\epsilon)M(\widetilde{\sigma}+d)}\right)^{\widetilde{\sigma}}$. By construction, $\mathbf{tr}\left(B_t^{(0)}\right)$ is bounded above and $\det(B_t^{(0)})$ is bounded below, which means $\mathbf{tr}(B_t)$ is bounded above and $\det(B_t)$ is bounded below. Therefore there must exist $0 < c_1 \leq c_2$ such that

$$c_1 I \preceq B_t \preceq c_2 I.$$

■

**Proof** [Proof of Theorem 4]

Since $h(w)$ is constrained to be quadratic, we can absorb this term into the error term to get

$$\min_w \left\| \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix} \left( \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix} w - \begin{bmatrix} y \\ 0 \end{bmatrix} \right) \right\|.$$

Note that as long as $S$ satisfies (6), the effective encoding matrix $\mathrm{diag}([S, I])$ also satisfies the same. Therefore, without loss of generality we can ignore $h(w)$, and consider minimizing the unregularized quadratic problem

$$\min_w \|\hat{X}w - y\|, \qquad \hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix}.$$

Here,

$$(\mu + \lambda) I \preceq \hat{X}^\top \hat{X} \preceq (M + \lambda) I.$$

We also define the extreme eigenvalues of the encoding matrices as $\lambda_{\min} = 1 - \epsilon$ and $\lambda_{\max} = 1 + \epsilon$ for convenience. Using convexity and the closed-form expression for the step size, we have

$$\begin{aligned}
&\widetilde{f}^A(w_{t+1}) - \widetilde{f}^A(w_t) \leq \alpha_t \nabla \widetilde{f}^A(w_t)^\top d_t + \frac{1}{2}\alpha_t^2 d_t^\top \hat{X}^\top S_A^\top S_A \hat{X} d_t \\
&= -\frac{\rho\left(\nabla \widetilde{f}^A(w_t)^\top d_t\right)^2}{d_t^\top \hat{X}^\top S_D^\top S_D \hat{X} d_t} + \frac{1}{2}\frac{\rho^2\left(\nabla \widetilde{f}^A(w_t)^\top d_t\right)^2}{d_t^\top \hat{X}^\top S_D^\top S_D \hat{X} d_t} \cdot \frac{d_t^\top \hat{X}^\top S_A^\top S_A \hat{X} d_t}{d_t^\top \hat{X}^\top S_D^\top S_D \hat{X} d_t} \\
&= \left( \frac{d_t^\top \hat{X}^\top \left(\rho^2 S_A^\top S_A - 2\rho S_D^\top S_D\right) \hat{X} d_t}{2\left(d_t^\top \hat{X}^\top S_D^\top S_D \hat{X} d_t\right)^2} \right) \left(d_t^\top \nabla \widetilde{f}^A(w_t)\right)^2 \\
&\overset{(a)}{=} -\rho \left( \frac{z^\top \left(S_D^\top S_D - \frac{\rho}{2} S_A^\top S_A\right) z}{\left(z^\top S_D^\top S_D z\right)^2} \right) \frac{\left(d_t^\top \nabla \widetilde{f}^A(w_t)\right)^2}{\|\hat{X} d_t\|^2} \\
&\overset{(b)}{\leq} -\rho \left( \frac{\lambda_{\min} - \frac{\rho}{2}\lambda_{\max}}{\lambda_{\min}^2} \right) \frac{\left(d_t^\top \nabla \widetilde{f}^A(w_t)\right)^2}{\|\hat{X} d_t\|^2} \overset{(c)}{\leq} -\frac{\rho}{M+\lambda} \left( \frac{\lambda_{\min} - \frac{\rho}{2}\lambda_{\max}}{\lambda_{\min}^2} \right) \frac{\left(d_t^\top \nabla \widetilde{f}^A(w_t)\right)^2}{\|d_t\|^2} \\
&\overset{(d)}{=} -\frac{\rho}{M+\lambda} \left( \frac{\lambda_{\min} - \frac{\rho}{2}\lambda_{\max}}{\lambda_{\min}^2} \right) \frac{\left(\nabla \widetilde{f}^A(w_t)^\top B_t \nabla \widetilde{f}^A(w_t)\right)^2}{\|B_t \nabla \widetilde{f}^A(w_t)\|^2}
\end{aligned}$$

