2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Gradient Descent with Low-Rank Objective Functions

Romain Cosson, Ali Jadbabaie, Anuran Makur, Amirhossein Reisizadeh, Devavrat Shah

***Abstract*— Several recent empirical studies demonstrate that important machine learning tasks, e.g., training deep neural networks, exhibit low-rank structure, where the loss function varies significantly in only a few directions of the input space. In this paper, we leverage such low-rank structure to reduce the high computational cost of canonical gradient-based methods such as gradient descent (GD). Our proposed *Low-Rank Gradient Descent* (LRGD) algorithm finds an ϵ-minimizer of a p-dimensional function by first identifying $r \leq p$ significant directions, and then estimating the true p-dimensional gradient at every iteration by computing directional derivatives only along those r directions. We establish that the "directional oracle complexity" of LRGD for strongly convex objective functions is $\mathcal{O}(r\log(1/\epsilon) + rp)$. Therefore, when $r \ll p$, LRGD provides significant improvement over the known complexity of $\mathcal{O}(p\log(1/\epsilon))$ of GD in the strongly convex setting. Furthermore, using real and synthetic data, we empirically find that LRGD provides significant gains over GD when the data has low-rank structure, and in the absence of such structure, LRGD does not degrade performance compared to GD.**

I. Introduction

First order optimization methods such as Gradient Descent (GD) and its variants have become the cornerstone of training modern machine learning models due to their simplicity and feasibility of implementation at scale. The optimization problem of interest can be viewed as $\min_{\theta\in\mathbb{R}^p} f(\theta)$, with objective function $f : \mathbb{R}^p \to \mathbb{R}$ which captures dependence on data and underlying model constraints (e.g., regularization). The computational model of interest measures the number of gradient computations, also known as *oracle complexity*. Subsequently, within the optimization meets machine learning community, there has been a significant interest in reducing oracle complexity of first order methods, cf. [1], [2]. Formally, we shall measure *oracle complexity* as the number of evaluations of *directional derivatives* of f to achieve a certain accuracy level $\epsilon > 0$. Under this definition, the vanilla GD algorithm requires an oracle complexity of $\mathcal{O}(p\log(1/\epsilon))$ to find an ϵ-minimizer of a strongly convex objective f [3]. This complexity grows linearly with parameter dimension p (note that each iteration of GD costs p directional derivatives as it requires a full gradient computation), which can be prohibitive, e.g., in deep neural networks. However, it has recently been observed that many empirical risk minimization problems have objective functions with *low-rank structure* in their gradients [4]–[7]. In what follows, we will refer to them as *low-rank functions*. They naturally appear in many scenarios, a few of which are listed here.

Motivation 1: Low-rank Hessians. Low-rank structures have been found in large-scale deep learning scenarios irrespective of the architecture, training methods, and tasks [4]–[6], [8], where the Hessians exhibit a sharp decay in their eigenvalues. For instance, in classification tasks with k classes, during the course of training a deep neural network model, the gradient lives in the subspace spanned by the k eigenvectors of the Hessian with largest eigenvalues [4].

Motivation 2: Ridge functions. Ridge functions are generally defined as functions that only vary on a given low-dimensional subspace of the ambient space [9]. There are many standard examples of ridge function losses in machine learning, e.g., least-squares regression, logistic regression, one hidden layer neural networks, etc. [10], [11]. The natural model for Principal Component Regression (PCR) is another such example [12].

Given the above motivations, we will consider a real-valued function as low-rank if its gradients live *close* to a low-dimensional subspace. Such low-rank structure has been exploited to theoretically improve the running times of federated optimization algorithms recently in [7]. Yet, canonical GD methods do not exploit this additional structure. Hence, the goal of this work is to address the following question: *Can we leverage low-rank structure to design optimization algorithms with running times that have better dependence on the dimension?*

Main contributions. To address this question, as the main contribution of this work, we propose *Low-Rank Gradient Descent* (LRGD) methods to leverage such low-rank structure in the objective to reduce oracle complexity. Specifically, we restrict the gradient of the function to the low-rank subspace defined by some significant directions–the *active subspace*–and perform descent iterations. More precisely, LRGD identifies a fairly accurate proxy for such an active subspace. Then, in each descent iteration, it approximates the true gradient vector by computing only r (dimension of active subspace) directional derivatives of the objective along the active subspace. Note that each iteration of LRGD costs only r directional derivative computations. Intuitively, this is far less than canonical methods such as GD, which has iteration cost growing linearly with p, as noted earlier. In particular, for sufficiently low-rank objectives, LRGD is able to reduce GD's directional oracle complexity $\mathcal{O}(p\log(1/\epsilon))$ to $\mathcal{O}(r\log(1/\epsilon) + rp)$.

Related work. Low-rank structures have been exploited extensively in various disciplines. For example, in machine

The author ordering is alphabetical.

