On the Oracle Complexity of Interpolation-Based Gradient Descent

Dongmin Lee, William Lu, and Anuran Makur

***Abstract*— Recent work on first-order optimizers for empirical risk minimization (ERM) has suggested that smoothness of ERM loss functions in the training data, rather than in the optimization parameters, can be leveraged to improve the oracle complexity of gradient descent (GD) methods. In this paper, we propose an inexact gradient method, *piecewise polynomial interpolation-based gradient descent* (PPI-GD), which approximates the full gradient in each iteration by querying the first-order oracle at equidistant points in the data domain to construct polynomial interpolants of the resulting gradient samples over appropriately sized patches of the data domain. We analyze the oracle complexity of PPI-GD for strongly convex and non-convex loss functions when the data space dimension is bounded by a polylogarithmic function of the number of training samples, and find it to outperform several GD variants in key regimes when the loss function is sufficiently smooth. Furthermore, our analysis extends several techniques from the error analysis of bicubic spline interpolants to the setting of d-variate tensor product polynomial interpolants which may be of independent interest in interpolation analysis.**

***Index Terms*— First-order optimization, inexact gradient descent, oracle complexity, polynomial interpolation**

I. Introduction

First-order optimization algorithms such as gradient descent (GD) and its variants are commonly used within the paradigm of empirical risk minimization (ERM) to optimize loss functions over a finite training dataset. Theoretical work on the performance of such algorithms has centered around the notion of *oracle complexity* introduced by [1], defined as the number of gradient queries necessary to guarantee convergence (e.g., to the optimal objective value) up to some absolute error ε. Classical results in this area include upper bounds for GD under assumptions like convexity and strong convexity [2], upper bounds for stochastic gradient descent (SGD) with constant and decreasing step sizes [3], [4], and information-theoretic lower bounds [5], among others.

Building upon work on inexact gradient descent methods (e.g., [6]–[9]), a recent line of research initiated by [10] aims to improve upon the oracle complexity of classical gradient-based optimizers by exploiting the smoothness of the ERM loss function with respect to the data, a characteristic of many machine learning objectives which was hitherto neglected in previous literature on the subject. Specifically, the authors consider the ERM formulation (cf. [11])

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i;\boldsymbol{\theta})\,,$$

where n is the number of training samples and the loss function f belongs to an (ℓ, L_h)-Hölder class with respect to the training instance $\mathbf{x}_i$. The authors propose a method dubbed *local polynomial interpolation-based gradient descent* (LPI-GD), which evaluates $\boldsymbol{\nabla}_{\boldsymbol{\theta}} f$ at a grid of virtual data points and performs local polynomial regression to estimate the full training gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}} F$ on each iteration without incurring n oracle calls. For strongly convex problems with $d = O(\log\log(n))$ data dimensions and p parameter dimensions, the authors establish the oracle complexity of LPI-GD as $\tilde{O}((p/\varepsilon)^{d/(2\ell)})$, beating the $O(n\log(p/\varepsilon))$ complexity of GD and the $O(p/\varepsilon)$ complexity of SGD when ℓ is sufficiently large, $p = O(\mathsf{poly}(n))$, and $\varepsilon = \Theta(\mathsf{poly}(1/n))$. However, the assumption that $d = O(\log\log(n))$ limits the theoretical relevance of LPI-GD to very low-dimensional datasets, and it is conjectured that this dependence can be improved to d scaling polylogarithmically in n.

In this paper, motivated by this conjecture and intrinsic theoretical interest in the application of numerical analysis techniques within the oracle complexity domain, we introduce the *piecewise polynomial interpolation-based gradient descent* (PPI-GD) method, which achieves the aforementioned $\tilde{O}((p/\varepsilon)^{d/(2\ell)})$ oracle complexity for strongly convex ERM problems in the substantially broader regime of data dimensionality $d = O(\log^{0.49}(n))$. Hence, this method and its analysis resolve the conjectured scaling regime of d. In addition, we also analyze the oracle complexity of PPI-GD for non-convex loss functions and show that it outperforms GD and SGD in a similar regime. At a high level, PPI-GD divides the data space into patches with $\ell+1$ grid points along each axis and fits a tensor product polynomial interpolant to the gradient values at the grid points for each patch. We establish the oracle complexity of our method by generalizing proof techniques initially considered in the context of bicubic splines to the setting of multivariate polynomial interpolants. Next, we enumerate the principal contributions of our work and summarize the relevant literature on oracle complexity and first-order optimizers.

The author ordering is alphabetical. This work was supported by the National Science Foundation (NSF) CAREER Award under Grant CCF-2337808.

Dongmin Lee and William Lu are with the Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA (e-mail: lee4818@purdue.edu; lu909@purdue.edu).

Anuran Makur is with the Department of Computer Science and the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (e-mail: amakur@purdue.edu).

A. Main Contributions

Our work makes the following contributions:

1) We present a new first-order optimization algorithm, PPI-GD (Algorithm 1), to solve the ERM optimization problem when the loss function satisfies standard Lipschitz continuity and Hölder smoothness assumptions in the parameters and data instance, respectively. Although theoretical analysis is the focus of our work, we also discuss potential optimizations and other practical considerations in implementing our algorithm.
2) We derive the oracle complexity of PPI-GD in the strongly convex setting (Theorem 7). We remark that PPI-GD improves upon the LPI-GD algorithm proposed in [10] by achieving an equivalent oracle complexity under weaker assumptions on the order of d, as summarized in Table I. Moreover, in the important regime where $p = O(\mathsf{poly}(n))$ and $\varepsilon = \Theta(\mathsf{poly}(1/n))$, PPI-GD outperforms GD, SGD, and their variants in oracle complexity when the loss function is sufficiently smooth with respect to the data instance (Proposition 8).
3) Analogously to the above, we establish the oracle complexity of PPI-GD in the non-convex setting (Theorem 9) and show asymptotic dominance over GD and SGD in the regime of $p = O(\mathsf{poly}(n))$ and $\epsilon = \Theta(\mathsf{poly}(1/n))$ (Proposition 10)—also see Table I.
4) In service of deriving the oracle complexity results above, we make several key observations regarding multivariate polynomial interpolation which have not been covered in the existing literature on numerical analysis. In particular, we establish an error bound for tensor product polynomial interpolants (Theorem 1) by extending the notion of *blended interpolants* from classical analyses of bicubic splines to the general case of arbitrary degree and dimensionality.