$$\overset{\text{(e)}}{\leq} -\frac{4\rho}{M+\lambda}\left(\frac{\lambda_{\min}-\frac{\rho}{2}\lambda_{\max}}{\lambda_{\min}^2}\right)\frac{c_1c_2}{(c_1+c_2)^2}\|\nabla\widetilde{f}^A(w_t)\|^2$$
$$\overset{\text{(f)}}{\leq} -\frac{8(\mu+\lambda)\rho}{M+\lambda}\left(\frac{\lambda_{\min}-\frac{\rho}{2}\lambda_{\max}}{\lambda_{\min}^2}\right)\frac{c_1c_2}{(c_1+c_2)^2}\left(\widetilde{f}\left(w_t\right)-\widetilde{f}\left(\widetilde{w}_t^*\right)\right)$$
$$\overset{\text{(g)}}{=} -\frac{4(\mu+\lambda)c_1c_2}{(M+\lambda)(1+\epsilon)\left(c_1+c_2\right)^2}\left(\widetilde{f}\left(w_t\right)-\widetilde{f}\left(\widetilde{w}_t^*\right)\right)\overset{\text{(h)}}{=}-\left(1-\gamma\right)\left(\widetilde{f}^A\left(w_t\right)-\widetilde{f}^A\left(\widetilde{w}_t^*\right)\right).$$

where (a) follows by defining $z=\frac{\hat{X}d_t}{\|\hat{X}d_t\|}$; (b) follows by (6); (c) follows by the assumption that $\hat{X}^\top\hat{X}\preceq(M+\lambda)I$; (d) follows by the definition of $d_t$; (e) follows by Lemmas 14 and 3; (f) follows by strong convexity of $\widetilde{f}$ (by Lemma 13), which implies $\|\nabla\widetilde{f}^A(w_t)\|^2\geq 2(\mu+\lambda)\left(\widetilde{f}\left(\theta_t\right)-\widetilde{f}\left(\widetilde{w}_t^*\right)\right)$; (g) follows by choosing $\rho=\frac{\lambda_{\min}}{\lambda_{\max}}$; and (h) follows using the definition of $\gamma$.

Re-arranging the inequality, we obtain

$$\widetilde{f}_{t+1}^A-\widetilde{f}^A\left(\widetilde{w}_t^*\right)\leq\gamma\left(\widetilde{f}_t^A-\widetilde{f}^A\left(\widetilde{w}_t^*\right)\right),$$

and hence applying Lemma 12, we get the desired result. ■

## Appendix B. Proof of Theorem 5

Throughout this appendix, we will define $p(w)=\frac{1}{2}\|Xw-y\|^2$ and $\widetilde{p}_t(w)=\frac{1}{2}\|S_{A_t}\left(Xw-y\right)\|^2$ for convenience, where the normalization by $\sqrt{\eta}$ is absorbed into $S_A$. We will omit the normalization by $n$ for brevity. Let us also define

$$w^*=\arg\min_w p(w)+\lambda h(w)$$

to be the true solution of the optimization problem.

By $M$-smoothness of $p(w)$,

$$\begin{aligned}
p(w_{t+1}) &\leq p(w_t)+\langle\nabla p(w_t),w_{t+1}-w_t\rangle+\frac{M}{2}\|w_{t+1}-w_t\|^2\\
&\leq p(w^*)-\langle\nabla p(w_t),w^*-w_t\rangle+\langle\nabla p(w_t),w_{t+1}-w_t\rangle+\frac{M}{2}\|w_{t+1}-w_t\|^2\\
&\leq p(w^*)-\langle\nabla p(w_t),w^*-w_t\rangle+\langle\nabla p(w_t),w_{t+1}-w_t\rangle+\frac{1}{2\alpha}\|w_{t+1}-w_t\|^2\\
&\leq p(w^*)+\langle\nabla p(w_t),w_{t+1}-w^*\rangle+\frac{1}{2\alpha}\|w_{t+1}-w_t\|^2 \qquad (17)
\end{aligned}$$

where the second line follows by convexity of $p$, and the third line follows since $\alpha<\frac{1}{M}$. Since $w_{t+1}=\arg\min_w\widetilde{F}_t(w)$, by optimality conditions

$$0\in\lambda\cdot\partial h(w_{t+1})+\nabla\widetilde{p}_t(w_t)+\frac{1}{\alpha}\left(w_{t+1}-w_t\right). \qquad (18)$$