R. Cosson is with Inria, Paris, France (email: cosson@inria.edu).
A. Jadbabaie, A. Reisizadeh, and D. Shah are with the Laboratory for Information & Decision Systems, MIT, Cambridge, MA 02139, USA (emails: {jadbabai,amirr,devavrat}@mit.edu).
A. Makur is with the Department of Computer Science and the School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (email: amakur@purdue.edu).

learning, matrix estimation methods typically rely on low-rank assumptions [13]. In classical statistics, projection pursuit methods rely on low-rank approximations of functions [9], [14]. In a similar vein, in scientific computing, approximation methods have been developed to identify influential input directions of a function for uncertainty quantification [15].

In contrast to these settings, we seek to exploit low-rank structure in optimization algorithms. Recently, the authors of [16] developed a local polynomial interpolation based GD algorithm for empirical risk minimization that learns gradients at every iteration using smoothness of loss functions in data. The work in [7] extended these developments to a federated learning context (cf. [17], [18] and the references therein), where low-rank matrix estimation ideas were used to exploit such smoothness of loss functions.

Our work falls within the broader effort of speeding up first order optimization algorithms, which has been widely studied in the literature. In this literature, the running time is measured using first order oracle complexity (i.e., the number of full gradient evaluations until convergence) [3]. The first order oracle complexity of GD for, e.g., strongly convex functions, is analyzed in the standard text [3]. Similar analyses for stochastic versions of GD that are popular in large-scale empirical risk minimization problems, such as (mini-batch) stochastic GD [19], [20]. There are several standard approaches to theoretically or practically improving the running times of these basic algorithms, e.g., acceleration [3], variance reduction [21]–[23], and adaptive learning rates [24], [25]. More related to our problem, random coordinate descent-type methods, such as stochastic subspace descent [26], have also been studied. In addition, various other results pertaining to GD with inexact oracles (see [27] and the references therein), fundamental lower bounds on oracle complexity [28], etc. have been established in the literature.

II. Formal setup and low-rank structure

A. Preliminaries

We consider a real-valued differentiable function $f : \Theta \to \mathbb{R}$, where $\Theta = \mathbb{R}^p$ for some positive integer p. We assume that f is L-smooth and μ-strodgly convex.

Assumption 1 (L-smoothness)**.** *The function f is L-smooth for a given parameter $L > 0$ if for any $\theta, \theta' \in \Theta$, $f(\theta') \leq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{L}{2}\|\theta' - \theta\|^2$.*

Assumption 2 (Strong convexity)**.** *The function f is μ-strongly convex for a given parameter $\mu > 0$ if for any $\theta, \theta' \in \Theta$, $f(\theta') \geq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{\mu}{2}\|\theta' - \theta\|^2$.*

The goal we pursue in this setting is to find an ϵ-minimizer for f, which is defined as $\theta \in \Theta$ satisfying

$$f(\theta) - f^* \leq \epsilon, \tag{1}$$

where $\epsilon > 0$ is a predefined approximation accuracy and $f^* = \inf_{\theta \in \Theta} f(\theta) \in \mathbb{R}$. In this paper, we focus on the computational complexity of gradient descent methods, which we concretely measure using the following computational model.

Computational model. To characterize the running time of a GD-type algorithm for solving (1), we employ a variant of the well-established notion of *oracle complexity* [3]. In particular, we tailor this notion to count the number of oracle calls to *directional* derivatives of the objective, where the directional derivative of f along a unit-norm vector $u \in \mathbb{R}^p$ is a scalar defined as

$$\partial_u f(\theta) \coloneqq \lim_{t \to 0} \frac{f(\theta + tu) - f(\theta)}{t} = \langle \nabla f(\theta), u \rangle.$$

This computation model—which slightly differs from most of the literature on first order methods that typically assumes the existence of a full-gradient oracle $\nabla f(\theta)$—will allow us to illustrate the utility of a low-rank optimization method.

Definition 1 (Oracle complexity)**.** *Given an algorithm* ALG*, for a predefined accuracy $\epsilon > 0$ and function class $\mathcal{F}$, we denote by $\mathcal{C}_{\text{ALG}}$ the maximum number of oracle calls to directional derivatives required by* ALG *to reach an ϵ-approximate solution as in* (1) *for any function in $\mathcal{F}$.*

Note that computing a full-length gradient vector of dimension p requires p calls to the directional gradient oracle. As a consequence, for the class of smooth and strongly convex functions, the oracle complexity of vanilla GD to find an ϵ-minimizer of f is $\mathcal{C}_{\text{GD}} = \mathcal{O}(p \log(1/\epsilon))$ [3].

In this paper, we improve the (directional) oracle complexity of gradient-based algorithms like GD by proposing a computationally efficient algorithm called *Low-Rank Gradient Descent* (LRGD). The main idea of LRGD is to leverage the potential low-rank structure of the objective in order to reduce the oracle complexity during iterations. To do so, LRGD first identifies a low-rank subspace H, known as the *active subspace*, that (approximately) contains the gradients of the objective f. Let $\{u_1, \cdots, u_r\}$ denote an orthonormal basis for H. In each iteration with current model parameter θ, LRGD computes r directional derivatives of f, i.e., $\partial f_{u_1}(\theta), \cdots, \partial f_{u_r}(\theta)$, and uses $\hat{\nabla} f(\theta) = \sum_{i \in [r]} \partial_{u_i} f(\theta) u_i$ as an approximation to the true full-gradient $\nabla f(\theta)$ and updates the model parameter as $\theta \leftarrow \theta - \alpha \hat{\nabla} f(\theta)$ with an appropriate stepsize α.

