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

Adaptive Low-Rank Gradient Descent

Ali Jadbabaie, Anuran Makur, Amirhossein Reisizadeh

***Abstract*— *Low-rank structures* have been observed in several recent empirical studies in many machine and deep learning problems, where the loss function demonstrates significant variation only in a lower dimensional subspace. While traditional gradient-based optimization algorithms are computationally costly for high-dimensional parameter spaces, such low-rank structures provide an opportunity to mitigate this cost. In this paper, we aim to leverage low-rank structures to alleviate the computational cost of first-order methods and study *Adaptive Low-Rank Gradient Descent* (AdaLRGD). The main idea of this method is to begin the optimization procedure in a very small subspace and gradually and adaptively augment it by including more directions. We show that for smooth and strongly convex objectives and any target accuracy ϵ, AdaLRGD's complexity is $\mathcal{O}(r\ln(r/\epsilon))$ for some rank r no more than dimension d. This significantly improves upon gradient descent's complexity of $\mathcal{O}(d\ln(1/\epsilon))$ when $r \ll d$. We also propose a practical implementation of AdaLRGD and demonstrate its ability to leverage existing low-rank structures in data.**

I. Introduction

Several recent empirical studies have demonstrated *low-rank structures* in common empirical risk minimization problems including deep learning tasks [1]–[7]. Considering image classification tasks as an example, [1] studies such low-rank structures in different neural network models trained with gradient-based methods over MNIST dataset with ten classes. As Figure 1 (left) demonstrates, after only a few training iterations of a fully connected neural network, more than 90% of the gradient norm is encapsulated in a low-rank subspace of the input space well before the convergence (right). In this plot, the y-axis measures f_{top} defined as the relative norm of the gradient vector at each iteration projected onto a low-rank subspace corresponding to the top ten leading eigenvectors of the Hessian matrix. Therefore, the gradient vectors of dimension $d = 784$ live (almost entirely) in a low-rank subspace of dimension $r = 10$. This example suggests an opportunity to leverage such low-rank structures in order to speed up the training of different machine learning and deep learning problems.

Following this new line, our aim in this work is to leverage low-rank structures in the objective function and mitigate the computational cost of gradient-based optimization methods.
Problem setup. We consider the problem of minimizing an objective function $f : \mathbb{R}^d \to \mathbb{R}$, that is, $\min_{\theta\in\mathbb{R}^d} f(\theta)$. Our goal is to find an ϵ-optimal solution θ such that $f(\theta) - f^* \le \epsilon$,

The author ordering is alphabetical.

A. Jadbabaie and A. Reisizadeh are with the Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (emails: {jadbabai,amirr}@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).

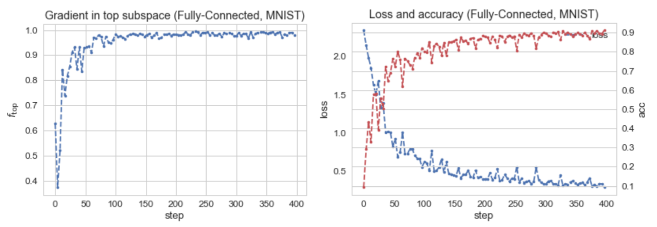

Fig. 1: Fraction of the gradient in the top subspace (left), training loss and test accuracy (right) (Figures from [1].)

where $f^* = \min_{\theta\in\mathbb{R}^d} f(\theta)$ is the global optimum function value, and the target accuracy $\epsilon > 0$ is given. Throughout the paper, we assume that f is differentiable, L-smooth, and μ-strongly convex with condition number $\kappa := L/\mu$.

In this paper, we focus on *first-order* methods to solve the minimization problem described above, and particularly aim to reduce the *directional* oracle complexity, i.e., number of oracle calls, of such methods. One call to the directional oracle of f along the unit vector direction $\mathbf{u}$ returns the following real-valued derivative

$$\partial_{\mathbf{u}} f(\theta) := \lim_{t\to 0} \frac{f(\theta + t\mathbf{u}) - f(\theta)}{t} = \langle \nabla f(\theta), \mathbf{u}\rangle.$$

This oracle complexity metric is different from canonical notions which measure the number of gradient computations, each consisting of d directional derivative computations [8], [9]. Indeed, we study an optimization algorithm that takes such finer complexity into account.

Most recently, the above mentioned notion of directional oracle complexity was employed in [10], where the authors proposed the *Low-Rank Gradient Descent* (LRGD) algorithm. LRGD leverages the low-rank structure in certain functions to reduce the oracle complexity of canonical gradient descent (GD) type methods. More precisely, it is shown in [10] that LRGD is able to reduce the oracle complexity of GD, provided that the objective function f is "approximately low-rank" (i.e., the gradient vectors ∇f live in a rank-r subspace with a small deviation). Roughly speaking, the LRGD method first determines such an r-dimensional subspace, and then performs first-order updates solely along these r directions.

However, LRGD is prone to certain drawbacks. Firstly, since all iteration updates of LRGD take place in a smaller and fixed r-dimensional subspace, there are always residual errors (from the orthogonal subspace) which LRGD is unable to handle. Secondly, the approximately low-rank condition required for convergence of LRGD is fairly restrictive.

To mitigate these challenges, we study *Adaptive Low-Rank Gradient Descent* (AdaLRGD) introduced in [10] in this paper. The main idea of AdaLRGD is to begin with only a few significant directions of the parameter space and gradually expand the size of this "active subspace". More precisely, AdaLRGD is an iterative method and consists of several stages. During each stage, iterates are updated only along directions of the active subspace. After a certain termination condition is met, the dimension of the active subspace is doubled in the next stage, and iterates continue to be updated in the new and larger subspace. This procedure continues until all d directions of the parameter space are activated or the target accuracy is reached.