Since $h$ is convex, any subgradient $g \in \partial h$ at $w = w_{t+1}$ satisfies

$$h(w^*) \geq h(w_{t+1}) + \langle g, w^* - w_{t+1} \rangle,$$

and therefore (18) implies

$$\lambda h(w^*) \geq \lambda h(w_{t+1}) - \langle \nabla \widetilde{p}_t(w_t), w^* - w_{t+1} \rangle - \frac{1}{\alpha} \langle w_{t+1} - w_t, w^* - w_{t+1} \rangle. \tag{19}$$

Combining (17) and (19),we have

$$\begin{aligned}
f(w_{t+1}) &= p(w_{t+1}) + \lambda h(w_{t+1}) && (20)\\
&= p(w^*) + \langle \nabla p(w_t), w_{t+1} - w^* \rangle + \frac{1}{2\alpha} \|w_{t+1} - w_t\|^2 && (21)\\
&\leq f(w^*) + \langle \nabla p(w_t) - \nabla \widetilde{p}_t(w_t), w_{t+1} - w^* \rangle \\
&\quad - \frac{1}{\alpha} \langle w_t - w_{t+1}, w^* - w_{t+1} \rangle + \frac{1}{2\alpha} \|w_t - w_{t+1}\|^2 \\
&= f(w^*) + \langle \nabla p(w_t) - \nabla \widetilde{p}_t(w_t), w_{t+1} - w^* \rangle \\
&\quad + \frac{1}{2\alpha} \left( \|w_t\|^2 - 2w_t^\top w^* + \|w^*\|^2 + 2w_{t+1}^\top w^* - \|w^*\|^2 - \|w_{t+1}\|^2 \right) \\
&= f(w^*) + \langle \nabla p(w_t) - \nabla \widetilde{p}_t(w_t), w_{t+1} - w^* \rangle \\
&\quad + \frac{1}{2\alpha} \left( \|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2 \right) && (22)
\end{aligned}$$

Define $\Delta = I - S_A^\top S_A$, and consider the second term on the right-hand side of (22).

$$\begin{aligned}
&\langle \nabla p(w_t) - \nabla \widetilde{p}_t(w_t), w_{t+1} - w^* \rangle = \left\langle X^\top \Delta (X w_t - y), w_{t+1} - w^* \right\rangle \\
&= \langle \Delta (X w_t - y), X w_{t+1} - y \rangle - \langle \Delta (X w_t - y), X w^* - y \rangle \\
&= \frac{1}{2} \Big[ \left( X \left( w_t + w_{t+1} \right) - 2y \right)^\top \Delta \left( X \left( w_t + w_{t+1} \right) - 2y \right) \\
&\quad - \left( X w_{t+1} - y \right)^\top \Delta \left( X w_{t+1} - y \right) + \left( X w^* - y \right)^\top \Delta \left( X w^* - y \right) \\
&\quad - \left( X \left( w_t + w^* \right) - 2y \right)^\top \Delta \left( X \left( w_t + w^* \right) - 2y \right) \Big] \\
&= 2 \left( X \left( \frac{w_t + w_{t+1}}{2} \right) - y \right)^\top \Delta \left( X \left( \frac{w_t + w_{t+1}}{2} \right) - y \right) \\
&\quad - 2 \left( X \left( \frac{w_t + w^*}{2} \right) - y \right)^\top \Delta \left( X \left( \frac{w_t + w^*}{2} \right) - y \right) \\
&\quad - \frac{1}{2} \left( X w_{t+1} - y \right)^\top \Delta \left( X w_{t+1} - y \right) + \frac{1}{2} \left( X w^* - y \right)^\top \Delta \left( X w^* - y \right) \\
&\leq 4\epsilon p \left( \frac{w_t + w_{t+1}}{2} \right) + 4\epsilon p \left( \frac{w_t + w^*}{2} \right) + \epsilon p(w_{t+1}) + \epsilon p(w^*) \\
&\overset{(a)}{\leq} 2\epsilon p(w_t) + 2\epsilon p(w_{t+1}) + 2\epsilon p(w_t) + 2\epsilon p(w^*) + \epsilon p(w_{t+1}) + \epsilon p(w^*) \\
&= \epsilon \left[ 4p(w_t) + 3p(w_{t+1}) + 3p(w^*) \right] \\
&\overset{(b)}{\leq} \epsilon \left[ 4f(w_t) + 3f(w_{t+1}) + 3f(w^*) \right],
\end{aligned}$$

