SGD with Low-Dimensional Gradients with Applications to Private and Distributed Learning

Shiva Prasad Kasiviswanathan[1]

[1]Amazon, Palo Alto, USA

Abstract

In this paper, we consider constrained optimization problems subject to a convex set $\mathcal{C}$. Stochastic gradient descent (SGD) is a simple and popular stochastic optimization algorithm that has been the workhorse of machine learning for many years. We show a new and surprising fact about SGD, in that depending on the constraint set $\mathcal{C}$, one can operate SGD with much lower-dimensional stochastic gradients without affecting its performance. In particular, we design an optimization algorithm that operates with the lower-dimensional (compressed) stochastic gradients, and establish that with the right set of parameters it has the exact same dimension-free convergence guarantees as that of regular SGD for popular convex and nonconvex optimization settings. We also present two applications of these bounds, one for improving the empirical risk minimization bounds in differentially private nonconvex optimization, and other for reducing communication costs with distributed SGD. Additionally, we also show that this connection between constraint set structure and gradient compression also extends beyond SGD to the conditional gradient (Frank-Wolfe) method. The geometry of the constraint set, captured by its Gaussian width, plays an important role in all our results.

1 INTRODUCTION

In this paper, we consider the classic optimization problem of minimizing a function over a convex set:

$$\min_{\mathbf{w} \in \mathcal{C}} F(\mathbf{w}), \tag{1}$$

where $\mathcal{C} \subseteq \mathbb{R}^d$ is a compact convex set and $F : \mathbb{R}^d \to \mathbb{R}$.

(Projected) Stochastic Gradient Descent (SGD) is one of the simplest and most popular stochastic first-order optimization algorithms for solving (1). Stochastic gradient descent is widely used in machine learning applications due to its favorable computational scalability properties. This is notably true in the deep learning setting, where gradients can be computed efficiently via backpropagation. In convex optimization, SGD can be used to optimize any convex function F over a convex domain $\mathcal{C}$, given access only to unbiased estimates of F's gradients (or more generally, subgradients[1]) through an oracle. This feature makes it very useful for learning problems, where our goal is to minimize generalization error based only on a finite sampled training set.

While classical results have focused on analyzing the performance of SGD in convex optimization problems, the most notable recent successes in machine learning have involved nonconvex optimization. For example, in the unconstrained setting (i.e., $\mathcal{C} = \mathbb{R}^d$) for a differentiable function that has an μ-Lipschitz continuous gradient (i.e., μ-*smooth*), it is well known that SGD finds an α-*first-order stationary point* (a point $\mathbf{w}$ with $\|\nabla F(\mathbf{w})\| \leq \alpha$) with $O(\mu(F(\mathbf{w}_1) - F^\star)/\alpha^2)$ iterations [Nesterov, 1998], where $\mathbf{w}_1$ is the initial point and $F^\star$ is the optimal value of F. For constrained nonconvex smooth functions, a variety of similar dimension-free convergence results are known under the appropriate notion of stationarity [Ghadimi et al., 2016, Mokhtari et al., 2018]. These dimension-free convergence guarantees make SGD extremely attractive when the parameter space is very high dimensional.

1.1 OUR RESULTS

Our main result is a novel connection between constraint set structure and compression of the gradients.[2] We design a new SGD-style optimization algorithm that operates with

[1]Following a common convention, we still refer to the algorithm in this case as "gradient descent".

[2]The term *compressed gradient* has been used in a variety of contexts in prior literature, here we use the term to denote a lower-dimensional representation of the gradient.

just the lower-dimensional (compressed) stochastic gradients and has the exact same dimension-free convergence guarantees as that of regular SGD for common convex and nonconvex optimization settings. Formally, instead of the usual stochastic gradient oracle, we assume the existence of a *compressed stochastic gradient* oracle, that on inputs $\Phi \in \mathbb{R}^{m \times d}$ and $\mathbf{w}_t \in \mathcal{C}$, returns $\vartheta_t = \Phi \hat{\mathbf{g}}_t \in \mathbb{R}^m$ where $\hat{\mathbf{g}}_t$ is a stochastic subgradient of F at $\mathbf{w}_t$, and m could be much smaller than d. Our new SGD procedure (Algorithm COMPSGD), that has access only to this compressed stochastic gradient oracle, first projects the current iterate $\mathbf{w}_t$ onto the lower-dimensional space using Φ, then updates the iterate here using the compressed stochastic gradient oracle, and then "lifts" back the result to $\mathcal{C}$ to get the new iterate. We use ideas from geometry and high-dimensional estimation to perform this lifting. An immediate advantage of using compressed gradients (over regular SGD) comes in a distributed setup for reducing the cost of transmitting gradients.

We next investigate the convergence guarantees of this compressed SGD algorithm for various classes of functions. Our interest will be on Φ's that are popularly referred to as *random projection* matrices such as subgaussian random matrices or sparse random matrices. Our analysis is based on exploiting the geometric structure of $\mathcal{C}$. We assume that in each iteration t of the algorithm, compressed stochastic gradient oracle is invoked using an *independent* random projection matrix Φ_t, and provide conditions on the learning rate and the projection dimension under which the compressed SGD has the same (up to constant factors) dimension-free convergence guarantees as regular (uncompressed) SGD.

More precisely, let $\Phi_t \in \mathbb{R}^{m_t \times d}$ be the random projection matrix used in iteration t with the compressed stochastic gradient oracle. We show that with appropriate setting of m_t one could match the regular SGD guarantees. The geometric parameter, *Gaussian width*, defined as $\omega(\mathcal{C}) = \mathbb{E}_{\mathbf{r} \in \mathcal{N}(0,1)^d}[\sup_{\mathbf{w} \in \mathcal{C}} \langle \mathbf{w}, \mathbf{r} \rangle]$ plays an important role in our analysis, and shows up repeatedly as a geometric measure of the size of the set $\mathcal{C}$. Gaussian width roughly captures the expected width of $\mathcal{C}$ along a random direction. Many constraint sets have a small Gaussian width, for example, a common choice of $\mathcal{C}$ for sparsity purposes, the ℓ_1-ball in $\mathbb{R}^d$, has a width of $O(\sqrt{\log d})$. Setting, $m_t = \Omega(\omega(\mathcal{C})^2/\beta_t^2)$, for a parameter $\beta_t > 0$, we establish the following bounds for the convergence guarantees of our compressed SGD algorithm over T iterations. For simplicity, we ignore dependence on other parameters such as variance of the stochastic gradient, the diameter of the convex set $\mathcal{C}$, etc.

(a) Nonconvex Smooth Functions: For a nonconvex function F that is μ-smooth, we investigate two measures of (non)stationarity. Firstly, we show that a minibatch version of compressed SGD converges to α-first-order stationary point (α-FOSP)[3] in $T = \Omega(\mu(F(\mathbf{w}_1) - F^\star)/\alpha^2)$ iterations with $\eta_t = 1/\mu$ and $\beta_t \approx \alpha^2/\mu$ for all $t \in [T]$, and appropriate minibatch size (Theorem 2.4). We also investigate another popular measure of (non)stationarity defined through *gradient mapping*, which is a natural generalization of gradient for constrained problems [Nesterov, 1998, Ghadimi et al., 2016] (see Definition 8), and reach to a similar conclusion (Theorem B.5).