B. Related Literature

There is a wealth of existing literature on first-order optimization, and we provide a survey of the salient developments therein. The notion of oracle complexity introduced in [1], defined as the number of gradient calls necessary for a given algorithm to achieve convergence within some absolute error ε, is used as a measure of algorithmic performance agnostic to the objective function under optimization. We refer readers to [2] for further exposition on oracle complexity, and we remark that the convergence rates of GD and SGD for smooth functions in the convex, strongly convex, and non-convex settings are well-known results [2], [12]–[14]. Earlier work implicitly analyzed SGD in the context of stochastic approximation algorithms [15], and more recent work [16] has analyzed SGD in canonical and robust settings [13]. Since then, numerous variations of GD and SGD have been proposed to bolster their performance in a number of scenarios. For example, mini-batch SGD (MBGD) [13], [17] improves wall-clock time in parallel computation settings and convergence rate in sparse regression problems [18], momentum (heavy-ball method) [19] and Nesterov acceleration [20] improve convergence rate in, e.g., strongly convex settings, and specialized versions of GD provide performance gains when optimizing low-rank functions [21]–[23]. SGD has been augmented by several variance reduction methods originating from Monte Carlo sampling theory [24, Chapter 9], most notably stochastic variance reduced gradient (SVRG) [25], stochastic average gradient (SAG) [26], [27], and stochastic dual coordinate ascent (SDCA) [28]. Adaptive learning rates for GD have been proposed, including adaptive gradient (AdaGrad) [29] and its generalization to non-convex objectives [30], root mean squared propagation (RMSprop) [31], and adaptive moments (Adam) [32]. We refer readers to [13], [33] for a broader overview of the aforementioned methods, and to [34], [35] for an overview of empirical risk minimization in machine learning. Lastly, classical results on oracle complexity also include the lower bounds in [2], [5], [36]–[38]; we focus exclusively on upper bounds in this work.

Of particular relevance to our present work is the literature on *inexact gradient descent* methods, which includes well-known results in the convex [6], [9] and strongly convex [7], [8] settings. More recently, [39] analyzes the convergence behavior of GD methods with non-vanishing gradient error, [40] utilizes inexact GD methods to study the convergence rate of a dual decomposition problem, and [41] investigates how bias in the gradient oracle assists first-order optimizers in escaping saddle points on non-convex objectives. Inexact GD methods have been generalized to incorporate Nesterov acceleration [42]–[44], and rates of convergence have been studied for such extensions. Similarly in spirit to our work, [45] introduces *biased gradient oracles* to model the optimization of functions estimated with a batch size parameter, and quantifies the convergence rate of a randomized stochastic gradient algorithm in this regime.

Due to the utility of gradient information in inference tasks such as variable selection, a prior line of work on *gradient estimation* aims to simultaneously learn both the model and the gradient of the loss function, using local polynomial fitting [46], local polynomial regression [47], and kernel methods [48], among others. More recent work has utilized gradient estimation based on methods such as nearest neighbors to design zeroth-order (gradient-free) optimizers [49], [50]. Lastly, first-order methods which use gradient estimation include the recently proposed LPI-GD [10], as well as techniques which leverage smoothness of loss functions for federated optimization [51]. We remark that work in this vein often makes the assumption that d grows moderately with respect to n (for example, $d = O(\log n)$), due to the *curse of dimensionality* as described in [46, Section 7.1]. This assumption holds for practical applications in domains such as healthcare [52] and control systems [53], or problems where dimensionality reduction methods such as principal component analysis [54], Laplacian eigenmaps [55], modal decompositions [56], or isometric embedding theorems such as the *Johnson-Lindenstrauss lemma* [57] may be applied before performing ERM.

C. Outline

Finally, we delineate the structure of our paper. In Section II, we introduce applicable notation, formally define the ERM

TABLE I: Comparison of the oracle complexity of some first-order optimization algorithms, where the loss function satisfies the assumptions outlined in Section II-C and $\gamma \in (0, 1/2)$ is an arbitrary constant.

Algorithm	Required order of d	Oracle complexity for strongly convex loss	Oracle complexity for non-convex loss
GD	-	$O\big(n \log\big(\frac{p}{\varepsilon}\big)\big)$ [2]	$O\big(n \frac{p}{\varepsilon^2}\big)$ [12]
SGD	-	$O\big(\frac{p}{\varepsilon}\big)$ [13]	$O\Big(\big(\frac{p}{\varepsilon^2}\big)^2\Big)$ [14]
LPI-GD	$O(\log\log(n))$	$O\Big(\exp\big(2\sqrt{\log(n)}\big)\big(\frac{p}{\varepsilon}\big)^{d/(2\ell)} \log\big(\frac{p}{\varepsilon}\big)\Big)$ [10]	-
PPI-GD	$O(\log^{\gamma}(n))$	$O\Big(\exp\big(\log^{2\gamma}(n)\big)\big(\frac{p}{\varepsilon}\big)^{d/(2\ell)} \log\big(\frac{p}{\varepsilon}\big)\Big)$ (Theorem 7)	$O\Big(\exp\big(\log^{2\gamma}(n)\big)\big(\frac{p}{\varepsilon^2}\big)^{1+d/(2\ell)}\Big)$ (Theorem 9)

optimization problem under consideration, and specify the assumptions we impose in our analysis of this problem. In Section III-A, we formally present the PPI-GD algorithm for solving the ERM optimization problem. Auxiliary results concerning polynomial interpolation error are presented in Section III-B in preparation for the ensuing discussion on PPI-GD's oracle complexity in Section III-C. We defer formal proofs of the polynomial interpolation error bounds and oracle complexity results to Sections IV and V, respectively. Lastly, we provide some numerical simulations of PPI-GD in Section VI and suggest directions for future inquiry in Section VII.