B. Approximately low-rank functions

Intuitively, a low-rank function maintains its gradient vectors in a low-dimensional subspace. This is inspired by the motivating examples described in Section I. For instance, any ridge function on $\mathbb{R}^p$ satisfies the condition that its gradient vectors *exactly* live in a subspace of dimension $r \leq p$. Our notion, however, is broader and includes ridge functions as a special case and allows low-dimensional subspaces to *approximately* satisfy the aforementioned condition.

Definition 2 (Approximate rank of function)**.** *The function f is (η, ϵ)-approximately rank-r, if there exist $\eta \in [0, 1)$, $\epsilon \geq 0$, and a subspace $H \subseteq \mathbb{R}^p$ of dimension $r \leq p$, such that for any $\theta \in \Theta$,*

$$\|\nabla f(\theta) - \Pi_H(\nabla f(\theta))\| \leq \eta \|\nabla f(\theta)\| + \epsilon, \tag{2}$$

where Π_H denotes the orthogonal projection operator onto H.

Note that this definition has two error terms represented by two constants η (multiplicative) and ϵ (additive). Both of these constants are required for the generality of the definition of approximate low-rankness. In particular, the role of the additive term is emphasized in the following Proposition 1.

Proposition 1 (Non-singular Hessian and approximate low-rankness)**.** *If f attains its minimum at $\theta^* \in \Theta$ where $\nabla^2 f(\theta^*)$ is invertible and f is approximately low-rank with parameters (η, ϵ) where $\eta < 1$, then necessarily $\epsilon > 0$.*

The proof in Section V-B is a direct application of the inverse function theorem to ∇f at θ^*. This result shows that in most settings, the approximate rank requires an affine approximation error (i.e., $\epsilon > 0$ is necessary).

Exactly low-rank functions. As noted earlier, the notion of approximately low-rank functions in Definition 2 subsumes the well-known class of ridge functions, which have $\eta = \epsilon = 0$. We refer to functions that satisfy condition (2) with $\eta = \epsilon = 0$ as *exactly* low-rank functions. It is worth noting that such functions have already been widely studied in the literature under different equivalent guises. However, the class of such functions is fairly restricted. Next, we define the rank of a function for this particular case of exactly low-rank functions.

Definition 3 (Rank of function)**.** *The function f has rank r if the minimal subspace $H \subseteq \mathbb{R}^p$ satisfying $\nabla f(\theta) \in H$ for all $\theta \in \Theta$ has dimension r.*

III. Algorithms and Theoretical Guarantees

A. LRGD algorithm

The proposed algorithm, LRGD, is an iterative gradient-based method. It starts by identifying a subspace H of rank $r \leq p$ that is a good candidate to match our definition of approximately low-rank function for the objective f, i.e., Definition 2. More precisely, we pick r random models $\theta^1, \cdots, \theta^r$ and construct the matrix $G = [g_1/\|g_1\|, \cdots, g_r/\|g_r\|]$, where we let $g_j := \nabla f(\theta^j)$ for $j \in [r]$. Then, assuming that G is full-rank, the active subspace H will be the space induced by the r dominant left singular vectors of G. In other words, if we denote $G = U\Sigma V^\top$ as the singular value decomposition (SVD) of G, we pick $H = \text{span}(U) = \text{span}(u_1, \cdots, u_r)$. Having set the active subspace H, LRGD updates the model parameters θ_t in each iteration t by $\theta_{t+1} = \theta_t - \alpha\hat{\nabla} f(\theta_t)$ for an appropriate stepsize α. Note that $\hat{\nabla} f(\theta_t)$ here denotes the projection of the true gradient $\nabla f(\theta_t)$ onto the active subspace H, which LRGD uses as a low-rank proxy to $\nabla f(\theta_t)$. Computing each approximate gradient $\hat{\nabla} f$ requires only r (and not p) calls to the directional gradient oracle as we have $\hat{\nabla} f(\theta_t) = \sum_{j=1}^r \partial_{u_j} f(\theta_t) u_j$ (see Algorithm 1).

B. Oracle complexity analysis for strongly convex setting

Next, we characterize the oracle complexity of the LRGD method for strongly convex objectives starting with the more general case of approximately low-rank functions.