(b) Convex and Strongly Convex Functions: In this case, the goal is to bound the expected optimization error, defined as $\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^\star)]$, where $\mathbf{w}^\star \in \mathcal{C}$ is some minimizer of F. For a general convex function F (without any smoothness assumption), setting $\beta_t = 1/t$ and $\eta_t = 1/\sqrt{t}$, we get an error bound of $O(\log(T)/\sqrt{T})$ (Theorem B.9). For a λ-strongly convex functions F, setting $\beta_t = 1/t$ and $\eta_t = 1/(\lambda t)$, we get a $O(\mu/(\lambda^2 T))$ bound on the expected error if F is μ-smooth and a bound of $O(\log T/(\lambda T))$ without smoothness (Theorems B.7 and B.8). These match the known regular SGD bounds, see, e.g., [Shamir and Zhang, 2013, Rakhlin et al., 2011].

These results demonstrate that for many problems, the compression of the gradients comes for "free". The intuition behind this connection between constraint set structure and compression is as follows. Given a Gaussian random matrix and a finite set S of vectors, it is well-known (e.g., Johnson-Lindenstrauss Lemma) that projection with this matrix preserves the norms of all these vectors in S when the projection dimension is roughly $\log(|S|)$ [Johnson and Lindenstrauss, 1984]. More generally, by measuring the complexity of the constraint set through the notion of Gaussian width, then the above compression guarantee can be achieved with a projection dimension that is roughly square of the Gaussian width of S [Gordon, 1988]. In our case, we compress the gradients (and not the iterates), and we show that by setting the projection dimension based on the square of the Gaussian width of the constraint set, suffices to well-approximate various norms that matter in convergence results (for example, the potential function defined as $\|\mathbf{w}_t - \mathbf{w}\|$ for $\mathbf{w} \in \mathcal{C}$).

To achieve this, we carefully combine ideas from modern SGD analyses, with ideas in convex geometry and high-dimensional estimation. One of main challenge comes in ensuring that the noise introduced by the random projection is always bounded. The compressed SGD algorithm updates the iterates in a lower-dimensional projected space, but we track the SGD progress in the original higher-dimensional space. To the best of our knowledge, these are the *first* rigorous results in SGD based on strictly utilizing *lower-dimensional* gradients.

Applications. We mention two applications of these com-

[3] $\bar{\mathbf{w}} \in \mathcal{C}$ is an α-first-order stationary point if for all $\mathbf{w} \in \mathcal{C}$, $\langle \nabla F(\bar{\mathbf{w}}), \mathbf{w} - \bar{\mathbf{w}} \rangle \geq -\alpha$, $\nabla F(\bar{\mathbf{w}})$ denotes the gradient of F at $\bar{\mathbf{w}}$.

pressed SGD results. Both these problems benefit from the reduced dimensionality of the gradients.

(i) Differentially Private Empirical Risk Minimization (ERM). Machine learning algorithms are frequently run on sensitive data, and this has motivated the study of learning algorithms that have good performance guarantees while providing strong (mathematically proven) privacy protections for the training data. Differential Privacy (DP) is a formal algorithmic guarantee provides provable protection against adversaries with arbitrary side information and computational power, allows clear quantification of privacy losses, and satisfies graceful composition over multiple access to the same data [Dwork et al., 2006].
We provide the first results in differentially private nonconvex optimization for achieving first-order stationarity where the sample size n needed for a non-trivial result grows as $\omega(\mathcal{C})$ and not as $\sqrt{d}$ (see Table 1). As an example, if $\mathcal{C}$ is the ℓ_1-ball, then $\omega(\mathcal{C}) = O(\sqrt{\log d})$, and there is roughly an exponential improvement from $\sqrt{d}$ to $\sqrt{\log d}$ in the sample size needed compared to [Wang et al., 2017, Zhang et al., 2017].

(ii) Reducing Communication Costs in Distributed Synchronous SGD. Distributed stochastic gradient descent plays a very important role in distributed learning. In this work, we consider the data-distributed model of distributed SGD where the data sets are partitioned across various compute nodes. In each iteration of synchronous SGD, the compute nodes send their computed local gradients to a parameter server that averages and updates the global parameter. Since clients could be constrained devices that are on slow/expensive connections, communication cost is a principal constraint, especially, the cost of uploading gradients back to the server.
Firstly, our approach of utilizing lower-dimensional gradients already provides a way of reducing communication costs in many settings, without any change in the convergence rate. In addition, we show that combining this idea with any other gradient communication cost reduction scheme that provides unbiased estimates of the gradient can lead to even further communication cost savings, however now with an increase in convergence rate that only depends on the other chosen scheme. Our results provide the *first* communication cost bounds for various distributed optimization problems that depend on the geometry of $\mathcal{C}$ (see Table 2).

Extension to Conditional Gradient Method. We show that the above-mentioned connection between constraint set and gradient compression extends beyond SGD to conditional gradient (Frank-Wolfe) method [Frank and Wolfe, 1956]. This method is a natural candidate for constrained optimization because of its projection free property and its ability to handle structured constraints [Jaggi, 2013]. We present a Frank-Wolfe style optimization algorithm that utilizes the compressed stochastic gradient oracle for its gradient evaluations, and requires a linear minimization oracle over the set $\Phi\mathcal{C}$. As in the case of SGD, we show that a gradient dimension that is based on the square of the Gaussian width of $\mathcal{C}$ suffices to obtain convergence bounds that match those of regular stochastic Frank-Wolfe method in convex/nonconvex settings (Theorems E.1 and E.2, Appendix E). Again, these are the first rigorous results for the Frank-Wolfe method based on strictly utilizing *lower-dimensional* gradients.

Due to space limitation, we leave many details, formal statements, proofs, and experimental studies in the supplement.

1.2 RELATED WORK

There has been a growing interest in understanding the convergence properties of SGD where instead of the true gradients some quantized/sparsified/sketched version of gradients is used in the SGD update step. The most common application of these techniques are in distributed/federated learning with the goal of reducing the communication costs, e.g., [Seide et al., 2014, Strom, 2015, De Sa et al., 2015, Konecnỳ et al., 2016, Wen et al., 2017, Alistarh et al., 2017, Agarwal et al., 2018, Lin et al., 2017, Khirirat et al., 2018, Bernstein et al., 2018, Wu et al., 2018, Wang et al., 2018, Karimireddy et al., 2019, Stich et al., 2018, Mishchenko et al., 2019, Acharya et al., 2019, Alistarh et al., 2018, Ivkin et al., 2019, Gandikota et al., 2019, Horváth et al., 2019]. The general idea here is that a client could shrink the gradient communication cost by applying some encoding on the gradient before transmitting it to the server. Generally, these schemes come with an *increase* in the gradient variance. In other words, these schemes are generally "lossy" and will result in a slower convergence that using the true gradients. This is one major difference compared to our results, where we show that with the right set of parameters based on the geometry of the constraint set, our dimensionality-reduction scheme is "lossless", in that we get the same convergence rate as if using the true gradients.

We also investigate whether our idea of utilizing lower-dimensional gradients can be combined with these existing gradient encoding techniques. Existing gradient encoding techniques can be categorized as either *unbiased* or *biased* based on whether the gradient estimates are unbiased or not. A number of gradient quantization methods are crafted specifically to yield unbiased estimates [Alistarh et al., 2017, Wen et al., 2017, Wang et al., 2018, Gandikota et al., 2019]. We show that one can combine our dimensionality-reduction scheme with *any* unbiased gradient encoding technique to get a reduction in communication cost at the expense of increased variance in gradient estimate.