The AdaLRGD method does not require the objective function to satisfy the restrictive low-rank conditions required by LRGD. However, AdaLRGD still demonstrates significant benefit compared to GD for certain functions. In particular, the total oracle complexity of AdaLRGD for smooth and strongly convex functions with condition number κ scales as $\mathcal{O}(\kappa r \ln(r/\epsilon))$, where the dimension $r \le d$ is determined by the function's particular characteristics. This significantly improves the oracle complexity of GD, which is $\mathcal{O}(\kappa d \ln(1/\epsilon))$ [9], when $r \ll d$.

Related work. As alluded in Figure 1 and its corresponding discussion above, the Hessian matrix and its spectrum are central to low-rank structures in certain applications such as deep learning [1]. Indeed, several works have extensively studied the spectrum of the Hessian matrix in deep learning applications and devised optimization algorithms that incorporate "low-rankness" of the Hessian [11]–[18]. Other applications of such low-rank structures in statistics and machine learning include projection pursuit methods with ridge functions [19], [20], principal component regression [21], low-rank matrix completion [22].

Recently, a new line of research has attempted to utilize such low-rank (and related smoothness) structure to improve the computational complexity of gradient-based optimization methods [1], [10], [23], [24]. This direction of exploiting low-rank structure for optimization has been complementary to the broader effort of improving the running times of gradient-based methods [25]–[28].

II. AdaLRGD Algorithm

In this section, we describe the AdaLRGD algorithm [10]. Consider the minimization problem $\min_{\theta \in \mathbb{R}^d} f(\theta)$ and fix an orthonormal matrix $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_d] \in \mathbb{R}^{d \times d}$. The AdaLRGD algorithm consists of a number of stages indexed by $s = 0, \cdots, S$, each including T_s iterations. We denote by $\theta_{s,t}$ the tth iteration in stage s. Starting with stage $s = 0$, θ_0 is initialized and a *small* $d_0 \ll d$ is picked as the dimension of the active subspace for this stage. For simplicity, we assume that $d_0 = 1$ and d is a power of 2. Iterates of AdaLRGD are updated as $\theta_{0,t+1} = \theta_{0,t} - \eta \sum_{i=1}^{d_0} \partial_{\mathbf{u}_i} f(\theta_{0,t}) \mathbf{u}_i$, in this stage with stepsize η, initialization $\theta_{0,0} := \theta_0$ and $0 \le t \le T_0 - 1$. In other words, the iterates are updated only along the directions $\mathbf{u}_1, \cdots, \mathbf{u}_{d_0}$ costing d_0 directional gradient computation as opposed to $d \gg d_0$ computations required by each iteration of GD. After T_0 iterations of stage $s = 0$, the next one $s = 1$ is initialized with the last iterate of the previous stage, i.e. $\theta_{1,0} := \theta_{0,T_0}$. Moreover, the dimension of the active subspace is *doubled*, that is, $d_1 = 2d_0 = 2$, and the iterates are updated along directions $\mathbf{u}_1, \cdots, \mathbf{u}_{d_1}$ for T_1 iterations. This procedure continues till all total d directions are included, or a target accuracy is reached. The procedure is described in Algorithm 1. To fully characterize AdaLRGD, we next elaborate on the choice of its parameters $\mathbf{U}, S, T_s$.

Algorithm 1 Adaptive Low-Rank Gradient Descent (AdaLRGD)

```
Require: initialization $\theta_0 \sim \rho$ and rank $d_0$, stepsize $\eta$, # of
  stages $S$, # of itr./stage $T_s$, orthonormal $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_d]$
  for $s = 0, \cdots, S$ do
    Initialize $\theta_{s,0} = \theta_{s-1,T_{s-1}}$        ▷ $\theta_{0,0} = \theta_0$ for stage $s = 0$
    Double the rank of the active subspace $d_s = 2d_{s-1}$
    Pick active subspace $\mathbf{u}_1, \cdots, \mathbf{u}_{d_s}$
    for $t = 0, \cdots, T_s - 1$ do
      Update $\theta_{s,t+1} = \theta_{s,t} - \eta \sum_{i=1}^{d_s} \partial_{\mathbf{u}_i} f(\theta_{s,t}) \mathbf{u}_i$
    end for
  end for
```

A. Determining the parameters of AdaLRGD

In the following, we discuss how the parameters of AdaLRGD are determined.

1) Number of iterations per stage T_s: Let us first set a few notations which is central to our following discussions. For fixed matrix $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_d]$ and $\theta \in \mathbb{R}^d$, let $\mathcal{S}(\theta; \mathbf{U}_r)$ denote the r-dimensional affine subspace passing through θ and spanned by $\mathbf{U}_r := [\mathbf{u}_1 \cdots \mathbf{u}_r]$, that is, $\mathcal{S}(\theta; \mathbf{U}_r) = \theta + \mathrm{span}(\mathbf{U}_r)$. We denote by Δ_r, the expected suboptimality of f when restricted to subspace $\mathcal{S}(\theta; \mathbf{U}_r)$, that is,

$$\Delta_r := \mathbb{E}_{\theta \sim \rho}[F_r^*(\theta)] - f^*, \ F_r^*(\theta) := \min_{\theta' \in \mathcal{S}(\theta; \mathbf{U}_r)} f(\theta'). \quad (1)$$

From this definition, optimizing the objective f in stage $s = 0$ with initialization θ_0 and restricted to rank-1 subspace $\mathrm{span}(\mathbf{u}_1)$ yields the residual Δ_1 in expectation. Therefore, there is no point in optimizing the restricted function. Rather, this stage is terminated after T_0 iteration where the average suboptimality is at most $2\Delta_1$, that is, $\mathbb{E}[f(\theta_{0,T_0})] - f^* \le 2\Delta_1$. We will later show that running AdaLRGD in stage $s = 0$ for $T_0 = \kappa \ln(\Delta_0/\Delta_1)$ guarantees such suboptimality where $\kappa = L/\mu$ denotes the condition number and $\eta = 1/L$ is the stepsize. The same logic is used in the following stages to set the number of iteration per stage. Particularly, for each stage $1 \le s \le S - 1$, we show that running AdaLRGD for $T_s = \kappa \ln(2\Delta_{d_{s-1}}/\Delta_{d_s})$ iterations guarantees that the final iterate of stage s is within $2\Delta_{d_s}$ of the optimal value f^*. We defer the details to proof of Theorem 1.