where (a) is by convexity of $p(w)$ and Jensen's inequality, and (b) follows by non-negativity of $h$. Plugging this back in (22),

$$(1-3\epsilon)\, f(w_{t+1}) - 4\epsilon f(w_t) \leq (1+3\epsilon)\, f(w^*) + \frac{1}{2\alpha}\left(\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2\right).$$

Adding this for $t = 0, \ldots, (T-1)$,

$$\begin{aligned}(1-7\epsilon)\sum_{t=1}^{T-1} f(w_t) + (1-3\epsilon) f(w_T) &\leq T\,(1+3\epsilon)\, f(w^*) + 4\epsilon f(w_0) + \frac{1}{2\alpha}\left(\|w_0 - w^*\|^2 - \|w_T - w^*\|^2\right) \\ \Rightarrow (1-7\epsilon)\sum_{t=1}^{T} f(w_t) &\leq T\,(1+3\epsilon)\, f(w^*) + 4\epsilon f(w_0) + \frac{1}{2\alpha}\left(\|w_0 - w^*\|^2 - \|w_T - w^*\|^2\right) \\ &\leq T\,(1+3\epsilon)\, f(w^*) + 4\epsilon f(w_0) + \frac{1}{2\alpha}\|w_0 - w^*\|^2.\end{aligned}$$

Defining $\bar{f}_t = \frac{1}{T}\sum_{t=1}^{T} f(w_t)$, and $\kappa = \frac{1+3\epsilon}{1-7\epsilon}$, we get

$$\bar{f}_T - \kappa f(w^*) \leq \frac{4\epsilon f(w_0) + \frac{1}{2\alpha}\|w_0 - w^*\|^2}{(1-7\epsilon)\, T},$$

which proves the first part of the theorem. To establish the second part of the theorem, note that the convexity of $h$ implies

$$h(w_t) \geq h(w_{t+1}) + \langle g, w_t - w_{t+1}\rangle,$$

where $g \in \partial h(w_{t+1})$. By the optimality condition (18), this implies

$$\lambda h(w_t) \geq \lambda h(w_{t+1}) - \langle \nabla \widetilde{p}_t(w_t), w_t - w_{t+1}\rangle + \frac{1}{\alpha}\|w_{t+1} - w_t\|^2.$$

Combining this with the smoothness condition of $p(w)$,

$$p(w_{t+1}) \leq p(w_t) + \langle \nabla p(w_t), w_{t+1} - w_t\rangle + \frac{M}{2}\|w_{t+1} - w_t\|^2$$

and using the fact that $\alpha < \frac{1}{M}$, we have

$$f(w_{t+1}) \leq f(w_t) + \langle \nabla p(w_t) - \nabla \widetilde{p}_t(w_t), w_{t+1} - w_t\rangle - \frac{1}{2\alpha}\|w_t - w_{t+1}\|^2.$$

As in the previous analysis, we can show that

$$\langle \nabla p(w_t) - \nabla \widetilde{p}_t(w_t), w_{t+1} - w_t\rangle \leq \epsilon\left[7 f(w_t) + 3 f(w_{t+1})\right],$$

and therefore

$$\begin{aligned} f(w_{t+1}) &\leq \frac{1+7\epsilon}{1-3\epsilon} f(w_t) - \frac{1}{2\alpha(1-3\epsilon)}\|w_t - w_{t+1}\|^2 \\ &\leq \frac{1+7\epsilon}{1-3\epsilon} f(w_t).\end{aligned}$$

## Appendix C. Proof of Theorem 6

For an iterate $v_t$, let $w_t := Sv_t$. Define the solution set $\mathcal{S} = \arg\min_w g(w)$, and $w_t^* = \mathcal{P}_{\mathcal{S}}(w_t)$, where $\mathcal{P}_{\mathcal{S}}(\cdot)$ is the projection operator onto the set $\mathcal{S}$. Let $v_t^*$ be such that $w_t^* = S^\top v_t^*$, which always exists since $S$ has full column rank.