II. Formal Model and Setup

A. Notation

Let $\mathbb{Z}_{>0}$ denote the natural numbers starting from 1 and let $\mathbb{Z}_{\geq 0} = \mathbb{Z}_{>0} \cup \{0\}$. Let $[\![a, b]\!] = [a, b] \cap \mathbb{Z}$ and $[a] = [\![1, a]\!]$ denote integer intervals. Lowercase and uppercase bold letters denote vectors and matrices, respectively. Uppercase calligraphic letters denote sets unless otherwise stated. Uppercase non-italic letters denote operators. Given a set $\mathcal{S}$, let $2^{\mathcal{S}}$ denote its power set. Given a collection of sets $\mathcal{S}_i$ indexed by $i \in [d]$, let $\bigtimes_{i\in[d]} \mathcal{S}_i = \mathcal{S}_1 \times \cdots \times \mathcal{S}_d$ denote their Cartesian product. Let $\mathcal{C}^p(\mathcal{X})$ denote the class of all functions $f : \mathcal{X} \to \mathbb{R}$ which are p times continuously differentiable (i.e., all pth order partial derivatives exist and are continuous on $\mathcal{X}$). In the context of Landau notation, we use $\tilde{O}(\cdot)$ to hide subpolynomial factors, i.e., factors which are asymptotically dominated by *every* function n^{β} for $\beta > 0$. We use $\mathbb{1}\{\cdot\}$ to denote the Iverson bracket, which equals 1 if the input proposition is true and 0 otherwise. Let $\|\cdot\|_p$ denote the $\mathcal{L}^p$-norm on vectors or functions for $p \in [1, \infty]$.

The standard multi-index notation is used for conciseness in the multivariate setting. Namely, given a d-tuple multi-index $\mathbf{s} = (s_1, \ldots, s_d) \in \mathbb{Z}^d_{\geq 0}$, we let $|\mathbf{s}| = s_1 + \cdots + s_d$, $\mathbf{x}^{\mathbf{s}} = x_1^{s_1} \cdots x_d^{s_d}$ for $\mathbf{x} \in \mathbb{R}^d$, and $\mathbf{s}! = s_1! \cdots s_d!$. In addition, given a function $f : \mathbb{R}^d \to \mathbb{R}$, we let

$$f^{(\mathbf{s})} = \frac{\partial^{|\mathbf{s}|} f}{\partial x_1^{s_1} \cdots \partial x_d^{s_d}} .$$

Given a domain $\mathcal{X} \subset \mathbb{R}$, a function $h : \mathcal{X} \to \mathbb{R}$, and $n+1$ points $\{z_i\}_{i=1}^{n+1} \subset \mathcal{X}$ in the domain, let $h[z_1, \ldots, z_{n+1}]$ denote the *nth divided difference* of h at $\{z_i\}_{i=1}^{n+1}$, given by the recurrence [58, p. 308]

$$\forall i \leq j,\ h[z_i, \ldots, z_j] = \begin{cases} h(z_i)\,, & \text{if } i = j\,, \\ \frac{h[z_{i+1}, \ldots, z_j] - h[z_i, \ldots, z_{j-1}]}{z_j - z_i}\,, & \text{otherwise}\,. \end{cases} \quad (1)$$

B. Problem Statement

Let $p \in \mathbb{Z}_{>0}$ be the parameter space dimension and $d \in \mathbb{Z}_{>0}$ be the data space dimension. Let $\{\mathbf{x}^{(i)} : i \in [n]\} \subset \mathcal{X}$ be a training set of n data samples, where $\mathcal{X}$ is a bounded data space which we take to be $\mathcal{X} = [0, 1]^d$ for simplicity. Let $f : \mathcal{X} \times \mathbb{R}^p \to \mathbb{R}$ be a loss function. The loss for a data sample $\mathbf{x}^{(i)}$ and a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$ is $f(\mathbf{x}^{(i)}; \boldsymbol{\theta})$, and the ERM objective function $F : \mathbb{R}^p \to \mathbb{R}$ is the empirical average of the loss over the training set:

$$F(\boldsymbol{\theta}) \triangleq \frac{1}{n} \sum_{i=1}^{n} f\big(\mathbf{x}^{(i)}; \boldsymbol{\theta}\big)\,.$$

We assume that the infimum of the objective function,

$$F^* \triangleq \inf_{\boldsymbol{\theta} \in \mathbb{R}^p} F(\boldsymbol{\theta}),$$

exists and is finite. In the strongly convex setting, we consider algorithms A for finding an ε-approximation of the optimum F^*, i.e., A returns a parameter vector $\boldsymbol{\theta}^* \in \mathbb{R}^p$ which satisfies

$$F(\boldsymbol{\theta}^*) - F^* \leq \varepsilon\,, \quad (2)$$

for a given $\varepsilon > 0$. In the non-convex setting where finding the global minimum can be intractable, we instead require algorithms A to use the notion of an ε-stationary point, i.e., they return a parameter vector $\boldsymbol{\theta}^* \in \mathbb{R}^p$ satisfying

$$\|\boldsymbol{\nabla} F(\boldsymbol{\theta}^*)\|_2 \leq \varepsilon. \quad (3)$$

To this end, the algorithm is provided with a first-order oracle $\mathcal{O} : \mathcal{X} \times \mathbb{R}^p \to \mathbb{R}$ [1], which returns the gradient of f at a given point $\mathbf{x} \in \mathcal{X}$ in the data space and a given parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$:

$$\mathcal{O}(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta})\,.$$

We remark that computing the full gradient of F requires n oracle calls.

The *first-order oracle complexity* $\Gamma(\mathrm{A})$ of an algorithm A is the minimum number of oracle calls necessary for A to find an ε-approximate solution as defined in (2) (or an ε-stationary solution as in (3)). This notion of algorithmic complexity, first suggested by [1], provides an accurate representation of the algorithm's performance when gradient computation is the bottleneck, as is often the case in practice.