2) Number of stages $S + 1$: The main parameters that determine the total number of AdaLRGD's stages are the residuals Δ_r and the target accuracy ϵ. In Theorem 1 we show that if the target accuracy ϵ satisfies the condition $\Delta_r \le \epsilon/2 < \Delta_{r/2}$ for some rank r, then AdaLRGD reaches an ϵ-optimal solution in stage at most $S = \log(r)$ with $T_S = \kappa \ln(4\Delta_{d_{S-1}}/\epsilon)$ iterations (See Figure 2). Note that in the worst case, this condition is satisfied with $r = d$ as $\Delta_d = 0$.

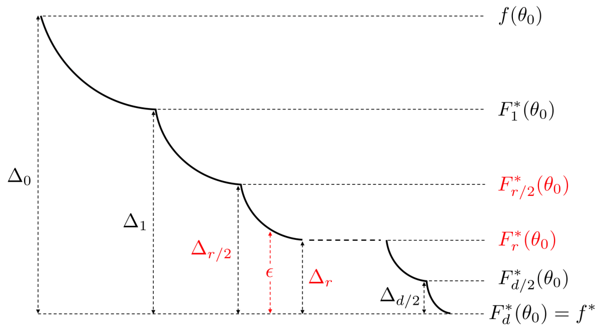

Fig. 2: Illustration of AdaLRGD.

3) Matrix $\mathbf{U}$*:* As we will show in Theorem 1, AdaLRGD can be run with any choice of matrix $\mathbf{U}$. However, as we elaborated above, both the target accuracy ϵ and residuals Δ_r determine the number of stages, which are functions of the choice of $\mathbf{U}$. A poor choice of $\mathbf{U}$ would yield a large number of stages S and hence large oracle complexity which defies the purpose of AdaLRGD. Therefore, in order for AdaLRGD to save in directional gradient computation, it is critical that for small accuracy ϵ, the stopping criteria $\Delta_r \leq \epsilon/2 < \Delta_{r/2}$ is satisfied with as small rank r. Though it remains a challenge to find such $\mathbf{U}$ in general, we will show in Section III-A for special cases that *mean gradient outer-product* (MeGO) matrix may indeed be a proper candidate [29]. For a fixed distribution ρ over $\mathbb{R}^d$, the MeGO matrix $\mathbf{C}$ is defined as

$$\mathbf{C} := \mathbb{E}_{\theta\sim\rho}[\nabla f(\theta)\nabla f(\theta)^\top]. \tag{2}$$

In Section IV, we discuss a practical implementation of AdaLRGD by empirically approximating the MeGO matrix in (2). There, we also propose a heuristic criteria to determine the number of iterations per stage, i.e. T_s.

III. Main Results

In this section, we characterize the oracle complexity of the proposed AdaLRGD method and compare it with GD.

Theorem 1. *Consider the optimization problem* $\min_\theta f(\theta)$ *for L-smooth and* μ*-strongly convex objective* f *with condition number* $\kappa := L/\mu$*. Assume that the target accuracy* ϵ *satisfies the condition* $\Delta_r \leq \epsilon/2 < \Delta_{r/2}$ *for some rank* $1 \leq r \leq d$ *with residuals defined in* (1)*. Then, the oracle complexity of* AdaLRGD *in Algorithm 1 with stepsize* $\eta = 1/L$ *is at most* $\mathcal{C}_{\texttt{AdaLRGD}} \leq \kappa r \ln(2r\Delta_0/\epsilon)$.

Note that the oracle complexity of GD for the same problem as described in Theorem 1 is $\mathcal{C}_{\texttt{GD}} = \kappa d \ln(\Delta_0/\epsilon)$ which scales linearly with dimension d. This is due to the fact that GD computes d-dimensional gradient vectors in *every* iteration while AdaLRGD avoids such costly computation by starting from a small subspace and gradually expanding its dimension. Moreover, there always exists rank r that satisfies the condition $\Delta_r \leq \epsilon/2 < \Delta_{r/2}$ stated in the theorem. That is, in the worst case when the target accuracy ϵ is very small, $r = d$. Therefore, the complexity of AdaLRGD is no worse than that of GD modulo the logarithmic factor.

The AdaLRGD method provides significant oracle complexity gain (compared to GD) particularly for "low-rank" functions. In the language of Theorem 1, such functions satisfy the condition $\Delta_r \leq \epsilon/2 < \Delta_{r/2}$ for a *small* rank r when the accuracy ϵ is *small*. In other words, the sequence of residuals $\Delta_0, \Delta_1, \Delta_2, \Delta_4, \cdots$ for such "low-rank" functions admits a large gap $\Delta_{r/2} \gg \Delta_r$ for a small $r \ll d$.

The AdaLRGD method's ability to reduce the oracle complexity is however prone to a challenge which is characterizing the residuals Δ_r for general functions and distributions. In the following, we discuss a simple and intuitive examples in which such residuals are exactly characterized and AdaLRGD provably slashes the oracle complexity.

A. Intuitive example: convex quadratic

In this section, we demonstrate the benefit of AdaLRGD over GD for the case of quadratic objective as this particular example lets us to characterize a tighter gain for the proposed adaptive method. Consider the quadratic function $f(\theta) = \frac{1}{2}(\theta-\theta^*)^\top Q(\theta-\theta^*)$, where Q is positive definite and admits the SVD $Q = \mathbf{U}\Lambda\mathbf{U}^\top$ for diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_d)$ with $\lambda_1 \geq \cdots \geq \lambda_d$. In Proposition 1, we exactly characterize the residuals Δ_r for initialization particularly picked from the normal distribution around θ^*.