We also define $L' := L(1+\epsilon)$, and $g^* = \min_w g(w) = g(w_t^*)$ for any $t$.

### C.1. Lemmas

**Lemma 15** *$\widetilde{g}(v)$ is $L'$-smooth.*

**Proof** For any $u, v$,

$$\begin{aligned}\widetilde{g}(u) = g(S^\top u) &\le g(S^\top v) + \langle \nabla g(S^\top v), S^\top(u-v)\rangle + \frac{L}{2}\|S^\top(u-v)\|^2,\\ &\overset{(a)}{\le} g(S^\top v) + \langle S\nabla g(S^\top v), u-v\rangle + \frac{L(1+\epsilon)}{2}\|u-v\|^2,\\ &\overset{(b)}{=} \widetilde{g}(v) + \langle \nabla \widetilde{g}(v), u-v\rangle + \frac{L(1+\epsilon)}{2}\|u-v\|^2,\end{aligned}$$

where $(a)$ follows from smoothness of $g$, and from $(m,\eta,\epsilon)$-BRIP property, and $(b)$ is by the chain rule of derivatives and the definition of $\widetilde{g}(v)$. Therefore $\widetilde{g}$ is $L(1+\epsilon)$-smooth. ■

**Lemma 16** *For any $t$,*

$$\widetilde{g}^* := \min_v \widetilde{g}(v) = \min_w g(w) =: g^*.$$

**Proof** It is clear that

$$\min_v \tilde{g}(v) = \min_v g(S^\top v) \ge \min_w g(w).$$

To show the other direction, set $v^* = S(S^\top S)^{-1}w^*$, where $S^\top S$ is invertible since $S$ has full column rank. Then $g(w^*) = \tilde{g}(v^*) \ge \min_v \tilde{g}(v)$. ■

**Lemma 17** *If $g$ is $\nu$-restricted-strongly convex, then*

$$g(w) - g^* \ge \frac{\nu}{2}\|w - w^*\|^2,$$

*where $w^* = \mathcal{P}_{\mathcal{S}}(w)$.*

**Proof** We follow the proof technique in Zhang and Yin (2013). We have

$$\begin{aligned}g(w) &= g^* + \int_0^1 \langle \nabla g(w^* + \tau(w-w^*)), w - w^*\rangle d\tau\\ &= g^* + \int_0^1 \frac{1}{\tau}\langle \nabla g(w^* + \tau(w-w^*)), \tau(w-w^*)\rangle d\tau\end{aligned}$$

$$\begin{aligned}
&\overset{(a)}{\geq} g^* + \int_0^1 \frac{1}{\tau}\nu\tau^2\|w - w^*\|^2 d\tau \\
&= g^* + \nu\|w - w^*\|^2 \int_0^1 \tau d\tau \\
&= g^* + \frac{1}{2}\nu\|w - w^*\|^2,
\end{aligned}$$

which is the desired result, where in (a) we used $\nu$-restricted strong convexity, and the fact that

$$\mathcal{P}_{\mathcal{S}}(w^* + \tau(w - w^*)) = w^*,$$

for all $\tau \in [0, 1]$, since $w^* = \mathcal{P}_{\mathcal{S}}(w)$ is the orthogonal projection. ■

### C.2. Proof of Theorem 6

Recall that the step for block $i$ at time $t$, $\Delta_{i,t}$, is defined by

$$\Delta_{i,t} := \begin{cases} -\alpha\nabla_i\widetilde{g}(v_{t-1}), & \text{if } i \in A_t \\ 0, & \text{otherwise.} \end{cases}$$

By smoothness and definition of $\Delta_t$,

$$\begin{aligned}
\widetilde{g}(v_{t+1}) - \widetilde{g}(v_t) &\leq \langle\nabla\widetilde{g}(v_t), \Delta_t\rangle + \frac{L'}{2}\|\Delta_t\|^2 \\
&= \sum_{i\in A_t}\left(\langle\nabla_i\widetilde{g}(v_t), \Delta_{i,t}\rangle + \frac{L'}{2}\|\Delta_{i,t}\|^2\right) \\
&= \sum_{i\in A_t}\left(-\frac{1}{\alpha}\langle\Delta_{i,t}, \Delta_{i,t}\rangle + \frac{L'}{2}\|\Delta_{i,t}\|^2\right) \\
&= -\left(\frac{1}{\alpha} - \frac{L'}{2}\right)\|\Delta_t\|^2. \qquad (23)
\end{aligned}$$