Theorem 1 (Oracle complexity in approximately low-rank and strongly convex case)**.** *Let the objective function f be L-smooth and μ-strongly convex with condition number $\kappa := L/\mu$, i.e., Assumptions 1 and 2 hold. We also assume that f is $(\eta, \sqrt{\mu\epsilon/5})$-approximately rank-$r$ according to Definition 2, and the following condition holds:*

$$\eta\left(1 + \frac{2r}{\sigma_r}\right) + \frac{2r}{\sigma_r \epsilon'}\sqrt{\frac{\mu\epsilon}{5}} \leq \frac{1}{\sqrt{10}},$$

where σ_r is the smallest singular value of the matrix $G = [g_1/\|g_1\|, \cdots, g_r/\|g_r\|]$ with $g_j := \nabla f(\theta^j)$ and $\|g_j\| > \epsilon'$ for all $j \in [r]$. Then, for stepsize $\alpha = \frac{1}{8L}$, the proposed LRGD method in Algorithm 1 reaches an ϵ-minimizer of f with oracle complexity

$$\mathcal{C}_{\text{LRGD}} = 16\kappa r \log(2\Delta_0/\epsilon) + pr,$$

where $\Delta_0 = f(\theta_0) - f^*$ is the initial suboptimality.

Algorithm 1: Low-Rank Gradient Descent (LRGD)

```
1  Require: rank $r \leq p$, stepsize $\alpha$
2  pick $r$ points $\theta^1, \cdots, \theta^r$ and evaluate gradients
     $g_j = \nabla f(\theta^j)$ for $j \in [r]$
3  set $G = [g_1/\|g_1\|, \cdots, g_r/\|g_r\|]$
4  compute SVD for $G$: $G = U\Sigma V^\top$ with
     $U = [u_1, \cdots, u_r]$ (Gram-Schmidt or QR is also possible)
5  set $H = \text{span}(u_1, \cdots, u_r)$
6  initialize $\theta_0$
7  for $t = 0, 1, 2, \cdots$ do
8      compute gradient approximation
         $\hat{\nabla} f(\theta_t) = \Pi_H(\nabla f(\theta_t)) = \sum_{j=1}^r \partial_{u_j} f(\theta_t) u_j$
9      update $\theta_{t+1} = \theta_t - \alpha\hat{\nabla} f(\theta_t)$
10 end
```

We defer the proof to Section V-A. Theorem 1 shows that in the strongly convex setting, for approximately low-rank functions with sufficiently small parameters, LRGD is able to reduce the oracle complexity of GD, which is $\mathcal{C}_{\text{GD}} = \mathcal{O}(\kappa p \log(1/\epsilon))$, to $\mathcal{C}_{\text{LRGD}} = \mathcal{O}(\kappa r \log(1/\epsilon) + pr)$. This gain is particularly significant as the term depending on the accuracy ϵ does not scale with the dimension p, and rather scales with the rank r which can be much smaller than p.

In general, strongly convex functions cannot be exactly low-rank. However, an exactly low-rank function may be strongly convex when restricted to its active subspace. The next result characterizes the oracle complexity for LRGD in such scenarios.

Proposition 2 (Oracle complexity in exactly low-rank and strongly convex case)**.** *Let the objective function f be L-smooth (Assumption 1) and exactly rank-r according to Definition 3, where $\nabla f(\theta) \in H$, $\forall\theta$. Moreover, assume that f restricted to H is μ-strongly convex (Assumption 1) with condition number $\kappa := L/\mu$. Then, the proposed LRGD method in Algorithm 1 with stepsize $\alpha = 1/L$ reaches an ϵ-minimizer of f with oracle complexity*

$$\mathcal{C}_{\text{LRGD}} = \kappa r \log(\Delta_0/\epsilon) + pr,$$

where $\Delta_0 = f(\theta_0) - f^*$ is the initial suboptimality.

Algorithm 2: Iterated Low-Rank Gradient Descent

```
1  Require: $r \geq 0,\ \epsilon > 0,\ \alpha \in \left(0, \frac{1}{L}\right],\ \mu > 0,\ \theta_0 \in \mathbb{R}^p$
2  initialize $\theta \leftarrow \theta_0$
3  while $\|\nabla f(\theta)\| > \sqrt{2\mu\epsilon}$ do
4      $\theta_1, ..., \theta_r = \mathtt{GD}(\theta, r)$          ($r$ iterations of GD)
5      $u_1, ..., u_r = \text{Gram-Schmidt}(\nabla f(\theta_1), ..., \nabla f(\theta_r))$
6      $\hat{\nabla} f(\theta) \leftarrow \sum_{k=1}^{r} \partial_{u_k} f(\theta) u_k$
7      while $\|\hat{\nabla} f(\theta)\| \geq \sqrt{2\mu\epsilon}$ do
8          $\theta \leftarrow \theta - \alpha \hat{\nabla} f(\theta)$
9          $\hat{\nabla} f(\theta) \leftarrow \sum_{k=1}^{r} \partial_{u_k} f(\theta) u_k$
10     end
11 end
12 return $\theta$
```

C. *Variant of* `LRGD`

Algorithm 1 (`LRGD`) can be extended to broader settings, especially in order to make it less sensitive to low-rank assumptions on the function f, since such assumptions are often unknown a priori before the optimization task is defined.

In particular, Algorithm 2 provides a variant of `LRGD` where the active subspace is updated as soon as the low-rank descent converges. Note that this guarantees the termination of the optimization algorithm on any function, not just on functions satisfying the assumptions of the theoretical results. This algorithm is particularly adapted to situations where the function is *locally* approximately rank r as the active subspace is updated through a local sampling (that is also used to accelerate the descent with r iterations of `GD`). This algorithm still takes the parameter r as an input. It is used for the empirical experiments in Section IV.