Proposition 1. *For the convex quadratic form above with* λ_i*s as eigenvalues of* Q *and distribution* $\theta_0 \sim \mathcal{N}(\theta^*, \mathbf{I})$*, the residuals for any* $0 \leq r \leq d$ *are* $\Delta_r = 1/2\sum_{i=r+1}^d \lambda_i$*.*

This simple case of convex quadratic with the specified initialization reveals a few insightful remarks. Firstly, the exact characterization of the residuals in Proposition 1 determines the number of iterations per stage, T_s, required by AdaLRGD. Next, consider a fixed target accuracy ϵ and rank $1 \leq r \leq d$ that satisfy the condition $\Delta_r \leq \epsilon/2 < \Delta_{r/2}$. For the case of quadratic function, this condition is equivalent to

$$\lambda_{r+1} + \cdots + \lambda_d \leq \epsilon < \lambda_{r/2+1} + \cdots + \lambda_d. \tag{3}$$

According to Theorem 1, the total oracle complexity of AdaLRGD to reach ϵ-accuracy is bounded by $\mathcal{C}_{\texttt{AdaLRGD}} \leq \kappa r \ln(2r\Delta_0/\epsilon)$, where $\Delta_0 = 1/2\sum_{i=1}^d \lambda_i$ and $\kappa = \lambda_1/\lambda_d$. On the one hand, the accuracy ϵ is typically picked as small as desired. On the other, small ϵ yields larger ranks r that satisfy the condition (3) which further induces larger oracle complexity for AdaLRGD. This tradeoff is indeed critical to determine how much the proposed AdaLRGD is able to save in oracle complexity compared to canonical GD. The condition (3) implies that if the eigenvalues λ_is drop "sharply", then one might be able to satisfy (3) with small ϵ and r as desirable. This particularly holds when λ_is drop exponentially fast discussed in the following example.

Example 1. Consider the quadratic loss defined above where the eigenvalues of the Q matrix drop exponentially fast. More precisely, $\lambda_i = 2^{-i}$ for all $1 \leq i \leq d = 10^4$. For any fixed (normalized) accuracy ϵ/Δ_0, condition (3) yields the feasible rank r which determines the number of stages and also the total oracle complexity of AdaLRGD. Figure 3 demonstrates

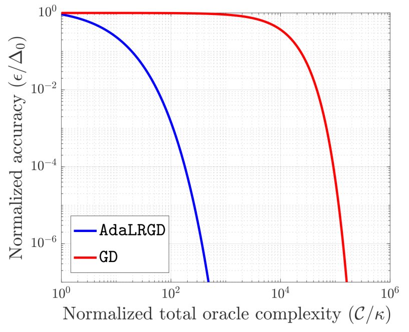

Fig. 3: Accuracy vs. oracle complexity curves for quadratic loss.

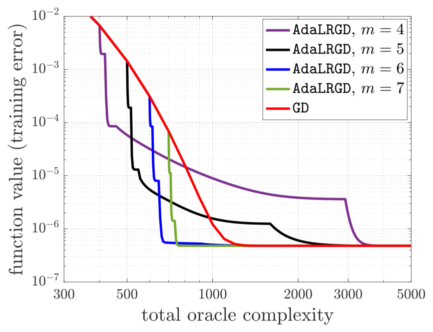

Fig. 4: Total oracle complexity for linear regression with $(n,d)=(10^3,10^2)$.

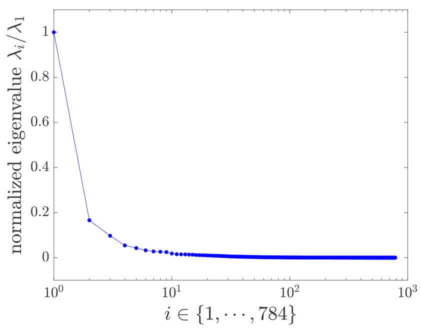

Fig. 5: Eigenvalues of the covariate matrix $XX^\top$ for MNIST digits $\{0,8\}$.

the achievable accuracy-complexity pairs for both AdaLRGD and GD. As demonstrated, AdaLRGD requires orders of magnitude fewer calls to the directional oracle compared to GD for the same target accuracy. This is mainly due to the fast drop of the eigenvalues λ_i which makes the condition (3) hold with small ϵ and r simultaneously. For any accuracy ϵ/Δ_0, condition (3) yields the smallest rank r and thus AdaLRGD's complexity by Theorem 1. As demonstrated in Figure 3, AdaLRGD requires an order of magnitudes fewer oracle calls compared to GD.

IV. Numerical Simulations

In the previous section, we elaborated in detail on the case of quadratic loss function. Here, we provide more evidence from real datasets and demonstrate the applicability of AdaLRGD on different machine learning tasks.