Now, for any $t$,

$$\begin{aligned}
\widetilde{g}(v_t) - \widetilde{g}^* &\leq \langle\nabla\widetilde{g}(v_t), v_t^* - v_t\rangle = \left\langle S\nabla g(S^\top v_t), v_t^* - v_t\right\rangle \\
&\overset{(a)}{\leq} \left\|\nabla g(S^\top v_t)\right\| \cdot \left\|S^\top\left(v_t^* - v_t\right)\right\| = \left\|\nabla g(S^\top v_t)\right\| \cdot \|w_t^* - w_t\|, \qquad (24)
\end{aligned}$$

where $(a)$ is due to Cauchy-Schwartz inequality. Using

$$\Delta_t = -\alpha P_t \begin{bmatrix} S_{A_t}\nabla g(S^\top v_t) \\ 0 \end{bmatrix},$$

where $P_t$ is a block permutation matrix mapping $\{1, \ldots, k\}$ to the node indices in $A_t$, we have

$$\|\Delta_t\|^2 = \alpha^2\nabla g(S^\top v_t)^\top S_{A_t}^\top P_t^\top P_t S_{A_t}\nabla g(S^\top v_t) \geq (1 - \epsilon)\alpha^2\left\|\nabla g(S^\top v_t)\right\|^2. \qquad (25)$$

Because of (23), we have

$$\widetilde{g}(v_{t+1}) - \widetilde{g}(v_t) = g(w_{t+1}) - g(w_t) \leq 0$$

when $\alpha < 2/L'$, and hence $w_t$ is contained in the level set defined by the initial iterate for all $t$, *i.e.*,

$$w_t \in \{w : g(w) \leq g(w_0)\}.$$

By the diameter assumption on this set, we have $\|w_t - w_t^*\| \leq R$ for all $t$. Using this and (25) in (24), we get

$$\widetilde{g}(v_t) - \widetilde{g}^* \leq \frac{R}{\alpha}\sqrt{\frac{1}{1-\epsilon}}\|\Delta_t\|.$$

Note that the fact that $\epsilon > 0$ weakens the per-step bound here, since we do not take the steepest descent direction. Another penalty from non-zero $\epsilon > 0$ comes due to the increase in the smoothness parameter of the objective $L' = (1+\epsilon)L$, which slightly limits the maximum step size.

Combining the above with (23),

$$\widetilde{g}(v_{t+1}) - \widetilde{g}(v_t) \leq -\frac{(1-\epsilon)\alpha}{R^2}\left(1 - \frac{\alpha L'}{2}\right)(\widetilde{g}(v_t) - \widetilde{g}^*)^2.$$

Defining $\pi_t := \widetilde{g}(v_t) - \widetilde{g}^*$, and $C := \frac{(1-\epsilon)\alpha}{R^2}\left(1 - \frac{\alpha L'}{2}\right)$, this implies

$$\pi_{t+1} \leq \pi_t - C\pi_t^2.$$

Dividing both sides by $\pi_t\pi_{t+1}$, and noting that $\pi_{t+1} \leq \pi_t$ due to (23),

$$\frac{1}{\pi_t} \leq \frac{1}{\pi_{t+1}} - C\frac{\pi_t}{\pi_{t+1}} \leq \frac{1}{\pi_{t+1}} - C$$

Therefore

$$\frac{1}{\pi_t} \geq \frac{1}{\pi_0} + Ct,$$

which implies

$$\pi_t \leq \frac{1}{\frac{1}{\pi_0} + Ct}.$$

Since $g(w_t) = g(S^\top v_t) = \widetilde{g}(v_t)$ by definition, and $g^* = \widetilde{g}^*$ by Lemma 16, $\pi_t = g(w_t) - g^*$, and therefore we have established the first part of the theorem.