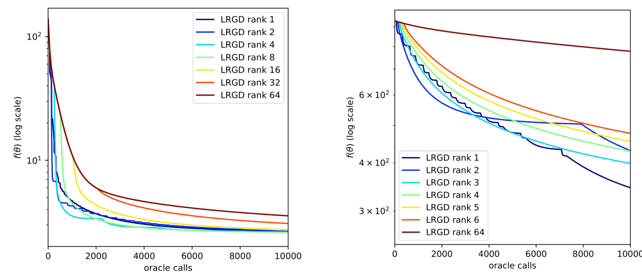
IV. Numerical simulations

In this section, we demonstrate the utility of the `LRGD` method—specifically the iterated variant described in Algorithm 2—through numerical simulations.

In the following experiments, the goal is to minimize the empirical risk $f(\theta) = \sum_{i \in [n]} \ell(x_i, y_i; \theta)$ for loss function ℓ over n data samples $\{(x_i, y_i) : i = 1, \dots, n\}$. Here, $X = [x_1^\top; \cdots; x_n^\top] \in \mathbb{R}^{n \times p}$ and $Y = [y_1 \cdots y_n]^\top \in \mathbb{R}^n$ denote the matrices of features and labels, respectively. We also consider loss functions $\ell(x_i, y_i; \theta) = \|x_i^\top \theta - y_i\|^2$ and $\ell(x_i, y_i; \theta) = \mathbf{1}(y_i = 1)\log(1/(1 + e^{-x_i^\top \theta})) + \mathbf{1}(y_i = 0)\log(e^{-x_i^\top \theta}/(1 + e^{-x_i^\top \theta}))$ for our regression and logistic regression problems, respectively.

Digits recognition dataset. The UCI handwritten digits dataset [29] contains $n = 2000$ images of $p = 8 \times 8$ pixels of handwritten digits along with their labels. We consider two different binary classification tasks on this dataset. The first task is a binary classification of digits 0 and 1. In the second task, we set our goal to detect loops in the digits, that is, classifying digits to either of the two groups $\{0, 6, 8, 9\}$ or $\{1, 2, 3, 5, 7\}$.

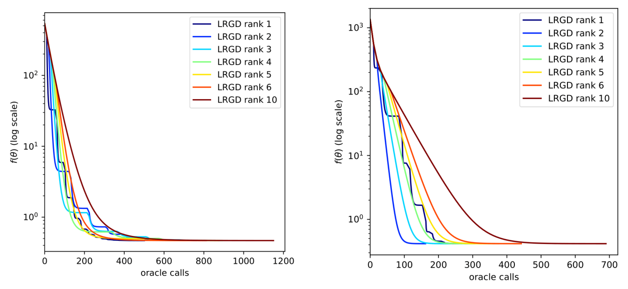
Figures 1a and 1b demonstrate the decay of the training loss versus the number of (directional) oracle calls for `LRGD` with several values of the rank. For this dataset, the optimization objective is that of logistic regression. For instance, Figure 1a demonstrates that the same level of accuracy is attained by `LRGD` with 10 times fewer oracle calls than `GD`. Furthermore, the better performance of `LRGD` with medium ranks reveals inherent low-rank structures in the dataset.



(a) Binary 0/1 classification $(n, p) = (200, 64), \epsilon = 10^2$ (LR).

(b) Binary loop detection $(n, p) = (1800, 64), \epsilon = 10^6$.

Fig. 1: Training error for digits recognition



(a) Full-rank data $(n, p) = (10, 50), \epsilon = 0.1$.

(b) Planted rank $r = 2$ $(n, p) = (10, 50), \epsilon = 0.1$.

Fig. 2: Training error for synthetic data.

Synthetic data with full rank. Here, the features are generated independently from a standard normal distribution $x_i \sim \mathcal{N}(0, I_p)$ for $i \in [n]$. The response variable y_i is a noisy linear measurement of a ground-truth model $\theta^* = [1 \cdots 1]^\top$, i.e., $y_i = \langle x_i, \theta^* \rangle + n_i$ with i.i.d. noises $n_i \sim \mathcal{N}(0, 0.1)$. For the logistic regression experiments, the quantity y_i is turned into a binary value by randomly sampling from a Bernoulli distribution with parameter $1/(1 + e^{-y_i})$.

Synthetic data with low rank. This dataset differs from the previous solely in the way the features $x_i, i \in [n]$ are sampled. To guarantee an approximate low-rank structure, we first sample matrices $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{p \times r}$, and set the feature matrix as $X = UV^\top$. The entries of U and V are i.i.d. and sampled from a standard normal distribution, i.e., $U_{i,j} \sim \mathcal{N}(0, 1)$ and $V_{i,j} \sim \mathcal{N}(0, 1)$.