Linear regression: We consider the linear regression problem $\min_{\theta\in\mathbb{R}^d} f(\theta) = \frac{1}{2n}\|X\theta - Y\|^2$, where $X\in\mathbb{R}^{n\times d}$ and $Y\in\mathbb{R}^n$ denote the feature and response variables for $n=1000$ data samples with dimension $d=100$. To embed the low-rank structure in the objective function, we let the covariate matrix be $X=UV^\top$ where $U\in\mathbb{R}^{n\times R}$ and $V\in\mathbb{R}^{d\times R}$ are low-rank random matrices with i.i.d. entries realized from standard normal distribution. Here, the design parameter R determines the efficient rank of X and we pick $R=10$ in our experiment. The response variables are then generated from a linear model $Y=X\theta^*+N$ where $\theta^*=[1\cdots1]^\top$ denotes the ground truth model and N is the matrix of small centered Gaussian noises with variance 0.01. As discussed in Section II, the AdaLRGD algorithm requires the orthogonal matrix $\mathbf{U}$ denoting the direction of the subspaces used in each stage. A practical implementation of AdaLRGD first identifies a fairly *good* MeGO matrix $\mathbf{U}$ by the following approximation (See (2)). AdaLRGD runs a few iterations–denoted by m–of GD and stores the gradient vectors $\nabla f(\theta^i)$ for $i=1,\cdots,m$. The SVD of following empirical MeGO matrix is then used to determine the directions of the active subspace, i.e. the $\mathbf{U}$ matrix in Algorithm 1: $\hat{\mathbf{C}} = 1/m\sum_{i=1}^m \nabla f(\theta^i)\nabla f(\theta^i)^\top$. Figure 4 demonstrates the decay of the training error (or the function value) for GD and AdaLRGD with different choices of the parameter m descried above. It is worth noting that the cost of AdaLRGD on this figure *includes* the initial gradient computation cost for computing the $\hat{\mathbf{C}}$ matrix. Lower picks for m induces less gradient computation cost initially, however, the corresponding $\hat{\mathbf{C}}$ matrix does not accurately identifies the significant directions, hence, AdaLRGD saturates in inaccurate subspaces till the last stage where all the directions are involved and the training error reaches the one for GD eventually. Nonetheless, a proper pick such as $m=7$ enables AdaLRGD to leverage the low-rank structure in the objective and outperform GD. We also monitor the function value and terminate each stage if such improvement is less than a predefined threshold.

Low-rank structures in MNIST: Consider the problem of linear least squares with covariate matrix $X\in\mathbb{R}^{n\times d}$ and response variables $Y\in\mathbb{R}^n$ where n denotes the number samples. The optimization problem can be stated as fitting the parameter vector $\theta\in\mathbb{R}^d$ such that $f(\theta)=\|X\theta-Y\|_2^2$ is minimized which is a quadratic function of the input θ. Therefore, the Hessian of the objective f is $\nabla^2 f(\theta) = X^\top X$. Recall from our discussion in Section III-A that AdaLRGD manifests significant computation reduction (compared to GD) when there is a "sharp" decay in the eigenvalues of $\nabla^2 f$. To examine this, we pick $n=11774$ samples from MNIST dataset consisting of the two digits 0 and 8 [30]. Here, the dimension of the parameter is $d=784$. Figure 5 demonstrates the fast decay of the normalized eigenvalues of $X^\top X$, further highlighting the inherent low-rank structures in data. For instance, in order to reach 0.01-optimal solution of the least square problem above, AdaLRGD will return the desired parameter after involving only $r=224$ directions.

V. Proof of Main Results

A. *Proof of Theorem 1*

As described in Algorithm 1, AdaLRGD is run through stages denoted by $s=0,1,2,\cdots$. For any stage s, let us denote by d_s and $\{\theta_{s,t}: t=0,1,2,\cdots,T_s\}$ the dimension of the active subspace and the iterates of AdaLRGD, respectively. Here, T_s is a positive integer denoting the number of iterate updates in stage s. For simplicity of presentation, we assume that $d_0=1$ and $d=2^S$ for some positive integer S. Recall that $1\le r\le d$ is the rank satisfying $\Delta_r \le \epsilon/2 \le \Delta_{r/2}$. We will later show that AdaLRGD requires only $R:=\log(r)$ stages analyzed in the following.

Stage $s=0$**.** The first stage is initialized with $\theta_{0,0}=\theta_0$ where $\theta_0\sim\rho$. Moreover, the active subspace of this stage is $\mathbf{U}_{d_0} = \mathbf{U}_1 = [\mathbf{u}_1]$. According to `AdaLRGD`, the iterates of the first stage are $\theta_{0,t+1} = \theta_{0,t} - \eta \mathbf{U}_1 \mathbf{U}_1^\top \nabla f(\theta_{0,t})$, for $t = 0, 1, \cdots, T_1 - 1$. Given an orthonormal basis $\mathbf{U}$ for $\mathbb{R}^d$, any fixed $\theta \in \mathbb{R}^d$ and dimension $1 \le r \le d$, we define the function $F_r(\cdot;\theta) : \mathbb{R}^r \to \mathbb{R}$ as follows

$$F_r(\omega;\theta) := f\big(\theta + \mathbf{U}_r(\omega - \mathbf{U}_r^\top \theta)\big), \quad \forall \omega \in \mathbb{R}^r. \quad (4)$$

Lemma 1. *(i) Let F_r denote the function f restricted to subspace $\mathcal{S}(\theta;\mathbf{U}_r)$ as defined in (4). Then, F_r is L-smooth and μ-strongly convex, as is f by assumption.*

(ii) Consider stage s of the `AdaLRGD` algorithm with iterates $\{\theta_{s,t} : t = 0, 1, \cdots\}$ generated with stepsize η and subspace rank d_s. We denote by $\{\omega_t : t = 0, 1, \cdots\}$ GD iterates on F_{d_s} generated by the same stepsize η and initialized with $\omega_0 := \mathbf{U}_{d_s}^\top \theta_{s,0}$, i.e., $\omega_{t+1} = \omega_t - \eta \nabla F_{d_s}(\omega_t;\theta_{s,0})$. Then, for every iteration $t = 0, 1, \cdots$, we have $\omega_t = \mathbf{U}_{d_s}^\top \theta_{s,t}$.