We note that a number of gradient encoding techniques also produce biased gradient estimates [Seide et al., 2014, Strom, 2015, Bernstein et al., 2018]. Using the idea of error-feedback, recently [Stich et al., 2018, Karimireddy et al., 2019] gave convergence guarantees for this kind of biased compression algorithm, showing that accumulating compression error locally in the workers can overcome the bias in the weight updates as long as the compression algorithm obeys certain properties. On a related note, gradient sparsification techniques with provable convergence were studied in [Alistarh et al., 2018, Stich et al., 2018]. Another idea proposed here is to use the *count-sketch* from the sketching literature for reducing communication [Ivkin et al., 2019, Rothchild et al., 2020]. An important point is that, in general, none of these prior encoding techniques inherently change the dimensionality of the gradient and also suffer from inflated variance at high compression rates. Also, in general it is hard to compared biased vs. unbiased approaches because of the different guarantees they are shown to provide. We leave the question of combining our dimensionality reduction idea with these biased gradient encoding schemes and/or stochastic variance reduction ideas [Alistarh et al., 2017, Horváth et al., 2019] for future work.

Another orthogonal idea to reduce communication, especially in the context of federated learning, is the idea of *local SGD* [Stich, 2019, Basu et al., 2019], where the clients perform local updates on their local data, and the clients communicate with the server only after few rounds.

Beyond Gradient Descent. It is well-known that for certain problems such as minimizing on the simplex with subgradients bounded in ℓ_∞-norm mirror descent outperforms SGD. Our focus here is on techniques that apply to a broad class of problems, including variety of constraint sets and non-convex functions. Since SGD is arguably the most popular and general optimization approach, we focus on it here.

Comparison with some additional related work based on sketching the Hessian matrix, differentially private ERM is presented in Appendix A.1.

1.3 PRELIMINARIES

Notation. We denote $[n] = \{1, \ldots, n\}$. Vectors are denoted by boldface letters. We use $\mathbb{I}_p$ to denote a $p \times p$ identity matrix. For a vector $\mathbf{v}$, $\|\mathbf{v}\|$ denotes its Euclidean-norm. Throughout, we assume that $\mathcal{C} \subseteq \mathbb{R}^d$ is a compact convex set. We define the diameter of $\mathcal{C}$ as, $\|\mathcal{C}\| = \sup_{\mathbf{w},\mathbf{w}' \in \mathcal{C}} \|\mathbf{w} - \mathbf{w}'\|$. Given a $\Phi \in \mathbb{R}^{m \times d}$, we define $\Phi\mathcal{C} := \{\Phi\mathbf{w} : \mathbf{w} \in \mathcal{C}\}$. Define the projection of $\mathbf{w} \in \mathbb{R}^d$ onto a closed convex set $\mathcal{C} \subseteq \mathbb{R}^d$ as: $\Pi_{\mathcal{C}}(\mathbf{w}) := \text{argmin}_{\mathbf{w}' \in \mathcal{C}} \|\mathbf{w} - \mathbf{w}'\|$. We review some optimization fundamentals in Appendix A.2.

Gaussian width. Our analysis is based on exploiting the geometric properties of the constraint set. We use the well-studied quantity of *Gaussian width* that captures the ℓ_2-geometric complexity of a set of points. Given a set $S \subset \mathbb{R}^d$, its Gaussian width $\omega(S)$ is defined as: $\omega(S) := \mathbb{E}_{\mathbf{r} \in \mathcal{N}(0, \mathbb{I}_d)} [\sup_{\mathbf{a} \in S} \langle \mathbf{a}, \mathbf{r} \rangle]$.

Many interesting examples of S have low Gaussian width. For example, the ℓ_1-ball and simplex in $\mathbb{R}^d$ have width of $O(\sqrt{\log d})$, whereas a convex hull of l vectors with bounded ℓ_2-norm of c has width $O(c\sqrt{\log l})$. See Table 3 in Appendix A.2 for additional examples of commonly used sets with low Gaussian width.

Given a set S, the square of Gaussian width of S measures the dimension size one needs to take on a random projection to still approximately preserve the norms of all the points in S. For Gaussian random matrices Φ, this was first shown in [Gordon, 1988], popularly referred to as Gordon's theorem (Theorem A.3 in Appendix A.2). This was recently extended to matrices drawn from subgaussian distributions [Dirksen, 2014] or distributions over sparse matrices [Bourgain et al., 2015] (Theorem A.4 in Appendix A.2). We will use this norm-preservation property and its consequences throughout our analyses.

2 SGD WITH LOW-DIM. GRADIENTS

In a regular SGD setup, the assumption is that we do not know F, and the only information is through a stochastic gradient oracle, which given $\mathbf{w} \in \mathcal{C}$, produces a vector $\hat{\mathbf{g}}$ such that $\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g}$ is a subgradient of F at $\mathbf{w}$. In this section, we introduce a dimensionality reduced gradient setup, where the stochastic gradient oracle does not return $\hat{\mathbf{g}}$ but only a lower-dimensional projection of $\hat{\mathbf{g}}$, say obtained by applying a dimensionality reducing random projection to the subgradient. For a formal analysis, we define a compressed stochastic gradient oracle as follows.

Definition 1 (Compressed Stochastic Subgradient Oracle (CSFO))**.** *Upon receiving query* $\mathbf{w}$ *and* $\Phi \in \mathbb{R}^{m \times d}$*, the compressed stochastic gradient oracle returns* ϑ *where* $\vartheta = \Phi\hat{\mathbf{g}}$*. Here,* $\hat{\mathbf{g}}$ *is an independent random vector whose expectation* $\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g}$ *is a subgradient of* F *at* $\mathbf{w}$*. Borrowing from the stochastic optimization literature, we denote this oracle as the compressed stochastic first-order oracle (*CSFO*), and use the notation* $\vartheta = \text{CSFO}(\mathbf{w}, \Phi)$.

Note that with $\Phi = \mathbb{I}_d$, CSFO defaults to the standard subgradient (SFO) oracle. Implementing a CSFO oracle is easy, as it just computes a projection of the subgradient. Hence, it can be efficiently implemented for any problem for which subgradients can be computed efficiently. Our convergence results, provide bounds on the number of calls needed to a CSFO oracle to get accurate results. Transmitting Φ to the oracle is unnecessary as long as both the oracle and algorithm agree upon it. For example, as discussed in Section 3, in a distributed setup one can avoid transmitting Φ by using shared randomness between clients and server.

Note that when $m < d$, the $\vartheta = \text{CSFO}(\mathbf{w}, \Phi) \in \mathbb{R}^m$ has a dimensionality lower than d.[4] This creates an immediate issue with using CSFO as there is now a dimensionality mismatch for the SGD iterate update step. We first design an SGD-style algorithm that overcomes this problem and operates with only access to CSFO.