C. Assumptions

Throughout this work, we make the following assumptions on the loss function which are standard in the optimization literature:

1) *(Smoothness in parameters)* There exists $L_l > 0$ such that for all $\mathbf{x} \in \mathcal{X}$, the gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta})$ is L_l-Lipschitz continuous with respect to $\boldsymbol{\theta}$:

$$\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^p, \\ \left\|\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta}_1) - \boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta}_2)\right\|_2 \le L_l \left\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\right\|_2 .$$

2) *(Hölder smoothness in data)* There exist $\ell \in \mathbb{Z}_{>0}$ and $L_h > 0$ such that for all $i \in [p]$, $g_i(\mathbf{x}) = \frac{\partial}{\partial \theta_i} f(\mathbf{x};\boldsymbol{\theta})$ is ℓ times differentiable and belongs in the $\mathcal{H}(\ell, L_h)$-Hölder class:

$$\forall \mathbf{s} \in \mathbb{Z}^d_{\ge 0} \text{ such that } |\mathbf{s}| = \ell - 1, \ \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}, \\ \left| g_i^{(\mathbf{s})}(\mathbf{x}_1) - g_i^{(\mathbf{s})}(\mathbf{x}_2) \right| \le L_h \left\|\mathbf{x}_1 - \mathbf{x}_2\right\|_1 .$$

When we analyze PPI-GD in the strongly convex setting, we additionally make the following assumption:

3) *(Strong convexity in parameters)* There exists $\mu > 0$ such that for all $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{x};\boldsymbol{\theta})$ is μ-strongly convex with respect to $\boldsymbol{\theta}$:

$$\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^p, \ f(\mathbf{x};\boldsymbol{\theta}_1) \ge f(\mathbf{x};\boldsymbol{\theta}_2) \\ + \boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta}_2)^{\mathsf{T}} (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) + \frac{\mu}{2} \left\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\right\|_2^2 .$$

Assumptions 1 and 3 are widely adopted in the analysis of gradient descent methods [2], [13], [22], [25], and the Hölder class used in Assumption 2 is standard in the literature to describe a function's higher-order smoothness [59]–[61]. Intuitively, Hölder classes roughly capture how closely a function matches its Taylor polynomials of a certain degree. Our definition of the multivariate Hölder class is a commonly adopted generalization of the usual univariate definition in [59] (see [60], [62], [63]). Note that $\mathcal{H}(\ell, L_h)$-Hölder smoothness implies

$$\forall i \in [p], \ \forall \mathbf{s} \in \mathbb{Z}^d_{\ge 0} \text{ with } |\mathbf{s}| = \ell, \ \sup_{\mathbf{x} \in \mathcal{X}} \left| g_i^{(\mathbf{s})}(\mathbf{x}) \right| \le L_h . \quad (4)$$

Lastly, we remark that the assumptions delineated above encompass a range of machine learning models. For example, $\mathcal{L}^2$-regularized linear regression (also known as ridge regression) with regularization parameter λ satisfies Assumption 1 with $L_l = 2d + \lambda - 2$, Assumption 2 with $\ell = 2$ and $L_h = 4$, and Assumption 3 with $\mu = \lambda$ [10]. Similarly, $\mathcal{L}^2$-regularized logistic regression satisfies Assumption 1 with $L_l = (d-1)/2 + \lambda$, Assumption 2 with $\ell = 2$ and $L_h = 1$, and Assumption 3 with $\mu = \lambda$ [10]. Note that the dependence of L_l on d does not affect our analysis. Furthermore, recent research concerning the low-rank phenomenon observed in neural networks [64]–[66] and known relations between low-rankness and Hölder smoothness [51], [62] suggest that neural networks may satisfy Assumption 2 in practice as well.

Algorithm 1 PPI-GD

Input: Training data $\mathcal{D} = \{\mathbf{x}^{(i)} : i \in [n]\} \subset \mathcal{X} = [0,1]^d$
Input: Oracle access to $\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta})$ at any given $\mathbf{x} \in \mathcal{X}$ and $\boldsymbol{\theta} \in \mathbb{R}^p$
Output: $\boldsymbol{\theta}^*$ satisfying $|F(\boldsymbol{\theta}^*) - F^*| \le \varepsilon$

Initial parameters $\boldsymbol{\theta}^{(0)} \leftarrow$ some vector $\in \mathbb{R}^p$;
Set number of iterations $T \in \mathbb{Z}_{>0}$ according to (19);
Choose the smallest $m \in \mathbb{Z}_{>0}$ that satisfies (15);
Construct uniform grid $\mathcal{G}_m^d$ according to (5);
Divide $\mathcal{X}$ into patches $\mathcal{P}$ as in (6);
Precompute coefficients of $\psi_{\mathbf{s},\mathbf{y}}$ for all $\mathbf{s} \in [\lceil m/\ell \rceil]^d$ and $\mathbf{y} \in \boldsymbol{\mathcal{G}}(P_{\mathbf{s}})$ according to (8);
for $t \in [T]$ **do**
  Make m^d oracle calls to get $\{\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{u};\boldsymbol{\theta}^{(t-1)}) : \mathbf{u} \in \mathcal{G}_m^d\}$;
  for all $\mathbf{x}^{(i)} \in \mathcal{D}$ **do**
    Find the patch $P_{\mathbf{s}}$ where $\mathbf{x}^{(i)}$ belongs;
    Compute the estimate $\widehat{\boldsymbol{\nabla}_{\boldsymbol{\theta}}} f(\mathbf{x}^{(i)};\boldsymbol{\theta}^{(t-1)})$ according to (7) using the queried $\{\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{u};\boldsymbol{\theta}^{(t-1)}) : \mathbf{u} \in \mathcal{G}_m^d\}$ and precomputed $\psi_{\mathbf{s},\mathbf{y}}$;
  end for
  $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \frac{\alpha}{n} \sum_{i=1}^n \widehat{\boldsymbol{\nabla}_{\boldsymbol{\theta}}} f(\mathbf{x}^{(i)};\boldsymbol{\theta}^{(t-1)})$;
end for
return $\boldsymbol{\theta}^{(T)}$;

III. Main Results and Discussion

A. Algorithm

The standard method to find F^* is through gradient descent (GD), which repeats the following iteration until convergence starting from the initial iterate $\boldsymbol{\theta}^{(0)}$:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \alpha_t \nabla_{\theta} F(\boldsymbol{\theta}^{(i-1)}) ,$$

The key idea of PPI-GD is to approximate each $\nabla_{\boldsymbol{\theta}} f(\mathbf{x}^{(i)};\boldsymbol{\theta}^{(i-1)})$ by interpolating over grid points across the data space. If the number of grid points is less than the number of data points n, each iteration of PPI-GD will use fewer oracle calls compared to GD.