To prove the second part, we make the additional assumption that $g$ satisfies $\nu$-restricted-strong convexity, which, through Lemma 17, implies $g(w) - g^* \geq \frac{\nu}{2}\|w - w^*\|^2$, for $w^* = \mathcal{P}_{\mathcal{S}}(w)$. Plugging in $w = w_t$ then gives the bound

$$\|w_t - w_t^*\|^2 \leq \frac{2}{\nu}\pi_t.$$

Using this bound as well as (25) in (24), we have

$$\pi_t^2 \leq \frac{2\|\Delta_t\|^2}{\nu(1-\epsilon)\alpha^2}\pi_t.$$

Using (23), this gives

$$\pi_t \leq \frac{2}{\nu(1-\epsilon)\alpha^2}\left(\frac{1}{\alpha}-\frac{L'}{2}\right)^{-1}(\pi_t-\pi_{t+1}),$$

which, defining $\xi = \frac{2}{\nu(1-\epsilon)\alpha}\left(1-\frac{L'\alpha}{2}\right)^{-1}$, results in

$$\pi_t \leq \left(1-\frac{1}{\xi}\right)^t \pi_0,$$

which shows the desired result.

## Appendix D. Full results of the Matrix factorization experiment

Tables 6 and 7 give the test and train RMSE for the Movielens 1-M recommendation task, with a random 80/20 train/test split.

| | uncoded | replication | gaussian | paley | hadamard |
|---|---|---|---|---|---|
| | $m = 8$, $k = 1$ | | | | |
| train RMSE | 0.804 | 0.783 | 0.781 | **0.775** | 0.779 |
| test RMSE | 0.898 | 0.889 | 0.877 | **0.873** | 0.874 |
| runtime | 1.60 | 1.76 | 2.24 | 1.82 | 1.82 |
| | $m = 8$, $k = 4$ | | | | |
| train RMSE | 0.770 | 0.766 | 0.765 | **0.763** | 0.765 |
| test RMSE | 0.872 | 0.872 | **0.866** | 0.868 | 0.870 |
| runtime | 2.96 | 3.13 | 3.64 | 3.34 | 3.18 |
| | $m = 8$, $k = 6$ | | | | |
| train RMSE | 0.762 | 0.760 | 0.762 | **0.758** | 0.760 |
| test RMSE | 0.866 | 0.871 | 0.864 | **0.860** | 0.864 |
| runtime | 5.11 | 4.59 | 5.70 | 5.50 | 5.33 |

Table 6: Full results for Movielens 1-M, distributed over $m = 8$ nodes total. Runtime is in hours. An uncoded scheme running full batch L-BFGS has a train/test RMSE of 0.756 / 0.861, and a runtime of 9.58 hours.

| | uncoded | replication | gaussian | paley | hadamard |
|---|---|---|---|---|---|
| | $m = 24$, $k = 3$ | | | | |
| train RMSE | 0.805 | 0.791 | 0.783 | **0.780** | 0.782 |
| test RMSE | 0.902 | 0.893 | 0.880 | **0.879** | 0.882 |
| runtime | 2.60 | 3.22 | 3.98 | 3.49 | 3.49 |
| | $m = 24$, $k = 12$ | | | | |
| train RMSE | 0.770 | **0.764** | 0.767 | **0.764** | 0.765 |
| test RMSE | 0.872 | 0.870 | **0.866** | 0.868 | 0.868 |
| runtime | 4.24 | 4.38 | 4.92 | 4.50 | 4.61 |

Table 7: Full results for Movielens 1-M, distributed over $m = 24$ nodes total. Runtime is in hours. An uncoded scheme running full batch L-BFGS has a train/test RMSE of 0.757 / 0.862, and a runtime of 14.11 hours.

## References


Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. Effective straggler mitigation: Attack of the clones. In *NSDI*, volume 13, pages 185–198, 2013.

Albert S Berahas, Jorge Nocedal, and Martin Takác. A multi-batch l-bfgs method for machine learning. In *Advances in Neural Information Processing Systems*, pages 1055–1063, 2016.

Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12): 5406–5425, 2006.

Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56 (2):74–80, 2013.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische mathematik*, 117(2):219–249, 2011.

Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. Short-dot: Computing large linear transforms distributedly using coded short dot products. In *Advances In Neural Information Processing Systems*, pages 2092–2100, 2016.