Our algorithm (Algorithm CompSGD) is based on a simple modification to the SGD procedure. Algorithm CompSGD is parameterized by two functions η_t and β_t, where η_t describes the learning rate and $\beta_t \in (0,1)$ relates to the dimension of the compressed gradient. In each iteration t, the algorithm picks an independent random projection matrix Φ_t of dimension $m_t \times d$. Various choices of Φ_t could work here. For example, Φ_t could be a Gaussian matrix with where each entry is sampled i.i.d. from $N(0, 1/m_t)$ or Φ_t could be a sparse JL matrix as defined by [Kane and Nelson, 2014]. Our approach can be informally described as: i) take an SGD step in the projected domain $\mathbb{R}^m_t$ using the lower-dimensional subgradient, ii) do a projection onto $\Phi\mathcal{C}$, and iii) lift back the solution to $\mathcal{C}$ to form the new iterate.

Algorithm CompSGD

```
Objective: $\min_{\mathbf{w}\in\mathcal{C}} F(\mathbf{w})$
Input: Convex set $\mathcal{C}$, learning rate parameters $\{\eta_t\}$, and projection dimension parameters $\{\beta_t\}$.
  1. Pick $\mathbf{w}_1$ as any point in $\mathcal{C}$
  2. for $t = 1$ to $T$ do
     a. Set $m_t \leftarrow \Omega\left(\min\{d, \omega(\mathcal{C})^2/\beta_t^2\}\right)$
     b. Let $\Phi_t \in \mathbb{R}^{m_t \times d}$ be an i.i.d. random projection matrix (e.g., subgaussian or sparse JL matrix)
     c. Let $\vartheta_t \leftarrow \mathrm{CSFO}(\mathbf{w}_t, \Phi_t)$ (from compressed stochastic gradient oracle)
     d. Let $\theta_t \leftarrow \Pi_{\Phi_t\mathcal{C}}(\Phi_t\mathbf{w}_t - \eta_t\vartheta_t)$
     e. Let $\mathbf{w}_{t+1} \leftarrow$ pick any element from the set $\mathcal{S}_t = \{\mathbf{w} \in \mathcal{C} : \Phi_t\mathbf{w} = \theta_t\}$ (e.g., by solving (2))
```

Note that for any $\Phi \in \mathbb{R}^{m\times d}$, $\Phi\mathcal{C}$ is a convex set, and since $\mathcal{C}$ is compact and closed, implies $\Phi\mathcal{C}$ is closed. This follows because if $\mathcal{C}$ is compact then the recess cone $R_\mathcal{C} = \{0\}$ (Definition 6), which implies that $\Phi\mathcal{C}$ is closed [Bertsekas, 2009]. Therefore, by properties of the projection operator $\Pi_{\Phi\mathcal{C}}$ is well-defined and satisfies the projection properties in (5) and (6). For common $\mathcal{C}$'s of interest here such as convex hulls (polytopes), the linear transformation (through Φ) is also a convex hull (polytope) in $\mathbb{R}^m$, so the standard techniques for projection onto these sets are applicable for projection onto $\Phi\mathcal{C}$.

[4]Given $\Phi\hat{\mathbf{g}} \in \mathbb{R}^m$ (with $m \ll d$) it is not possible to recover $\hat{\mathbf{g}}$ without further structural assumptions (such as sparsity) on $\hat{\mathbf{g}}$, which are generally not true.

In Algorithm CompSGD, $\mathbf{w}_{t+1}$ exists because $\theta_t \in \Phi_t\mathcal{C} = \{\Phi_t\mathbf{w} : \mathbf{w} \in \mathcal{C}\}$ so we can represent $\theta_t = \Phi\tilde{\mathbf{w}}$ with $\tilde{\mathbf{w}} \in \mathcal{C}$, therefore the set $\mathcal{S}_t = \{\mathbf{w} \in \mathcal{C} : \Phi_t\mathbf{w} = \theta_t\}$ has at least one entry $\tilde{\mathbf{w}}$. For our theoretical results, it suffices that we pick any $\mathbf{w}_{t+1} \in \mathcal{S}_t$. We now discuss an idea for constructing $\mathbf{w}_{t+1}$, based on estimating $\tilde{\mathbf{w}}$ which lies in $\mathcal{C}$, given $\theta_t = \Phi_t\tilde{\mathbf{w}}$. For any vector $\mathbf{w} \in \mathbb{R}^d$, the Minkowski functional of $\mathcal{C} \subseteq \mathbb{R}^d$ is the non-negative number $\|\mathbf{w}\|_\mathcal{C}$ defined by the rule: $\|\mathbf{w}\|_\mathcal{C} = \inf\{\tau \in \mathbb{R} : \mathbf{w} \in \tau\mathcal{C}\}$. Minkowski functional of a convex set is a convex function. In particular, when $\mathcal{C}$ is a symmetric convex body, then $\|\cdot\|_\mathcal{C}$ defines a norm (called Minkowski norm). Now consider the following optimization problem:

$$\mathrm{Inv}_\mathcal{C}(\theta_t, \Phi_t) \overset{\text{def}}{=} \mathrm{argmin}_{\mathbf{w}'\in\mathbb{R}^d} \|\mathbf{w}'\|_\mathcal{C} \text{ s.t. } \Phi_t\mathbf{w}' = \theta_t. \quad (2)$$

We now show that we can pick $\mathbf{w}_{t+1}$ by solving (2).

Lemma 2.1. *For any $\mathbf{w}_{t+1} \in \mathrm{Inv}_\mathcal{C}(\theta_t, \Phi_t)$, $\mathbf{w}_{t+1} \in \mathcal{S}_t$.*

For any convex set $\mathcal{C}$, the optimization problem in (2) is a convex program, and hence can be solved using convex programming solvers. In fact, if is a $\mathcal{C}$ polytope then this is a linear program. For example, if $\mathcal{C}$ is the popular ℓ_1-sparsity constraint, then (2) reduces to the well-studied *basis-pursuit* problem

$$\mathrm{argmin}_{\mathbf{w}'\in\mathbb{R}^d} \|\mathbf{w}'\|_1 \text{ subject to } \Phi_t\mathbf{w}' = \theta_t. \quad (3)$$

for which lots of efficient solution techniques are known [Hastie et al., 2015]. In our experiments, we noticed that using sparse random matrices and known tricks for solving (3) (see, e.g., [Lorenz et al., 2015]), the per-iteration cost of Algorithm CompSGD is almost identical to that of regular SGD. On the other hand, the reduction in gradient dimension could be quite substantial. Another point to note here is that in a distributed setup (as we discuss in Section 3.2), the selection of $\mathbf{w}_{t+1}$ will be performed on the server, which is assumed to be computationally powerful, therefore this step will not be a matter of concern.

2.1 CONVERGENCE ANALYSIS

We now analyze the convergence guarantees of Algorithm CompSGD. In all our results, the final expectation ($\mathbb{E}[\cdot]$) is over the stochastic gradient noise and other randomness in the SGD algorithm (as is standard), but not over the randomness introduced by the random projection. The following inequality will play an important role in keeping track of the algorithm's progress.

Lemma 2.2. *In Algorithm* CompSGD, *for any $t \in [T]$, $(1-\beta_t)\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 \leq \|(\mathbf{w}_t - \eta_t\hat{\mathbf{g}}_t) - \mathbf{w}\|^2$.*

The challenge in our analysis comes from the fact that we have access to only to lower-dimensional gradients, the updates happen in the compressed space, and the random projection introduces noise in the gradients. Our analyses will establish the values of η_t, β_t under which Algorithm COMPSGD has same (up to constant factors) dimension-free convergence guarantees as regular SGD. The analyses for the strongly convex and convex cases (with or without smoothness) are deferred to Appendix B.2 (Theorems B.7, B.8 and B.9). We get the following result.