First, let the uniform grid $\mathcal{G}_m^d \subset [0,1]^d$ with m points along each axis be defined as follows:

$$\mathcal{G}_m^d \triangleq \left\{ \frac{1}{m}\mathbf{u} : \mathbf{u} \in [\![0, m-1]\!]^d \right\} , \quad (5)$$

with $\mathcal{G}_m = \mathcal{G}_m^1$. At each iteration, m^d oracle calls are made to query the gradient of f at each grid point in $\mathcal{G}_m^d$.

PPI-GD divides the data space $[0,1]^d$ into hypercube patches $\mathcal{P}$ which cover the space completely but may overlap. Each patch will contain the same number of grid points $(\ell+1)^d$, but one grid point might belong to more than one patch. The number of grid points in each patch is chosen so that the polynomial interpolant of order ℓ is well defined.

Let $\mathcal{I}$ be a set of intervals as follows:

$$\mathcal{I} \triangleq \left\{ \frac{\ell}{m}[u-1, u] : u \in \left[\left\lceil \frac{m}{\ell} \right\rceil - 1 \right] \right\} \cup \left\{ \left[1 - \frac{\ell}{m}, 1 \right] \right\} \\ = \left\{ I_1, \ldots, I_{\lceil m/\ell \rceil} \right\}$$

where each I_i are the intervals in $\mathcal{I}$ indexed in order of increasing infimum. Then, the set of patches $\mathcal{P}$ can be defined as the Cartesian power

$$\mathcal{P} \triangleq \mathcal{I}^d = \left\{ P_{\mathbf{s}} = \bigtimes_{i\in[d]} I_{s_i} : \mathbf{s} \in \left[\left\lceil \frac{m}{\ell} \right\rceil\right]^d \right\}. \tag{6}$$

Let $\mathcal{G}(I_i) = \mathcal{G}_m \cap I_i$ be the grid points belonging in each interval I_i and $\boldsymbol{\mathcal{G}}(P_{\mathbf{s}}) = \mathcal{G}_m^d \cap P_{\mathbf{s}}$ the grid points in each patch $P_{\mathbf{s}}$. Note that $|\mathcal{G}(I_i)| = \ell + 1$ for all $I_i \in \mathcal{I}$ and $|\boldsymbol{\mathcal{G}}(P_{\mathbf{s}})| = (\ell+1)^d$ for all $P_{\mathbf{s}} \in \mathcal{P}$. In addition,

$$\sup_{x\in I_i} \left(\prod_{y\in\mathcal{G}(I_i)} (x-y) \right) \le \prod_{j=1}^{\ell} \frac{j}{m} = \frac{\ell!}{m^\ell}.$$

For each patch $P_{\mathbf{s}}$ and $i \in [p]$, a unique interpolating tensor product polynomial of degree ℓ along each dimension exists:

$$\exists \rho_{\mathbf{s},i}(\mathbf{x}) = \sum_{\mathbf{t}\in[\![0,\ell]\!]^d} a_{\mathbf{t}} \mathbf{x}^{\mathbf{t}}, \ \ \forall \mathbf{x} \in \boldsymbol{\mathcal{G}}(P_{\mathbf{s}}), \ \ \rho_{\mathbf{s},i}(\mathbf{x}) = \frac{\partial}{\partial \theta_i} f(\mathbf{x};\boldsymbol{\theta}).$$

This interpolant can be constructed with the tensor product Lagrange basis:

$$\rho_{\mathbf{s},i}(\mathbf{x}) = \sum_{\mathbf{y}\in\boldsymbol{\mathcal{G}}(P_{\mathbf{s}})} \frac{\partial}{\partial\theta_i} f(\mathbf{y};\boldsymbol{\theta}) \psi_{\mathbf{s},\mathbf{y}}(\mathbf{x}), \tag{7}$$

where

$$\psi_{\mathbf{s},\mathbf{y}}(\mathbf{x}) \triangleq \prod_{j=1}^{d} \left(\prod_{z\in\mathcal{G}(I_{s_j})\setminus\{y_j\}} \frac{x_j - z}{y_j - z} \right). \tag{8}$$

It is straightforward to show that such a construction produces a valid interpolant, and a constraint counting argument can prove its uniqueness. Note that each $\psi_{\mathbf{s},\mathbf{y}}(\mathbf{x})$ does not depend on f or $\boldsymbol{\theta}$, so they can be precomputed once m, d, and ℓ are known.

Thus, the interpolant can be evaluated by computing a linear combination of $\{\boldsymbol{\nabla}_{\boldsymbol{\theta}} f(\mathbf{u};\boldsymbol{\theta}) : \mathbf{u} \in \mathcal{G}_m^d\}$ with precomputed weights. The computed values will approximate the true gradient at each data point $\mathbf{x}$, and their mean will approximate $\boldsymbol{\nabla}_{\boldsymbol{\theta}} F$. At the end of each iteration, this approximation can be used instead of the true gradient to perform the gradient descent update, where α is the (fixed) step size:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \frac{\alpha}{n} \sum_{j=1}^{n} \left[\rho_{\mathbf{s},i}\left(\mathbf{x}^{(j)}\right) : i \in [p] \right]^{\mathsf{T}}.$$

This process is summarized in Algorithm 1.

We remark that due to the *curse of dimensionality*, a naïve implementation of PPI-GD requires $O(pm^d)$ space to store the gradients of f at the grid points, which becomes intractable as d increases. One approach to alleviate this is to preprocess the data with *dimensionality reduction* [67] methods before running PPI-GD. Another practical concern is that the information required to choose optimal values for hyperparameters such as m and ℓ may not be known analytically. Nevertheless, just as with any other hyperparameter, *hyperparameter optimization* methods [68] can be used to empirically find suitable values.