Figure 2 illustrates the training loss for the full-rank and low-rank datasets described above, respectively. Interestingly, `LRGD`'s performance is superior to `GD`'s performance even in the full-rank setting. Moreover, `LRGD` is particularly effective on the low-rank datasets as well. The convergence curves of `LRGD` have several successive drops in the function value–much like a "waterfall". This is particularly noticeable when the rank parameter of the algorithm is small. Moreover, Table I quantifies the total oracle complexity of iterated `LRGD` for different ranks r and the two linear and logistic regression

	rank 1	rank 2	rank 4	rank 6	GD (rank p)
LS, full rank	353	518	614	596	880
LS, low rank (2)	564	362	616	862	10000
LR, low rank (4)	3667	1050	940	1266	3580

TABLE I: Number of oracle calls for least square (LS) and logistic regression (LR).

α	rank 1	rank 2	rank 4	rank 6	GD (rank p)
1/2L	564	362	616	862	10000
1/4L	640	438	768	1090	10000
1/8L	792	590	1072	1546	14800

TABLE II: Least square regression, synthetic data with planted rank $r = 2$, $(n, p) = (50, 10)$, $\epsilon = 0.1$.

problems, where the case $r = p$ coincides with GD. For the same setting and problems described above, Table II demonstrates the total oracle complexity for different values of the stepsize α. Table I further confirms that picking the proper rank particularly for the planted low-rank datasets yields the lowest complexity as seen in columns with rank close to the planted rank.

V. Proofs of Oracle Complexity

A. *Proof of Theorem 1*

First, we characterize the gradient inexactness induced by the gradient approximation step of LRGD.

Lemma 1 (Gradient approximation)**.** *Assume that the f is (η, ϵ)-approximately rank-r according to Definition 2 and the following condition holds*

$$\frac{2r}{\sigma_r}\left(\eta + \frac{\epsilon}{\epsilon'}\right) < \frac{1}{\sqrt{10}}, \tag{3}$$

where σ_r is the smallest singular value of the matrix $G = [g_1/\|g_1\|, \cdots, g_r/\|g_r\|]$ with $g_j := \nabla f(\theta^j)$ and $\|g_j\| > \epsilon'$ for all $j \in [r]$. Then, the gradient approximation error of LRGD is bounded as $\|\hat{\nabla} f(\theta) - \nabla f(\theta)\| \leq \tilde{\eta}\|\nabla f(\theta)\| + \epsilon$, for $\tilde{\eta} := \eta + 2r(\eta + \frac{\epsilon}{\epsilon'})/\sigma_r$.

Proof. Let $G = U\Sigma V^\top$ and $G_H = U_H \Sigma_H V_H^\top$ respectively denote SVDs for matrices G and G_H, where

$$G = \left[\frac{g_1}{\|g_1\|}, \dots, \frac{g_r}{\|g_r\|}\right], \quad G_H = \left[\frac{\Pi_H(g_1)}{\|g_1\|}, \dots, \frac{\Pi_H(g_r)}{\|g_r\|}\right].$$

Since f is (η, ϵ)-approximately rank-r per Definition 2, we can bound the operator norm of $G - G_H$ as follows

$$\|G - G_H\|_{\text{op}} \leq \sqrt{r}\|G - G_H\|_{\text{F}}$$
$$\leq r \max_{j \in [r]} \left\| \frac{g_j}{\|g_j\|} - \frac{\Pi_H(g_j)}{\|g_j\|} \right\| \leq r\left(\eta + \frac{\epsilon}{\epsilon'}\right), \tag{4}$$

where in the last inequality we used the assumption that $\|g_j\| > \epsilon'$ for all $j \in [r]$. The condition stated in the lemma ensures that G is full-rank (since $\sigma_r > 0$). Next, we show that projecting columns of G on H does not degenerate its rank, that is, G_H is full-rank as well. To this end, we employ Weyl's inequality [30, Corollary 7.3.5(a)] to write that

$$\sigma_r(G_H) \geq \sigma_r - \sigma_1(G - G_H) \geq \sigma_r - \|G - G_H\|_{\text{op}}$$
$$\overset{(a)}{\geq} \sigma_r - r\left(\eta + \frac{\epsilon}{\epsilon'}\right) \overset{(b)}{>} 0.$$

In above, $\sigma_r(G_H)$ and $\sigma_1(G - G_H)$ respectively denote the smallest and largest singular values of G_H and $G - G_H$, (a) follows from the bound in (4), and (b) is an immediate result of the condition (3). Since H is a subspace of rank r and columns of G_H are all in H, we have that $H = \text{span}(U_H)$. Next, we employ Wedin's $\sin\Theta$ theorem [31] to conclude

$$\text{Sin}\Theta(\text{span}(U), \text{span}(U_H)) = \|\Pi_U - \Pi_H\|_{\text{op}}$$
$$\leq \frac{2}{\sigma_r}\|G - G^*\|_{\text{op}} \leq \frac{2r}{\sigma_r}\left(\eta + \frac{\epsilon}{\epsilon'}\right).$$

This yields that $\forall \theta \in \mathbb{R}^p$ we can uniformly bound the LRGD gradient approximation error as follows

$$\|\hat{\nabla} f(\theta) - \nabla f(\theta)\| = \|\Pi_U(\nabla f(\theta)) - \nabla f(\theta)\|$$
$$\leq \|\Pi_H \nabla f(\theta) - \nabla f(\theta)\| + \|\Pi_H \nabla f(\theta) - \Pi_U \nabla f(\theta)\|$$
$$\leq \tilde{\eta}\|\nabla f(\theta)\| + \epsilon,$$

for $\tilde{\eta} := \eta + 2r(\eta + \frac{\epsilon}{\epsilon'})/\sigma_r$. □

Next, we characterize the convergence of inexact gradient descent and use LRGD's gradient approximation error bound derived in Lemma 1 to complete the proof.