Particularly for rank $r = 1$ and stepsize $\eta = 1/L$, the convergence of the iterates $\{\theta_{0,t}\}$ is as follows

$$f(\theta_{0,t}) - F_1^*(\theta_{0,0}) \le e^{-t/\kappa}\big(f(\theta_{0,0}) - F_1^*(\theta_{0,0})\big).$$

This yields that after T_0 iterations in stage $s = 0$, the final suboptimality is bounded as $f(\theta_{0,T_0}) - f^* \le e^{-T_0/\kappa}\big(f(\theta_{0,0}) - f^*\big) + F_1^*(\theta_{0,0}) - f^*$. The only randomness here is the initialization $\theta_{0,0} = \theta_0 \sim \rho$. Therefore,

$$\mathbb{E}[f(\theta_{0,T_0}) - f^*] \le e^{-T_0/\kappa}\mathbb{E}[f(\theta_{0,0}) - f^*] + \mathbb{E}[F_1^*(\theta_{0,0}) - f^*] = e^{-T_0/\kappa}\Delta_0 + \Delta_1.$$

To determine T_0, we balance the tow terms in the RHS of the above equation, i.e. $e^{-T_0/\kappa}\Delta_0 = \Delta_1$ which yields that $T_0 = \kappa \ln(\Delta_0/\Delta_1)$, and $\mathbb{E}[f(\theta_{0,T_0})] - f^* \le 2\Delta_1$.

Stage $1 \le s \le R-1$. As described in Algorithm 1, the rank of the active subspace is doubled in each stage which implies that $d_s = 2d_{s-1} = 2^s$. Moreover, we initialize each stage with the final iterate of the previous stage, that is, $\theta_{s,0} = \theta_{s-1,T_{s-1}}$. By similar arguments made in stage $s = 0$ and employing Lemma 1, after T_s iteration in stage s, the final expected suboptimality can be bounded as follows

$$\mathbb{E}[f(\theta_{s,T_s}) - f^*] \le e^{-T_s/\kappa}\mathbb{E}[f(\theta_{s,0}) - F_{d_s}^*(\theta_{s,0})] + \mathbb{E}[F_{d_s}^*(\theta_{s,0}) - f^*]. \quad (5)$$

Next, we bound each of the two terms in the RHS of (5) separately. The first term can be bounded as follows

$$\mathbb{E}[f(\theta_{s,0}) - F_{d_s}^*(\theta_{s,0})] \le \mathbb{E}[f(\theta_{s,0})] - f^* = \mathbb{E}[f(\theta_{s-1,T-1})] - f^* \le 2\Delta_{d_{s-1}}, \quad (6)$$

where we used the initialization rule and the fact that $2\Delta_{d_{s-1}}$ upper bounds the suboptimality of the final iterate of the previous stage. The second term in the RHS of (5) is bounded as follows which is stated and proved in Lemma 2,

$$\mathbb{E}[F_{d_s}^*(\theta_{s,0}) - f^*] = \mathbb{E}[F_{d_s}^*(\theta_0)] - f^* = \Delta_{d_s}. \quad (7)$$

Lemma 2. *Consider* `AdaLRGD` *with initialization $\theta_0 \sim \rho$ and stage s with initialization $\theta_{s,0}$. Then, for any s, we have*

$$\mathbb{E}_{\theta_0}[F_{d_s}^*(\theta_{s,0})] - f^* = \mathbb{E}_{\theta_0}[F_{d_s}^*(\theta_0)] - f^* = \Delta_{d_s}.$$

where $F_r^*(\theta) := \min_{\theta' \in \mathcal{D}(\theta_0;r)} f(\theta')$ for any θ and $1 \le r \le d$.

Putting (6) and (7) together with (5) yields that $\mathbb{E}[f(\theta_{s,T_s})] - f^* \le 2e^{-T_s/\kappa}\Delta_{d_{s-1}} + \Delta_{d_s}$. Balancing the two terms above yields the required number of iterates for stage s and its final suboptimality as $T_s = \kappa \ln(2\Delta_{d_{s-1}}/\Delta_{d_s})$, and $\mathbb{E}[f(\theta_{s,T_s})] - f^* \le 2\Delta_{d_s}$. Note that the suboptimality of the final iterate in stage $R-1$ is at least $2\Delta d_{R-1} \ge \epsilon$ which may not be as small as desired.

Stage $s = R$. First note that after T_R iteration in this stage, the suboptimality is bounded as $\mathbb{E}[f(\theta_{R,T_R})] - f^* \le 2e^{-T_R/\kappa}\Delta_{d_{R-1}} + \Delta_{d_R}$. Second, given the assumption $\Delta_R \le \epsilon/2$, it suffices to run this stage of `AdaLRGD` for $T_R = \kappa \ln(4\Delta_{d_{R-1}}/\epsilon)$ iterations, which together with the assumption $\Delta_{d_R} \le \epsilon/2$ yields the final suboptimality $\mathbb{E}[f(\theta_{R,T_R})] - f^* \le \epsilon/2 + \epsilon/2 = \epsilon$.

Total oracle complexity of `AdaLRGD`. Note that in each stage $0 \le s \le R$, `AdaLRGD` updates the iterates for T_s iteration each costing $d_s = 2^s$ (directional) gradient computation. Putting all together yields the total oracle complexity

$$\mathcal{C}_{\texttt{AdaLRGD}} = \sum_{s=0}^{R} d_s T_s \le \kappa \ln\left(\frac{\Delta_0}{\Delta_1}\right) + \sum_{s=1}^{R-1} 2^s \kappa \ln\left(2\frac{\Delta_{d_{s-1}}}{\Delta_{d_s}}\right) + \kappa 2^R \ln\left(4\frac{\Delta_{d_{R-1}}}{\epsilon}\right) \le \kappa r \ln\left(2r\frac{\Delta_0}{\epsilon}\right).$$