Notwithstanding the remarks above, the primary objective of our work is theoretical, and we focus on establishing the oracle complexity of PPI-GD in the following sections. To this end, we first derive error bounds on the aforementioned polynomial interpolation process in Section III-B, because our subsequent oracle complexity analysis depends on the accuracy of the interpolated gradients. After establishing the interpolation error bounds, we discuss the oracle complexity of PPI-GD in Section III-C.

B. Interpolation Error Bounds

To begin our discussion of polynomial interpolation error, we introduce some preliminary notions and conventions used throughout this analysis. Let $\mathcal{G} = \{u_i\}_{i=1}^{n+1} \subset [a,b]$ be a grid of evenly spaced abscissae on $[a,b]$, namely, $a = u_1 < \cdots < u_{n+1} = b$ and $u_{i+1} - u_i = \Delta u = (b-a)/n$ for all $i \in [n]$. (Note that n has no relation to the size of the training set in the context of our interpolation error analysis.) For each $i \in [n+1]$, let $\phi_i : [a,b] \to \mathbb{R}$ be the univariate polynomial interpolation basis function (cf. [69, Corollary 2]) satisfying

$$\forall i' \in [n+1], \ \ \phi_i(u_{i'}) = \begin{cases} 1, & \text{if } i = i', \\ 0, & \text{otherwise}, \end{cases}$$

which in the Lagrange basis is the degree-n polynomial

$$\phi_i(x) = \prod_{i'\neq i} \frac{x - u_{i'}}{u_i - u_{i'}}.$$

Given a function $g : [a,b]^d \to \mathbb{R}$, let $\mathrm{P}g : [a,b]^d \to \mathbb{R}$ be the d-variate *total interpolant* of g at the lattice of abscissae $\mathcal{G}^d$, given by

$$\mathrm{P}g(x_1,\ldots,x_d) = \sum_{i_1=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} g(u_{i_1},\ldots,u_{i_d}) \prod_{s=1}^{d} \phi_{i_s}(x_s), \tag{9}$$

where we elide the dependence on n and $\mathcal{G}$ in the notation P for simplicity. For any $j \in [d]$, let $\mathrm{P}_j g : [a,b]^d \to \mathbb{R}$ be the *univariate partial interpolant* of g with respect to x_j, given by

$$\mathrm{P}_j g(x_1,\ldots,x_d) = \sum_{i=1}^{n+1} g(x_1,\ldots,x_{j-1},u_i,x_{j+1},\ldots,x_d)\, \phi_i(x_j).$$

This definition naturally generalizes to *multivariate partial interpolants* by taking multiple subscripts and computing linear combinations over the tensor product of basis interpolants in the subscripted dimensions:[1]

$$\mathrm{P}_{j_1,\ldots,j_r} g(x_1,\ldots,x_d) = \sum_{i_1=1}^{n+1} \cdots \sum_{i_r=1}^{n+1} g(x_1,\ldots,u_{i_s},\ldots,x_d) \prod_{s=1}^{r} \phi_{i_s}(x_{j_s}). \tag{10}$$

When all d dimensions are included in the subscript, we recover the total interpolant $\mathrm{P}_{1,\ldots,d} g = \mathrm{P}g$. We mention two

[1]The r-times tensor product of the vector space of degree-n univariate polynomials is a *strict subset* of the vector space of degree-rn r-variate polynomials.

remarks regarding the definitions above. Firstly, forming a partial interpolant $\mathrm{P}_{j_1,\ldots,j_r}g$ according to (10) and evaluating its value at some point $(v_1,\ldots,v_d) \in [a,b]^d$ is equivalent to first performing partial application on g to obtain $g(x_{j_1},\ldots,x_{j_r}) = g(v_1,\ldots,x_{j_s},\ldots,v_d)$, forming the total interpolant $\mathrm{P}g$ according to (9), and then evaluating the total interpolant at $(v_{j_1},\ldots,v_{j_r})$. Hence, interpolation error bounds for univariate functions in the literature apply *ipso facto* to univariate partial interpolants of multivariate functions. Secondly, evaluating a multivariate interpolant of the form (9) at some point $(v_1,\ldots,v_d) \in [a,b]^d$ may equivalently be done by successively forming and evaluating univariate partial interpolants (cf. [70, Section 3.6.2]):

$$
\begin{aligned}
&\mathrm{P}g(v_1,\ldots,v_d)\\
&\overset{(a)}{=} \sum_{i_1=1}^{n+1}\cdots\sum_{i_d=1}^{n+1} g(u_{i_1},\ldots,u_{i_d})\prod_{s=1}^{d}\phi_{i_s}(v_s)\\
&\overset{(b)}{=} \sum_{i_d=1}^{n+1}\left(\cdots\sum_{i_2=1}^{n+1}\left(\sum_{i_1=1}^{n+1}\right.\right.\\
&\qquad \left.\left. g(u_{i_1},\ldots,u_{i_d})\,\phi_{i_1}(v_1)\right)\phi_{i_2}(v_2)\cdots\right)\phi_{i_d}(v_d)\\
&\overset{(\dagger_1)}{=} \sum_{i_d=1}^{n+1}\left(\cdots\sum_{i_2=1}^{n+1}\mathrm{P}_1 g(v_1,u_{i_2},\ldots,u_{i_d})\,\phi_{i_2}(v_2)\cdots\right)\phi_{i_d}(v_d)\\
&\overset{(\dagger_2)}{=} \sum_{i_d=1}^{n+1}\left(\cdots \mathrm{P}_2\mathrm{P}_1 g(v_1,v_2,u_{i_3},\ldots,u_{i_d})\cdots\right)\phi_{i_d}(v_d)\\
&\;\;\vdots\\
&\overset{(\dagger_d)}{=} \mathrm{P}_d\cdots\mathrm{P}_1 g(v_1,\ldots,v_d)\,,
\end{aligned}
\tag{11}
$$

where (a) holds by (9) and (b) holds by reversing the order of the summations and applying the distributive property. Algorithmically, we initialize a d-dimensional array of values of g evaluated at the abscissae $\mathcal{G}^d$. In step $(\dagger_1)$, we use this array to construct a univariate interpolant along the x_1 axis for each $(u_{i_2},\ldots,u_{i_d}) \in \mathcal{G}^{d-1}$, and evaluate all the interpolants at v_1 to obtain a $(d-1)$-dimensional array. In step $(\dagger_2)$, we use this new array to construct a univariate interpolant along the x_2 axis for each $(u_{i_3},\ldots,u_{i_d}) \in \mathcal{G}^{d-2}$, and evaluate all the interpolants at v_2 to obtain a $(d-2)$-dimensional array. Repeating this process until step $(\dagger_d)$ yields the desired result.