Theorem 2.3. *1. [Strongly Convex, Smooth] Let F be a λ-strongly convex and μ-smooth function over a convex set $\mathcal{C}$, and $\mathbb{E}[\|\hat{\mathbf{g}}_t\|^2] \leq G^2$ for all $t \in [T]$ (where $\vartheta_t = \Phi_t \hat{\mathbf{g}}_t$). Then with $\eta_t = 2/(\lambda t)$ and $\beta_t = 1/t$, Algorithm* COMPSGD *satisfies: $\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^\star)] = O(\mu G^2/(\lambda^2 T))$.*

2. [Strongly Convex] Let F be a λ-strongly convex function over a convex set $\mathcal{C}$, and $\mathbb{E}[\|\hat{\mathbf{g}}_t\|^2] \leq G^2$ for all $t \in [T]$ (where $\vartheta_t = \Phi_t \hat{\mathbf{g}}_t$). Then with $\eta_t = 2/(\lambda t)$ and $\beta_t = 1/t$, Algorithm COMPSGD *satisfies: $\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^\star)] = O(G^2(1 + \log(T))/(\lambda T))$.*

3. [Convex] Let F be convex function over a convex set $\mathcal{C}$, and $\mathbb{E}[\|\hat{\mathbf{g}}_t\|^2] \leq G^2$ for all $t \in [T]$ (where $\vartheta_t = \Phi_t \hat{\mathbf{g}}_t$). Then with $\eta_t = \varphi/\sqrt{t}$ and $\beta_t = 1/t$, Algorithm COMPSGD *satisfies: $\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^\star)] = O(\left(\|\mathcal{C}\|^2/\varphi + \varphi G^2\right)(\log T/\sqrt{T}))$. In particular with $\varphi = \|\mathcal{C}\|/G$, $\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^\star)] = O(\|\mathcal{C}\| G \log T/\sqrt{T})$.*

Convergence Analysis for Nonconvex Functions. We now focus on the analysis for the nonconvex case. For nonconvex functions, in the constrained setting, several measures of (non)stationarity have been considered for projected gradient methods [Ghadimi et al., 2016, Mokhtari et al., 2018, Nouiehed et al., 2018]. We work with two popular stationarity notions here: a) α-FOSP (Definition 7)[3] and b) norm of the gradient mapping (deferred to Appendix B.1.2). Let $F^\star = \min_{\mathbf{w} \in \mathcal{C}} F(\mathbf{w})$. Missing details from this section are collected in Appendix B.1.

In this case, we assume a stronger stochastic approximation to the subgradient in that we use a minibatch version of Algorithm COMPSGD, where b is the minibatch size (formally described in Appendix B.1). Theorem 2.4 shows that after $\approx O(\alpha^{-2})$ iterations, even with compressed gradients, minibatch Algorithm COMPSGD converges to an α-FOSP (with high probability). The proof has two parts, first we show that when the difference between two consecutive iterates is small then we have an α-FOSP, and then we bound the number of iterations before which this iterate difference condition is achieved. We show that the algorithm fails with a low probability to produce an α-FOSP (in which case it outputs $\perp$). We assume the following.

Assumptions: $\mathbb{E}[\|\hat{\mathbf{g}}_t^{(i)}\|^2] \leq G^2$ and

$$\mathbb{E}[\|\nabla F(\mathbf{w}_t) - \hat{\mathbf{g}}_t^{(i)}\|^2] \leq \zeta^2, \ \ \forall i \in [b], t \in [T]. \quad (4)$$

Theorem 2.4. *Let F be μ-smooth and continuously differentiable function over a convex set $\mathcal{C}$. Let the assumptions in* (4) *hold. Let $\rho \in (0,1)$ and $\alpha > 0$. Consider the minibatch version of Algorithm* COMPSGD *with output of first $\mathbf{w}_\tau$ (if it exists) in $(\mathbf{w}_1, \ldots, \mathbf{w}_T)$ such that $\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\| \leq \alpha/(G + \mu\|\mathcal{C}\|)$, and $\perp$ (fail) otherwise. Set $\eta_t = \eta = 1/\mu$, $\beta_t = \beta = \min\{1/4, (\mu\alpha^2)/(64G\|\mathcal{C}\|(G + \mu\|\mathcal{C}\|)^2)\}$ for all $t \in [T]$, and batchsize $b = \Omega((1/\rho) \cdot \max\{1, (\|\mathcal{C}\|\zeta(G + \mu\|\mathcal{C}\|)^2/(\alpha^2\mu))^2\})$. Then, if $T = \Omega((F(\mathbf{w}_1) - F^\star)(G + \mu\|\mathcal{C}\|)^2/(\mu\alpha^2\rho))$, with probability at least $(1-\rho)^2$, this procedure outputs a $\mathbf{w}_\tau$ that is an α-FOSP for F.*

One may notice that in the above theorem β_t's are fixed across all iterations and scales as $\approx \alpha^2$. whereas in the convex settings we set $\beta_t = 1/t$. This distinction comes because the nature of guarantees differ in these two cases. In the convex case, as we get closer to the global optimum, we need a more precise estimate of the stochastic gradient (as compression adds noise), which is achieved by reducing β_t. In the nonconvex case, we can set β to a fixed value based on the required guarantee and there seems to be no advantage in adjusting it per iteration. We conjecture that for the convex cases, this dependence on β_t on t is unavoidable, if we want the corresponding convergence rates of Algorithm COMPSGD to match that of regular SGD.

The batchsize in Theorem 2.4 is carefully selected for our analysis to go through, and even with full stochastic gradients a similar batchsize would probably be needed, see for e.g., [Mokhtari et al., 2018] (note that the batchsize is independent of β_t and $\omega(\mathcal{C})$). In practice, small batchsizes suffices as observed in our experiments.

2.2 EXPERIMENTAL RESULTS

We now experimentally compare Algorithm COMPSGD to SGD with the goal of validating our theoretical findings. Missing details about the datasets and experimental setup are provided in Appendix B.3. There we also have additional experimental results under other constraint sets such as subspace and (positive) simplex (Figure 5).

Experimental Results on Convex Functions. In Figure 1, we compare the performance of SGD and Algorithm COMPSGD on sparse linear regression [Hastie et al., 2015] and logistic regression with ℓ_1-constraint [Lee et al., 2006].We use a sparse random projection matrix in our experiments. We use the linear program in (3) to pick w_{t+1}, and use heuristic stopping ideas from [Lorenz et al., 2015] on an ℓ_1-homotopy solver for efficiency. Based on the Gaussian width of the ℓ_1-ball, we set $m_t = \max\{d, t^2 \log(d)\}$

where d here is the dimensionality of the input. In Appendix B.3, we presents some results with other types of constraint sets.

With sparse linear regression, given a matrix $A \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ where $\mathbf{y} = A\mathbf{w}^\star$ with sparse $\mathbf{w}^\star$, we solve: $\min_{\mathbf{w}} \|\mathbf{y} - A\mathbf{w}\|^2$ subject to an ℓ_1-constraint on $\mathbf{w}$. In Figure 1a, we plot the objective value (error) $\|\mathbf{y} - A\mathbf{w}_t\|^2$ over SGD and COMPSGD iterations. We use synthetic random data for A with $n = 1000, d = 10000$. With logistic regression, we minimize the standard logistic regression objective but with an additional ℓ_1-constraint [Lee et al., 2006].