B. *Proof of Lemma 1*

(i) For $\theta' := \theta_0 + \mathbf{U}_r(\omega - \mathbf{U}_r^\top \theta_0)$, gradient and the Hessian of F_r (with respect to ω) are $\nabla F_r(\omega;\theta_0) = \mathbf{U}_r^\top \nabla f(\theta')$ and $\nabla^2 F_r(\omega;\theta_0) = \mathbf{U}_r^\top \nabla^2 f(\theta')\mathbf{U}_r$. For any $\omega' \in \mathbb{R}^r$,

$$\omega'^\top \nabla^2 F_r(\omega;\theta_0)\omega' = \omega'^\top \mathbf{U}_r^\top \nabla^2 f(\theta')\mathbf{U}_r\omega' \le L\|\mathbf{U}_r\omega'\|^2 = L\|\omega'\|^2,$$

where we used the facts that (i) $\mathbf{U}_r^\top \mathbf{U}_r = \mathbf{I}$ and (ii) the eigenvalues of the Hessian of f are at most L, i.e. $\nabla^2 f(\theta') \preceq L\mathbf{I}$ for any θ'. Similarly, one can verify that for any $\omega' \in \mathbb{R}^r$, we have that $\omega'^\top \nabla^2 F_r(\omega;\theta_0)\omega' \ge \mu\|\omega'\|^2$ which yields that $\mu\mathbf{I} \preceq \nabla^2 F(\omega;\theta_0) \preceq L\mathbf{I}$. In other words, $F_r(\cdot;\theta_0)$ is also μ-strongly convex and L-smooth.

(ii) Let us denote $r = d_s$ and drop the subscript s from all the indices for simplicity of notation. By our defined initialization, we have $\omega_0 = \mathbf{U}_r^\top \theta_0$. Now, for some $k \ge 0$, assume that $\omega_t = \mathbf{U}_r^\top \theta_t$ for all $0 \le t \le k$. Then, the next GD iterate $k+1$ we can write $\omega_{k+1} = \omega_k - \eta \nabla F(\omega_k;\theta_0) = \omega_k - \eta \mathbf{U}_r^\top \nabla f\big(\theta_0 + \mathbf{U}_r(\omega_k - \mathbf{U}_r^\top \theta_0)\big)$. The argument of $\nabla f(\cdot)$ here can be rewritten as $\theta_0 + \mathbf{U}_r(\omega_k - \mathbf{U}_r^\top \theta_0) = \theta_0 + \mathbf{U}_r(\mathbf{U}_r^\top \theta_k - \mathbf{U}_r^\top \theta_0) = \theta_k + \mathbf{U}_{r^\perp}\mathbf{U}_{r^\perp}^\top(\theta_0 - \theta_k) = \theta_k$ which implies that $\omega_{k+1} = \omega_k - \eta \mathbf{U}_r^\top \nabla f(\theta_k)$ with $\mathbf{U}_{r^\perp} := [\mathbf{u}_{r+1} \cdots \mathbf{u}_d]$. On the other hand, `AdaLRGD`'s iterates can be written as $\theta_{k+1} = \theta_k - \eta \mathbf{U}_r \mathbf{U}_r^\top \nabla f(\theta_k)$. All in all, we have $\mathbf{U}_r^\top \theta_{k+1} = \mathbf{U}_r^\top \theta_k - \eta \mathbf{U}_r^\top \mathbf{U}_r \mathbf{U}_r^\top \nabla f(\theta_k) = \omega_k - \eta \mathbf{U}_r^\top \nabla f(\theta_k) = \omega_{k+1}$ which concludes the induction lemma.

C. *Proof of Lemma 2*

The claim holds for $s = 0$ by definition. Consider any $s \ge 1$ and fix θ_0. Recall from definition that

$$F_{d_s}^*(\theta_0) = \min_{\theta \in \mathcal{S}(\theta_0;\mathbf{U}_{d_s})} f(\theta), \; F_{d_s}^*(\theta_{s,0}) = \min_{\theta \in \mathcal{S}(\theta_{s,0};\mathbf{U}_{d_s})} f(\theta). \; (8)$$

where we denote $\mathcal{S}(\theta; \mathbf{U}_r) := \theta + \operatorname{span}(\mathbf{U}_r)$ for any θ and r. Next, we show that the two subspaces $\mathcal{S}(\theta_0; \mathbf{U}_{d_s})$ and $\mathcal{S}(\theta_{s,0}; \mathbf{U}_{d_s})$ are indeed equal. To do so, first note that all the iterates of stage $s = 0$ live in $\mathcal{S}(\theta_0; \mathbf{U}_1)$. Particularly for the final iterate of stage $s = 0$ (which is equal to the first iterate of stage $s = 1$), $\theta_{1,0} = \theta_{0,T_0} \in \mathcal{S}(\theta_0; \mathbf{U}_1)$. Similarly, all the iterates of stage $s = 1$ live in $\mathcal{S}(\theta_{1,0}; \mathbf{U}_2)$. Given the fact that $\theta_{1,0} \in \mathcal{S}(\theta_0; \mathbf{U}_1)$ and $\operatorname{span}(\mathbf{U}_1) \subseteq \operatorname{span}(\mathbf{U}_2)$, we conclude that $\mathcal{S}(\theta_{1,0}; \mathbf{U}_2) = \mathcal{S}(\theta_0; \mathbf{U}_2)$ yielding the claim for $s = 1$. To continue the induction argument, assume that the claim holds for $s \geq 1$, i.e. $\mathcal{S}(\theta_0; \mathbf{U}_{d_s}) = \mathcal{S}(\theta_{s,0}; \mathbf{U}_{d_s})$. Note that $\theta_{s+1,0} = \theta_{s,T_s} \in \mathcal{S}(\theta_{s,0}; \mathbf{U}_{d_s})$, hence $\theta_{s+1,0} \in \mathcal{S}(\theta_0; \mathbf{U}_{d_s})$ and since $\mathcal{S}(\theta_0; \mathbf{U}_{d_s}) \subseteq \mathcal{S}(\theta_0; \mathbf{U}_{d_{s+1}})$, we conclude that $\theta_{s+1,0} \in \mathcal{S}(\theta_0; \mathbf{U}_{d_{s+1}})$. All in all, the claim is concluded, i.e. $\mathcal{S}(\theta_0; \mathbf{U}_{d_{s+1}}) = \mathcal{S}(\theta_{s+1,0}; \mathbf{U}_{d_{s+1}})$.