For any sequence $j_1,\ldots,j_r \in [d]$ of dimensions, let $\mathrm{E}_{j_1,\ldots,j_r}g : [a,b]^d \to \mathbb{R}$ be the *iterated interpolation error* given by

$$
\mathrm{E}_{j_1,\ldots,j_r}g = \begin{cases} \mathrm{P}_{j_1}g - g\,, & \text{if } r = 1\,,\\ \mathrm{P}_{j_r}\mathrm{E}_{j_1,\ldots,j_{r-1}}g - \mathrm{E}_{j_1,\ldots,j_{r-1}}g\,, & \text{otherwise}\,. \end{cases}
\tag{12}
$$

Note that multiple subscripts of E in (12) specify *successive* iterations of *uni*variate partial interpolation, each on the E term from the preceding iteration, while multiple subscripts of P in (10) specify *one* iteration of *multi*variate partial interpolation on g.[2]

With these preliminaries established, we present the main result of this subsection, an upper bound on the uniform norm interpolation error for d-variate functions.

Theorem 1 (Polynomial interpolation error bound)**.** *Let $g : [a,b]^d \to \mathbb{R}$ be a function in the Hölder class $\mathcal{H}(\ell, L_h)$. Assume that $\ell \geq d$ and $|\mathcal{G}| = n+1 \geq \ell$. Then, the error in the total interpolant of g satisfies*

$$
\left\|\mathrm{P}g - g\right\|_\infty \leq L_h\,(\Delta u)^\ell\left(\frac{2^n}{n}+1\right)^d\,.
$$

Theorem 1 is proved in Section IV. We split our argument into three steps, delineated by the following propositions which are also proved in Section IV. Firstly, we decompose the total interpolation error into a sum of iterated interpolation error terms, with one iterated term for each non-empty subset of the d variables.

Proposition 2. *For any continuous function $g : [a,b]^d \to \mathbb{R}$, the error in the total interpolant of g satisfies*

$$
\left\|\mathrm{P}g - g\right\|_\infty \leq \sum_{\substack{\mathcal{K}\in 2^{[d]}:\\ \mathcal{K}\neq\emptyset}} \left\|\mathrm{E}_{\mathcal{K}}g\right\|_\infty\,.
$$

To establish the intuition behind Proposition 2, we consider an example where g is a function of three variables x_1, x_2, and x_3. Figure 1 illustrates the decomposition in this case, and the steps labeled with letters below correspond to edges in the figure. Using the triangle inequality, we upper-bound $\left\|\mathrm{P}g - g\right\|_\infty = \left\|\mathrm{P}_{1,2,3}g - g\right\|_\infty$ as a sum of the magnitudes of three partial interpolants of error terms, where each partial interpolant is subscripted with a different suffix of the variables $[x_1,x_2,x_3]$:

$$
\begin{aligned}
&\left\|\mathrm{P}_{1,2,3}g - g\right\|_\infty\\
&\leq \left\|\mathrm{P}_{1,2,3}g - \mathrm{P}_{2,3}g\right\|_\infty + \left\|\mathrm{P}_{2,3}g - \mathrm{P}_3 g\right\|_\infty + \left\|\mathrm{P}_3 g - g\right\|_\infty\\
&\overset{(a)}{=} \left\|\mathrm{P}_{2,3}\mathrm{E}_1 g\right\|_\infty + \left\|\mathrm{P}_3\mathrm{E}_2 g\right\|_\infty + \left\|\mathrm{E}_3 g\right\|_\infty\,.
\end{aligned}
\tag{13}
$$

The partial interpolants $\mathrm{P}_{2,3}g$ and $\mathrm{P}_3 g$ in the triangle inequality are analogous to *blended interpolants* from the bicubic spline literature [71]. Next, we split each term in (13) into the sum of an iterated interpolation error of g and another term to be recursively decomposed:

$$
\begin{aligned}
\left\|\mathrm{P}_{2,3}\mathrm{E}_1 g\right\|_\infty &\overset{(b)}{\leq} \left\|\mathrm{E}_1 g\right\|_\infty + \left\|\mathrm{P}_{2,3}\mathrm{E}_1 g - \mathrm{E}_1 g\right\|_\infty\,, \qquad (14)\\
\left\|\mathrm{P}_3\mathrm{E}_2 g\right\|_\infty &\overset{(c)}{\leq} \left\|\mathrm{E}_2 g\right\|_\infty + \left\|\mathrm{P}_3\mathrm{E}_2 g - \mathrm{E}_2 g\right\|_\infty\,,\\
\left\|\mathrm{E}_3 g\right\|_\infty &= \left\|\mathrm{E}_3 g\right\|_\infty\,.
\end{aligned}
$$

Next, we recursively decompose $\left\|\mathrm{P}_{2,3}\mathrm{E}_1 g - \mathrm{E}_1 g\right\|_\infty$. Similarly to (13), we obtain

$$
\left\|\mathrm{P}_{2,3}\mathrm{E}_1 g - \mathrm{E}_1 g\right\|_\infty \overset{(d)}{\leq} \left\|\mathrm{P}_3\mathrm{E}_{1,2} g\right\|_\infty + \left\|\mathrm{E}_{1,3} g\right\|_\infty\,,
$$

and similarly to (14), we obtain

$$
\left\|\mathrm{P}_3\mathrm{E}_{1,2} g\right\|_\infty \overset{(e)}{\leq} \left\|\mathrm{E}_{1,2} g\right\|_\infty + \left\|\mathrm{P}_3\mathrm{E}_{1,2} g - \mathrm{E}_{1,2} g\right\|_\infty\,,
$$