The performance COMPSGD is almost identical (sometimes marginally better) than SGD. Computationally too Algorithm COMPSGD matches that of SGD. For example, on the MNIST, Reuters, and IMDB datasets, SGD took an average of 0.58, 6.36, 7.04 seconds per epoch respectively, whereas Algorithm COMPSGD took an average of 0.75, 6.76, 7.49 seconds per epoch respectively. So, for example on the IMDB dataset, CompSGD is only about 6.3% slower than SGD, however, as Figure 3d (Appendix B.3) shows the dimensionality of utilized gradients is significantly smaller, by a factor of ≈ 19 over the entire run. So, overall, Algorithm COMPSGD matches (or marginally improves) SGD performance, achieves significant gradient compression, with a minor increase in the computational cost. Training accuracy plots are in Figure 3 (Appendix B.3). The variance across runs for Algorithm COMPSGD is also low (see, e.g., Figure 4, Appendix B.3). In theory, this stems because of the tight concentration results we have with these random projections [Oymak et al., 2018, Theorem 1.3].

Experimental Results on Nonconvex Functions. We use an MLP with three hidden fully-connected layers (input dimension $\times 50$, 50×50, and $50\times$ number of output classes) with ReLU activations, followed by softmax classifier. We optimize the network under an ℓ_1-constraint on weights. Figure 2 presents the test accuracy plots for this network. For COMPSGD, we set $m_t = d_i/10$ for each layer i, where d_i is the original number of parameters in layer i. The results show that convergence of SGD and COMPSGD are near identical, even though COMPSGD uses a *factor* of 10 lower-dimensional gradients. The training accuracy plots presented in Figure 6 (Appendix B.3) also exhibit a similar behavior.

3 APPLICATIONS

We now consider two different problems in which reducing the dimensionality of the gradient proves helpful.

3.1 DIFF. PRIV. ERM WITH NONCONVEXITY

We consider the standard Empirical Risk Minimization (ERM) framework. Given a dataset $D = (\mathbf{z}_1, \dots, \mathbf{z}_n)$ from a data universe $\mathcal{D}^n$, the goal in ERM is to: $\min_{\mathbf{w}\in\mathcal{C}} F(\mathbf{w}; D) := \frac{1}{n}\sum_{i=1}^n f(\mathbf{w}; \mathbf{z}_i)$, where $f : \mathbb{R}^d \to \mathbb{R}$ is the loss of model $\mathbf{w} \in \mathbb{R}^d$ for data record $\mathbf{z}_i$. Let us start with the definition of differential privacy. Let $\mathcal{D}$ represent some domain. We say two datasets $D \in \mathcal{D}^n$ and $D' \in \mathcal{D}^n$ with n elements each are neighbors if they differ in one entry.

Definition 2 ((ϵ, δ)-DP [Dwork et al., 2006]). *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all neighboring datasets D, D' and for all outcomes Γ in the output space of $\mathcal{A}$, we have $\Pr[\mathcal{A}(D) \in \Gamma] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(D') \in \Gamma] + \delta$, where the probability is taken over the randomness of the algorithm.*

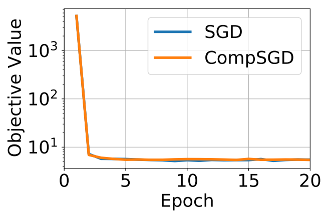
Missing details and formal statements from this section are provided in Appendix C.

One of the basic ideas for achieving DP for ERM problems is add noise to the gradients using the (DP-SGD) algorithm [Song et al., 2013, Bassily et al., 2014]. The DP-SGD iteration is of the form, $\theta_{t+1} \leftarrow \theta_t - \eta_t\left(\nabla F(\mathbf{w}; D) + \mathbf{e}_t\right)$, where $\mathbf{e}_t \sim \mathcal{N}(0, \sigma^2\mathbb{I}_d)$ is the calibrated noise to achieve (ϵ, δ)-differential privacy. The fact that $\nabla F(\mathbf{w}; D) \in \mathbb{R}^d$, means that the popular *Gaussian mechanism* (Theorem A.1) idea in DP requires $\mathbb{E}[\|\mathbf{e}_t\|^2] = \sigma^2 d$ that brings about the dependence on the dimension d in the utility analysis.

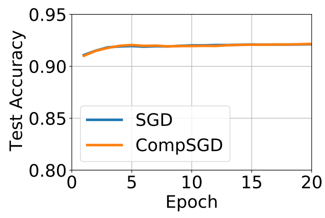
An approach to reduce the dependence on d arising from noise addition in DP-SGD iteration is to reduce the dimensionality of the gradient vector. For example, if we take $\Phi \in \mathbb{R}^{m\times d}$ with entries drawn i.i.d from $\mathcal{N}(0, 1/m)$, then $\Phi\nabla F(\mathbf{w}; D) \in \mathbb{R}^m$. Therefore, by using the Gaussian mechanism now, one could add noise as: $\Phi\nabla F(\mathbf{w}; D) + \mathbf{e}_t$ where $\mathbf{e}_t \sim \mathcal{N}(0, \sigma^2\mathbb{I}_m)$. This roughly changes the dependence on d to m in the convergence analysis. This idea is formalized in Algorithm PRIVSGD (Appendix C). Table 1 summarizes the improved sample size bounds. These are the first results in differentially private nonconvex optimization which provides meaningful guarantees when $n \gg \omega(\mathcal{C})$ rather than requiring $n \gg \sqrt{d}$. So for common $\mathcal{C}$'s, such as the ℓ_1-ball, this reduction in sample size would be exponential from roughly d to $\log d$.

3.2 REDUCING COMMUNICATION IN DIST. SGD

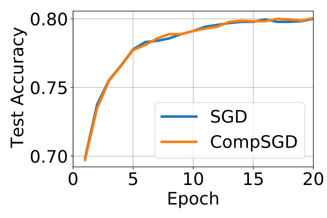
We consider the data-distributed model of distributed SGD. Let us assume that there are M clients, numbered $1, \dots, M$. Let $F(\mathbf{w}) : \mathbb{R}^d \to \mathbb{R}$. We investigate the following optimization problem: $\min_{\mathbf{w}\in\mathcal{C}} F(\mathbf{w}) := \frac{1}{M}\sum_{i=1}^M f_i(\mathbf{w})$, where each f_i resides at the ith client. As an illustration of the above setup, consider a machine learning problem, with data $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ partitioned on the clients, with P_i the set of indexes of datapoints on client i, then with $f_i(\mathbf{w}) = (M/n)\sum_{j\in P_i} f(\mathbf{w}; \mathbf{z}_j)$, we get $F(\mathbf{w}) = (1/n)\sum_{i=1}^n f(\mathbf{w}; \mathbf{z}_i)$. Missing details are provided in Appendix D.