Let us get back to the setting of Theorem 1. From smoothness of f in Assumption 1 and the update rule of LRGD, i.e. $\theta_{t+1} = \theta_t - \alpha\hat{\nabla} f(\theta_t)$, we can write

$$f(\theta_{t+1}) = f(\theta_t - \alpha\hat{\nabla} f(\theta_t))$$
$$\leq f(\theta_t) - \alpha\langle \nabla f(\theta_t), \hat{\nabla} f(\theta_t)\rangle + \frac{L}{2}\alpha^2\|\hat{\nabla} f(\theta_t)\|^2$$
$$= f(\theta_t) - \alpha\|\nabla f(\theta_t)\|^2 - \alpha\langle \nabla f(\theta_t), e_t\rangle + \frac{L}{2}\alpha^2\|\hat{\nabla} f(\theta_t)\|^2, \tag{5}$$

where $e_t := \hat{\nabla} f(\theta_t) - \nabla f(\theta_t)$ denotes the gradient approximation error at iteration t. Next, we employ the gradient approximation error bound derived in Lemma 1, that is $\|e_t\| \leq \tilde{\eta}\|\nabla f(\theta_t)\| + \tilde{\epsilon}$, where we denote $\tilde{\epsilon} := \sqrt{\mu\epsilon/5}$ and $\tilde{\eta} := \eta + 2r(\eta + \frac{\tilde{\epsilon}}{\epsilon'})/\sigma_r$ for notation simplicity. Together with simple algebra from (5), we have that

$$f(\theta_{t+1}) \leq f(\theta_t) - \alpha\|\nabla f(\theta_t)\|^2 + \frac{\alpha}{2}\|\nabla f(\theta_t)\|^2$$
$$+ \frac{\alpha}{2}\|e_t\|^2 + \alpha^2 L\|\nabla f(\theta_t)\|^2 + \alpha^2 L\|e_t\|^2,$$
$$\leq f(\theta_t) + \alpha(1 + 2\alpha L)\tilde{\epsilon}^2$$
$$- \frac{\alpha}{2}\left(1 - 2\alpha L - 2\tilde{\eta}^2(1 + 2\alpha L)\right)\|\nabla f(\theta_t)\|^2. \tag{6}$$

Next, we use the gradient dominant property of strongly convex loss functions, that is $\|\nabla f(\theta)\|^2 \geq 2\mu(f(\theta) - f^*)$, $\forall\theta$, which together with (6) implies that

$$f(\theta_{t+1}) - f^* \leq (1 - \tilde{\alpha}\mu)(f(\theta_t) - f^*) + \alpha(1 + 2\alpha L)\tilde{\epsilon}^2,$$

with notation $\tilde{\alpha} := \alpha\left(1 - 2\alpha L - 2\tilde{\eta}^2(1 + 2\alpha L)\right)$. Now we pick the stepsize $\alpha = \frac{1}{8L}$ and use the condition $\tilde{\eta} \leq 1/\sqrt{10}$ to conclude that

$$f(\theta_{t+1}) - f^* \leq \left(1 - \frac{1}{16\kappa}\right)(f(\theta_t) - f^*) + \frac{5}{32L}\tilde{\epsilon}^2. \tag{7}$$

The contraction bound in (7) implies that after T iterations of LRGD the final suboptimality is bounded as below

$$f(\theta_T) - f^* \leq \exp\left(-\frac{T}{16\kappa}\right)(f(\theta_0) - f^*) + \frac{\epsilon}{2}.$$

Finally, in order to reach an ϵ-minimizer of f, it suffices to satisfy the following condition $\exp(-\frac{T}{16\kappa})(f(\theta_0) - f^*) \leq \epsilon/2$, i.e., LRGD runs for $T = 16\kappa \log(2\Delta_0/\epsilon)$ iterations. Thus, the total oracle complexity of LRGD is

$$\mathcal{C}_{\texttt{LRGD}} = Tr + pr = 16\kappa r \log(2\Delta_0/\epsilon) + pr. \quad \square$$

B. Proof of Proposition 1

We write Taylor's expansion around θ^*,

$$\nabla f(\theta) = \nabla^2 f(\theta^*)(\theta - \theta^*) + o(\|\theta - \theta^*\|),$$

where $\nabla^2 f(\theta^*)$ is invertible, and the little-o notation applies entry-wise. For any given subspace H, we can consider $u = \nabla^2 f(\theta^*)^{-1} v$ where $v \in H^\perp$ and note that

$$\nabla f(\theta^* + tu) = tv + o(t), \quad \Pi_H(\nabla f(\theta^* + tu)) = o(t).$$