Matthew Fickus, Dustin G Mixon, and Janet C Tremain. Steiner equiangular tight frames. *Linear algebra and its applications*, 436(5):1014–1027, 2012.

Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytia. Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):347–360, 2015.

Stuart Geman. A limit theorem for the norm of random matrices. *The Annals of Probability*, pages 252–261, 1980.

J.M. Goethals and J Jacob Seidel. Orthogonal matrices with zero diagonal. *Canad. J. Math*, 1967.

Wael Halbawi, Navid Azizan-Ruhi, Fariborz Salehi, and Babak Hassibi. Improving distributed gradient descent using reed-solomon codes. *arXiv preprint arXiv:1706.05436*, 2017.

Martin Jaggi. An equivalence between the lasso and support vector machines. *arXiv preprint arXiv:1303.1152*, 2013.

Can Karakus, Yifan Sun, and Suhas Diggavi. Encoded distributed optimization. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2890–2894. IEEE, 2017a.

Can Karakus, Yifan Sun, Suhas Diggavi, and Wotao Yin. Straggler mitigation in distributed optimization through data encoding. In *Advances in Neural Information Processing Systems*, pages 5440–5448, 2017b.

Ming-Jun Lai and Wotao Yin. Augmented \ell_1 and nuclear-norm models with a globally linearly convergent algorithm. *SIAM Journal on Imaging Sciences*, 6(2):1059–1091, 2013.

Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2018.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.

Yu-Feng Li, Ivor W Tsang, Jame Kwok, and Zhi-Hua Zhou. Tighter and convex maximum margin clustering. In *Artificial Intelligence and Statistics*, pages 344–351, 2009.

Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.

Michael W Mahoney et al. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory BFGS. *Journal of Machine Learning Research*, 16:3151–3181, 2015.

Raymond EAC Paley. On orthogonal matrices. *Studies in Applied Mathematics*, 12(1-4): 311–320, 1933.

Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

Mert Pilanci and Martin J Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Transactions on Information Theory*, 61(9):5096–5115, 2015.

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

Amirhossein Reisizadeh, Saurav Prakash, Ramtin Pedarsani, and Salman Avestimehr. Coded computation over heterogeneous clusters. In *Information Theory (ISIT), 2017 IEEE International Symposium on*, pages 2408–2412. IEEE, 2017.

J Riedl and J Konstan. Movielens dataset, 1998.

Nihar B Shah, Kangwook Lee, and Kannan Ramchandran. When do redundant requests reduce latency? *IEEE Transactions on Communications*, 64(2):715–722, 2016.

Jack W Silverstein. The smallest eigenvalue of a large dimensional wishart matrix. *The Annals of Probability*, pages 1364–1368, 1985.

Tao Sun, Robert Hannah, and Wotao Yin. Asynchronous coordinate descent under more realistic assumptions. In *Advances in Neural Information Processing Systems*, pages 6183–6191, 2017.

Ferenc Szöllősi. Complex hadamard matrices and equiangular tight frames. *Linear Algebra and its Applications*, 438(4):1962–1967, 2013.

Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis. Gradient coding. *ML Systems Workshop (MLSyS), NIPS*, 2016.

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.

Da Wang, Gauri Joshi, and Gregory Wornell. Using straggler replication to reduce latency in large-scale parallel computing. *ACM SIGMETRICS Performance Evaluation Review*, 43(3):7–11, 2015.

Lloyd Welch. Lower bounds on the maximum cross correlation of signals (corresp.). *IEEE Transactions on Information theory*, 20(3):397–399, 1974.

N J. Yadwadkar, B. Hariharan, J. Gonzalez, and R H. Katz. Multi-task learning for straggler avoiding predictive job scheduling. *Journal of Machine Learning Research*, 17(4):1–37, 2016.

Yaoqing Yang, Pulkit Grover, and Soummya Kar. Coded distributed computing for inverse problems. In *Advances in Neural Information Processing Systems*, pages 709–719, 2017.

Yang You, Xiangru Lian, Ji Liu, Hsiang-Fu Yu, Inderjit S Dhillon, James Demmel, and Cho-Jui Hsieh. Asynchronous parallel greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pages 4682–4690, 2016.

Hui Zhang and Wotao Yin. Gradient methods for convex minimization: better rates under weaker conditions. *arXiv preprint arXiv:1303.4645*, 2013.