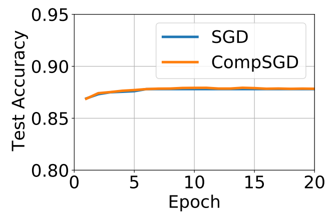
(a) Synthetic



(b) MNIST

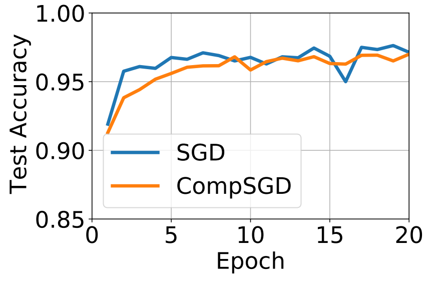


(c) Reuters

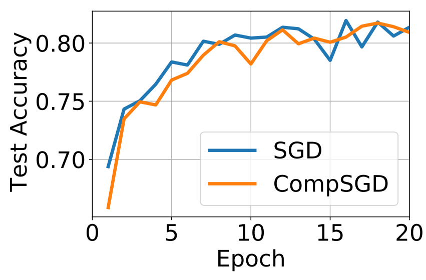


(d) IMDB

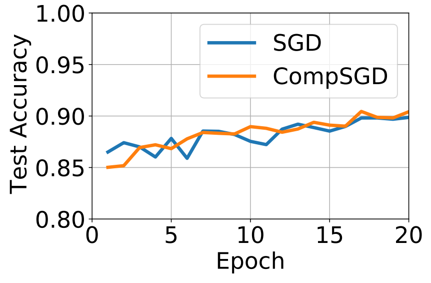
Figure 1: (a) Performance comparison for sparse linear regression on a synthetic dataset. (b), (c), and (d) Performance comparison for logistic regression with ℓ_1-constraint on the MNIST dataset, Reuters dataset, and IMDB datasets, respectively. We use a constant learning rate at 0.1 based on its good performance for SGD and use minibatch size of 32.



(a) MNIST



(b) Reuters



(c) IMDB

Figure 2: Performance comparison of training a MLP network with SGD vs. CompSGD on different datasets. Again, we use a constant learning rate at 0.1 based on its good performance for SGD and use minibatch size of 32.

Guarantee	DP-SGD Sample Size Reqd.	Algorithm PrivSGD Sample Size Reqd.
α-FOSP	$\Omega\left(\sqrt{d}\cdot\frac{\|\mathcal{C}\|(L+\mu\|\mathcal{C}\|)^2L\sqrt{T\log(1/\delta)}}{\sqrt{\rho}\alpha^2\mu\epsilon}\right)$	$\Omega\left(\min\left\{\frac{\omega(\mathcal{C})}{\beta},\sqrt{d}\right\}\cdot\frac{\|\mathcal{C}\|(L+\mu\|\mathcal{C}\|)^2L\sqrt{T\log(1/\delta)}}{\sqrt{\rho}\alpha^2\mu\epsilon}\right)$

Table 1: Comparison of sample size (n) needed for achieving α-FOSP for a nonconvex μ-smooth function under (ϵ,δ)-DP. Assume $\|\nabla f(\mathbf{w};\cdot)\|$ is uniformly bounded by L for all $\mathbf{w}\in\mathcal{C}$. The settings of β and the upper bound on T are from Corollary C.2. Additional results are presented in Table 4, Appendix C.

Assumptions on F and Guarantee	SGD (Total comm. cost)	SGD with Contraction C (Total comm. cost)	Algorithm CompDistSGD (Total comm. cost)
Convex $\mathbb{E}[F(\mathbf{w}_T)-F(\mathbf{w}^\star)]\le\alpha$	$\tilde{O}\left(\frac{G^2\|\mathcal{C}\|^2}{\alpha^2}d\right)$ Shamir and Zhang [2013]	$\tilde{O}\left(\frac{\chi_{C_d}G^2\|\mathcal{C}\|^2}{\alpha^2}\gamma_{C_d}\right)$	$\tilde{O}\left(\frac{\chi_{C_d}G^2\|\mathcal{C}\|^2}{\alpha^2}\min\{\gamma_{C_d},\gamma_{C_r}\}\right)$ (From Corollary D.2, Part 3)
μ-smooth and diff. nonconvex α-FOSP	$O(\kappa_1 Td)$ Mokhtari et al. [2018]	$O(\kappa T\gamma_{C_d})$	$O(\kappa T\min\{\gamma_{C_d},\gamma_{C_r}\})$ (From Corollary D.2, Part 4)

Table 2: Comparison of the total communication costs of distributed synchronous SGD, distributed synchronous SGD with contraction operator C, and our proposed Algorithm CompDistSGD. For convex case, we set $\kappa=1$ and $r=(\omega(\mathcal{C})\chi_{C_d}G^2\|\mathcal{C}\|^2/\alpha^2)^2$. The $\tilde{O}(\cdot)$ notation hides some logarithmic terms. The settings of κ,β,T for the nonconvex case is from Corollary D.2, Part 4 and $r=\omega(\mathcal{C})^2/\beta^2$. Here, κ_1 is the value of κ obtained by setting $\chi_{C_d}=1$. Additional results are presented in Table 5, Appendix D.

In each iteration t of synchronous SGD, the server randomly picks a set R_t of $\kappa\ge 1$ clients and sends them the current model parameter $\mathbf{w}_t$. Each of these selected client i computes $\hat{\mathbf{g}}_t^{(i)}$ an independent stochastic (sub)gradient of f_i at $\mathbf{w}_t$, and communicates $\hat{\mathbf{g}}_t^{(i)}$ back to the server. The central server then aggregates these gradients and applies the update $\mathbf{w}_{t+1}\leftarrow\mathbf{w}_t-\frac{\eta_t}{\kappa}\sum_{i\in R_t}\hat{\mathbf{g}}_t^{(i)}$. Naively for the above protocol, the resulting per round communication is roughly $\kappa\cdot(32d)$ bits (assuming 32-bit floating point numbers). Our results (Theorems B.7, B.8, B.9, 2.4, and B.5) already show that, by transmitting lower-dimensional gradients, we can reduce communication costs, while preserving the convergence guarantees to within constant factor of the regular distributed SGD. In this section, we build on these results and show that in fact one can use any unbiased gradient compression scheme on top of our idea of utilizing lower-dimensional gradients.

For a formal analysis, we define a *contraction operator* to capture a general class of existing unbiased gradient encoding schemes. Following [Stich et al., 2018], we define a contraction operator as a (possibly randomized) function C_a defined as mapping from $\mathbb{R}^a \to \mathbb{R}^a$ for $a \in \mathbb{N}$ that satisfies these following common assumptions: (i) $\forall \mathbf{v} \in \mathbb{R}^a, \mathbb{E}[C_a(\mathbf{v})] = \mathbf{v}$ (unbiasedness) and (ii) $\mathbb{E}[\|C_a(\mathbf{v}) - \mathbf{v}\|^2] \leq \chi_{C_a} \|\mathbf{v}\|^2$ (variance bound). Let γ_{C_a} denote the bits needed to communicate $C_a(\mathbf{v})$ from a client to server. This general notion captures many common unbiased quantization techniques such as *stochastic rounding* [Alistarh et al., 2017, Wen et al., 2017], vector quantization techniques such as *vqSGD* [Gandikota et al., 2019], and unbiased sparsification techniques such as *random sparsification* [Stich et al., 2018]. As an example, the quantization technique of [Alistarh et al., 2017] satisfies $\chi_{C_a} = 1$ and $\gamma_{C_a} \approx 2.8a + 32$ for all $a \in \mathbb{N}$.