Having proved that the feasible set of the two minimization defined in (8) are identical, we have that $F^*_{d_s}(\theta_0) = F^*_{d_s}(\theta_{s,0})$. Note that $\theta_0 \sim \rho$ is the only source of randomness. Taking expectation with respect to θ_0 from both sides of the last inequality yields the claim of the lemma.

D. *Proof of Proposition 1*

First, $\Delta_r = \min_{\theta \in \mathcal{S}(\theta_0; \mathbf{U}_r)} f(\theta) = \min_{\omega \in \mathbb{R}^r} F_r(\omega; \theta_0)$, since $f^* = 0$. Let us denote $\omega = [\omega(1), \cdots, \omega(r)]^\top$ and $\theta' = \theta + \mathbf{U}_r(\omega - \mathbf{U}_r^\top \theta)$. We have that $F_r(\omega; \theta_0) = f(\theta') = \frac{1}{2}(\theta' - \theta^*)^\top Q(\theta' - \theta^*) = \frac{1}{2}\sum_{i=1}^d \lambda_i \langle \mathbf{u}_i, \theta' - \theta^* \rangle^2$. Note that for $1 \leq i \leq r$ we have $\langle \mathbf{u}_i, \theta' - \theta^* \rangle = \langle \mathbf{u}_i, \theta_0 - \theta^* \rangle + \omega(i) - \langle \mathbf{u}_i, \theta_0 \rangle = \omega(i) - \langle \mathbf{u}_i, \theta^* \rangle$, and for $r+1 \leq i \leq d$ it holds that $\langle \mathbf{u}_i, \theta' - \theta^* \rangle = \langle \mathbf{u}_i, \theta_0 - \theta^* \rangle$. Therefore, for $\omega^*(i) = \langle \mathbf{u}_i, \theta^* \rangle$,

$$\min_{\omega \in \mathbb{R}^r} F_r(\omega; \theta_0) = \min_{\omega \in \mathbb{R}^r} \frac{1}{2} \sum_{i=1}^r \lambda_i \left(\omega(i) - \langle u_i, \theta^* \rangle \right)^2$$
$$+ \frac{1}{2} \sum_{i=r+1}^d \lambda_i \langle \mathbf{u}_i, \theta_0 - \theta^* \rangle^2 = \frac{1}{2} \sum_{i=r+1}^d \lambda_i \langle \mathbf{u}_i, \theta_0 - \theta^* \rangle^2.$$

As a result, we conclude $\Delta_r = \mathbb{E}\big[\min_{\omega \in \mathbb{R}^r} F_r(\omega; \theta_0)\big] = \frac{1}{2}\sum_{i=r+1}^d \lambda_i \mathbb{E}[\langle \mathbf{u}_i, \theta_0 - \theta^* \rangle^2] = \frac{1}{2}\sum_{i=r+1}^d \lambda_i$.

References

[1] 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

[2] 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.

[3] V. Papyan, "The full spectrum of deepnet Hessians at scale: Dynamics with SGD training and sample size," June 2019, arXiv:1811.07062v2 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1811.07062

[4] 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

[5] 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.

[6] T. Le and S. Jegelka, "Training invariances and the low-rank phenomenon: beyond linear networks," in *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*, Virtual, April 25-29 2022, pp. 1–26.

[7] T. Galanti and T. Poggio, "Sgd noise and implicit low-rank bias in deep neural networks," Center for Brains, Minds and Machines (CBMM), Tech. Rep., 2022.

[8] A. S. Nemirovskiĭ and D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*, ser. Wiley-Interscience Series in Discrete Mathematics and Optimization. New York, NY, USA: John Wiley & Sons Inc., 1983.

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

[10] R. Cosson, A. Jadbabaie, A. Makur, A. Reisizadeh, and D. Shah, "Gradient descent for low-rank functions," *arXiv preprint arXiv:2206.08257*, 2022.

[11] L. Sagun, L. Bottou, and Y. LeCun, "Eigenvalues of the hessian in deep learning: Singularity and beyond," *arXiv preprint arXiv:1611.07476*, 2016.

[12] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, "Empirical analysis of the hessian of over-parametrized neural networks," *arXiv preprint arXiv:1706.04454*, 2017.

[13] B. Ghorbani, S. Krishnan, and Y. Xiao, "An investigation into neural net optimization via hessian eigenvalue density," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2232–2241.

[14] A. R. Sankar, Y. Khasbage, R. Vigneswaran, and V. N. Balasubramanian, "A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, 2021, pp. 9481–9488.

[15] N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma, "Finding approximate local minima faster than gradient descent," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 1195–1199.

[16] V. Feinberg, X. Chen, Y. J. Sun, R. Anil, and E. Hazan, "Sketchy: Memory-efficient adaptive regularization with frequent directions," *arXiv preprint arXiv:2302.03764*, 2023.

[17] Z. Xie, Q.-Y. Tang, Y. Cai, M. Sun, and P. Li, "On the power-law hessian spectrums in deep learning," *arXiv preprint arXiv:2201.13011*, 2022.

[18] R. Pan, H. Ye, and T. Zhang, "Eigencurve: Optimal learning rate schedule for sgd on quadratic objectives with skewed hessian spectrums," in *International Conference on Learning Representations*.

[19] 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.

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

[21] 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.

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

[23] 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

[24] ——, "Federated optimization of smooth loss functions," January 2022, arXiv:2201.01954 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2201.01954

[25] 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.

[26] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, December 1964.

[27] Y. E. Nesterov, "A method of solving a convex programming problem with convergence rate $O\big(\frac{1}{k^2}\big)$," *Doklady Akademii Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983, in Russian.

[28] 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.

[29] P. G. Constantine, *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM, 2015.

[30] Y. LeCun, C. Cortes, and C. J. C. Burges, "THE MNIST DATABASE of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/