As a consequence, as $t \to 0$,

$$\frac{\|\nabla f(\theta^* + tu) - \Pi_H(\nabla f(\theta^* + tu))\|}{\|\nabla f(\theta^* + tu)\|} \to 1.$$

Assuming that $\eta < 1$ and $\epsilon = 0$ would imply that the above limit is strictly less than one, which gives a contradiction. $\square$

C. Proof Sketch of Proposition 2

We omit a detailed proof due to space constraints, and note that the proof follows from the known convergence analysis of GD for smooth and strongly convex functions (see, e.g., [3]). More precisely, one can show that the LRGD updates on f are identical to GD updates on the function projected to the r-dimensional subspace H, which is strongly convex. It takes $\kappa \log(\Delta_0/\epsilon)$ iterations of such updates to reach an ϵ-optimal solution, each iteration costing r directional derivative computations. Together with the cost of the initial r full gradient computations to construct G, the total oracle complexity is $\kappa r \log(\Delta_0/\epsilon) + pr$.

References

[1] S. Bubeck, *Convex Optimization: Algorithms and Complexity*, ser. Foundations and Trends in Machine Learning. Hanover, MA, USA: now Publishers Inc., 2015, vol. 8, no. 2-4.

[2] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, May 2018.

[3] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, ser. Applied Optimization. New York, NY, USA: Springer, 2004, vol. 87.

[4] G. Gur-Ari, D. A. Roberts, and E. Dyer, "Gradient descent happens in a tiny subspace," December 2018, arXiv:1812.04754 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1812.04754

[5] L. Sagun, U. Evci, V. U. Güney, Y. Dauphin, and L. Bottou, "Empirical analysis of the Hessian of over-parametrized neural networks," in *Proceedings of the Sixth International Conference on Learning Representations (ICLR) Workshop*, Vancouver, BC, Canada, April 30-May 3 2018, pp. 1–14.

[6] Y. Wu, X. Zhu, C. Wu, A. Wang, and R. Ge, "Dissecting Hessian: Understanding common structure of Hessian in neural networks," June 2021, arXiv:2010.04261v5 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2010.04261

[7] A. Jadbabaie, A. Makur, and D. Shah, "Federated optimization of smooth loss functions," January 2022, arXiv:2201.01954 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2201.01954

[8] S. P. Singh, G. Bachmann, and T. Hofmann, "Analytic insights into structure and rank of neural network hessian maps," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[9] B. F. Logan and L. A. Shepp, "Optimal reconstruction of a function from its projections," *Duke mathematical journal*, vol. 42, no. 4, pp. 645–659, 1975.

[10] V. Ismailov, "Notes on ridge functions and neural networks," *arXiv preprint arXiv:2005.14125*, 2020.

[11] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[12] I. T. Jolliffe, "A note on the use of principal components in regression," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 31, no. 3, pp. 300–303, 1982.

[13] E. J. Candès and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, June 2010.

[14] D. L. Donoho and I. M. Johnstone, "Projection-based approximation and a duality with kernel methods," *The Annals of Statistics*, pp. 58–106, 1989.

[15] P. G. Constantine, E. Dow, and Q. Wang, "Active subspace methods in theory and practice: applications to kriging surfaces," *SIAM Journal on Scientific Computing*, vol. 36, no. 4, pp. A1500–A1524, 2014.

[16] A. Jadbabaie, A. Makur, and D. Shah, "Gradient-based empirical risk minimization using local polynomial regression," November 2020, arXiv:2011.02522 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2011.02522

[17] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA, April 20-22 2017, pp. 1273–1282.

[18] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, Palermo, Italy, August 26-28 2020, pp. 2021–2031.

[19] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, January 2009.

[20] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. 6, pp. 165–202, January 2012.

[21] M. Schmidt, N. L. Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming, Series A*, vol. 162, p. 83–112, March 2017.

[22] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, pp. 567–599, February 2013.

[23] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proceedings of the Advances in Neural Information Processing Systems 26*, Lake Tahoe, NV, USA, December 5-10 2013, pp. 315–323.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[25] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 7-9 2015, pp. 1–13.

[26] D. Kozak, S. Becker, A. Doostan, and L. Tenorio, "Stochastic subspace descent," *arXiv preprint arXiv:1904.01145*, 2019.

[27] M. P. Friedlander and M. Schmidt, "Hybrid deterministic-stochastic methods for data fitting," *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. A1380–A1405, 2012.

[28] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, "Lower bounds for finding stationary points II: first-order methods," *Mathematical Programming, Series A*, pp. 1–41, September 2019.

[29] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[30] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge University Press, 2013.

[31] Y. Chen, Y. Chi, J. Fan, and C. Ma, "Spectral methods for data science: A statistical perspective," *arXiv preprint arXiv:2012.08496*, 2020.