Our procedure is presented in Algorithm COMPDISTSGD (Appendix D). The overall idea is simple, take a random projection of a gradient and then apply the contraction operator C_{m_t}. More formally, client i at iteration t will transmit $C_{m_t}(\Phi_t \hat{\mathbf{g}}_t^{(i)})$. The total communication cost in iteration t equals $\kappa \cdot \gamma_{C_{m_t}}$ bits. This saves at least a factor $\gamma_{C_d}/\gamma_{C_{m_t}}$ over just applying the contraction operator over the true gradients in each iteration t, which could be significant when $m_t \ll d$.[5] For example, if C is the quantization technique of [Alistarh et al., 2017], then $\gamma_{C_d}/\gamma_{C_{m_t}} \approx d/m_t$ where $m_t \approx \omega(\mathcal{C})^2/\beta_t^2$, and for sets $\mathcal{C}$ such as the unit l_1-ball, $\omega(\mathcal{C})^2 = O(\log d)$.

The cost of communicating Φ_t is not significant as it can be achieved using various techniques such as one-to-all broadcasting. In practice, Φ_t will be generated by a pseudorandom generator initialized by some seed, so by just communicating the seed we can regenerate Φ_t at each device. In Table 2, we summarize the total communication cost (summed over all rounds) for a convex and nonconvex setting considered. Additional results and formal statements are provided in Appendix D.

In Figure 7, we provide experiments supporting our theoretical results. For illustration, we used the simple stochastic gradient quantization technique of [Alistarh et al., 2017] (described in Appendix D.1) for the contraction operator and assume just one client. Otherwise, we use the same experimental setup as described in Section 2.2. Again, the main takeaway is that the performance of COMPDISTSGD matches that of SGD under the same contraction operator. The performance drop in both these schemes (compared to Figure 2 where we do not have any quantization) comes due to the applied quantization.

[5] γ_{C_a} is a non-decreasing function in a, i.e., cost of communicating a longer vector can't be less than one for a shorter vector.

4 EXTENSIONS AND CONCLUSIONS

Frank-Wolfe with Low-Dim. Gradients. The Frank-Wolfe (FW) optimization algorithm requires access to a linear optimization oracle over $\mathcal{C}$. In Appendix E, we show how one could recover the standard convergence guarantees of Frank-Wolfe algorithm (for both convex and nonconvex functions) with only access to compressed gradients provided through the CSFO oracle (see Algorithm COMPFW). The main difference, compared to a traditional Frank-Wolfe algorithm, is how we invoke the linear optimization oracle. A traditional (stochastic) FW update computes a stochastic approximation to the gradient at the current iterate $\mathbf{w}_t$ and invokes the LOO oracle with it. This gives the element in $\mathcal{C}$ that correlates the most with the steepest descent (the negative stochastic gradient). We use a similar idea but instead solve a linear minimization problem with the compressed gradients over the set $\Phi_t\mathcal{C}$. We then utilize the lifting idea from (2) to compute a direction to take the step.

In Theorem E.2 (Appendix E), we establish that for convex functions the convergence guarantees of Algorithm COMPFW matches the known results with stochastic Frank-Wolfe algorithm [Hazan and Luo, 2016, Theorem 3]. In Theorem E.1 (Appendix E), we show that for smooth nonconvex functions Algorithm COMPFW, after $O(\alpha^{-2})$ iterations, converges to an α-FOSP (with high probability).

Concluding Remarks. We introduce the setting of SGD with compressed gradients, a fundamental question that studies how much bits of gradient information are truly needed for SGD to continue providing its guarantees, and also captures practical applications in private nonconvex ERM and distributed SGD. This new setup requires a rethinking of the SGD algorithm, and a subsequent careful analyses of the SGD guarantees. We also show that these ideas extend beyond SGD to the conditional gradient method. While we focused on getting bounds which hold in expectation it is possible that under assumptions on the tail of the noise distribution (such as [Harvey et al., 2019]) one could obtain high probability bounds. A natural question that arises from this work is whether the connection between constraint set structure and gradient compression that we observe here is inherent for *any first-order* optimization scheme. Extending the presented techniques to compress higher-order derivatives is an interesting open research direction.

References

Jayadev Acharya, Chris De Sa, Dylan Foster, and Karthik Sridharan. Distributed learning with sublinear communication. In *International Conference on Machine Learning*, pages 40–50, 2019.

Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*, pages 7564–7575, 2018.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pages 5973–5983, 2018.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*. IEEE, 2014.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-sgd: Distributed sgd with quantization, sparsification, and local computations. *arXiv preprint arXiv:1906.02367 (Also in NeruIPS 2019)*, 2019.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SIGNSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 559–568, 2018.

Dimitri P Bertsekas. *Convex optimization theory*. Athena Scientific Belmont, 2009.

Jean Bourgain, Dirksen Sjoerd, and Jelani Nelson. Toward a unified theory of sparse dimensionality reduction in euclidean space. In *Proceedings of the 47th ACM Symposium on Theory of Computing*. Association for Computing Machinery, 2015.

Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of Hogwild!-style algorithms. In *Advances in neural information processing systems*, pages 2674–2682, 2015.

Sjoerd Dirksen. Dimensionality reduction with subgaussian matrices: a unified theory. *arXiv preprint arXiv:1402.3973*, 2014.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Venkata Gandikota, Raj Kumar Maity, and Arya Mazumdar. vqsgd: Vector quantized stochastic gradient descent. *arXiv preprint arXiv:1911.07971*, 2019.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Minibatch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

Yehoram Gordon. *On Milman's inequality and random subspaces which escape through a mesh in $\mathbb{R}^n$*. Springer, 1988.

Nicholas JA Harvey, Christopher Liaw, Yaniv Plan, and Sikander Randhawa. Tight analyses for non-smooth stochastic gradient descent. In *ICML*, 2019.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015.

Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271, 2016.

Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. Communication-efficient distributed sgd with sketching. In *Advances in Neural Information Processing Systems*, pages 13144–13154, 2019.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 427–435, 2013.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261, 2019.

Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson. Distributed learning with compressed gradients. *arXiv preprint arXiv:1806.06573*, 2018.

Jakub Konecnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y Ng. Efficient ℓ_1-regularized logistic regression. In *AAAI*, volume 6, pages 401–408, 2006.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Dirk A Lorenz, Marc E Pfetsch, and Andreas M Tillmann. Solving basis pursuit: Heuristic optimality check and solver comparison. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):1–29, 2015.

Konstantin Mishchenko, Eduard Gorbunov, Martin Takác, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

Aryan Mokhtari, Asuman Ozdaglar, and Ali Jadbabaie. Escaping saddle points in constrained optimization. In *Advances in Neural Information Processing Systems*, pages 3629–3639, 2018.

Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3(4):5, 1998.

Maher Nouiehed, Jason D Lee, and Meisam Razaviyayn. Convergence to second-order stationarity for constrained non-convex optimization. *arXiv preprint arXiv:1810.02024*, 2018.

Samet Oymak, Benjamin Recht, and Mahdi Soltanolkotabi. Isometric sketching of any set via the restricted isometry property. *Information and Inference: A Journal of the IMA*, 7(4):707–726, 2018.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 245–248. IEEE, 2013.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458, 2018.

Sebastian Urban Stich. Local sgd converges fast and communicates little. In *ICLR 2019 ICLR 2019 International Conference on Learning Representations*, 2019.

Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, pages 9850–9861, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, pages 5321–5329, 2018.

Jiaqi Zhang, Kai Zheng, Wenlong Mou, and Liwei Wang. Efficient private erm for smooth objectives. *arXiv preprint arXiv:1703.09947*, 2017.