[2]Equivalently, multiple subscripts of P specify successive iterations of univariate partial interpolation, each on the P term from the preceding iteration, as per the discussion in (11).

$$\|\mathrm{E}_{1,3}g\|_\infty = \|\mathrm{E}_{1,3}g\|_\infty \ .$$

Lastly, by definition of E, we have

$$\|\mathrm{P}_3\mathrm{E}_2 g - \mathrm{E}_2 g\|_\infty \overset{(f)}{=} \|\mathrm{E}_{2,3}g\|_\infty \ ,$$
$$\|\mathrm{P}_3\mathrm{E}_{1,2} g - \mathrm{E}_{1,2} g\|_\infty \overset{(g)}{=} \|\mathrm{E}_{1,2,3}g\|_\infty \ .$$

In a nutshell, by repeatedly applying the logic of steps (13) and (14), the recursive decomposition above generates an iterated error term $\|\mathrm{E}_{\mathcal{K}}g\|_\infty$ for every non-empty subset of variables $\mathcal{K}$. Hence, this process may be modeled with a recursion tree, and using strong induction over the tree proves Proposition 2. The power of d scaling within the power set $2^{[d]}$ is expected due to the curse of dimensionality.

Secondly, we establish an error bound for univariate polynomial interpolation, based on judicious usage of a Newton basis representation of the interpolant followed by application of the mean value theorem for divided differences:

Proposition 3. *Consider any function $h : [a,b] \to \mathbb{R}$ with $h \in \mathcal{C}^q([a,b])$. Assume $|\mathcal{G}| = n+1 \geq q$. Then, the error in the (degree-n) univariate interpolant of h satisfies*

$$\|\mathrm{P}h - h\|_\infty \leq \frac{2^{n+1-q}}{n} (\Delta u)^q \left\|h^{(q)}\right\|_\infty \ .$$

Thirdly, we upper-bound the magnitude of the iterated error terms from Proposition 2 by repeatedly applying Proposition 3 to bound the effect of each iteration:

Proposition 4. *Consider any function $g : [a,b]^d \to \mathbb{R}$ with $g \in \mathcal{C}^\ell([a,b]^d)$. Assume $\ell \geq d$ and $|\mathcal{G}| = n+1 \geq \ell$. For notational convenience, let $\alpha_q = 2^{n+1-q}/n$ for any $q \in \mathbb{Z}_{>0}$. Then, for any distinct $j_1, \ldots, j_r \in [d]$,*

$$\|\mathrm{E}_{j_1,\ldots,j_r}g\|_\infty \leq \alpha_{\ell-r+1}\, \alpha_1^{r-1}\, (\Delta u)^\ell \left\| \frac{\partial^\ell g}{\partial x_{j_1}^{\ell-r+1} \partial x_{j_2} \cdots \partial x_{j_r}} \right\|_\infty .$$

Each application of Proposition 3 gives rise to one of the α terms above. The proof of Proposition 4 also utilizes the following lemma, which states the equality of interchanging partial differentiation and partial interpolation and is proved in Section IV.

Lemma 5. *For any distinct $j_1, \ldots, j_r, k \in [d]$ and any function $g : [a,b]^d \to \mathbb{R}$ differentiable with respect to x_k, we have*

$$\frac{\partial \mathrm{P}_{j_1,\ldots,j_r} g}{\partial x_k} = \mathrm{P}_{j_1,\ldots,j_r}\left(\frac{\partial g}{\partial x_k}\right).$$

Finally, combining the intermediate results above and using the Hölder smoothness of g proves Theorem 1. As previously mentioned, we defer the technical details to Section IV.

C. Oracle Complexity

Now that we have analyzed the error bounds on multivariate polynomial interpolation in a general setting, we apply those results to our PPI-GD algorithm. First, we get the following result on the error of our approximate gradient.

Lemma 6. *For any constant $\delta > 0$, if the number of grid points m^d is large enough that*

$$m \geq \left(\frac{L_h}{\delta} \left(\frac{2^\ell}{\ell} + 1 \right)^d \right)^{1/\ell} \tag{15}$$

holds, then for all $i \in [p]$ and $\boldsymbol{\theta} \in \mathbb{R}^p$,

$$\sup_{\mathbf{x} \in \mathcal{X}} \left| \widehat{\frac{\partial}{\partial \theta_i}} f(\mathbf{x}; \boldsymbol{\theta}) - \frac{\partial}{\partial \theta_i} f(\mathbf{x}; \boldsymbol{\theta}) \right| \leq \delta.$$

Lemma 6 is proved in Section V. Then, we use the aforementioned result on interpolation error to derive the oracle complexity of our proposed PPI-GD method in the strongly convex setting.

Theorem 7 (Oracle complexity of PPI-GD in strongly convex setting)**.** *Suppose Assumptions 1 through 3 hold. Given some (small) accuracy $\varepsilon > 0$, let m be the smallest positive integer that satisfies inequality (15) in Lemma 6 with δ given by (20). Let the step size be $\alpha = 1/L_l$. If $\ell \geq d \geq 2$ and $d \leq \log^{-1/2}(2)\log^\gamma(n) - 1/2$ for some constant $0 < \gamma < 1/2$, the first-order oracle complexity of PPI-GD (to obtain an ε-approximate solution) is bounded by*

$$\Gamma(\text{PPI-GD}) \leq C_1 \exp\left(\log^{2\gamma}(n)\right) \left(\frac{p}{\varepsilon}\right)^{d/2\ell} \log\left(\frac{p}{\varepsilon}\right)$$

where the constant C_1 only depends on μ, L_l, ℓ, and L_h.

Theorem 7 is proved in Section V. To obtain the final scaling with p in the proof, we assume that the parameters $\boldsymbol{\theta}$ belong in a hypercube with constant edge length (that does not depend on d). This assumption is used to establish a relation between the parameter space dimension p and the $\mathcal{L}^2$-distance between the initial choice of parameters $\boldsymbol{\theta}^{(0)}$ and minimizer $\boldsymbol{\theta}^*$.

Next, we present the following proposition which illustrates a regime where PPI-GD outperforms GD, SGD, SVRG, and MBGD in terms of oracle complexity for strongly convex loss.

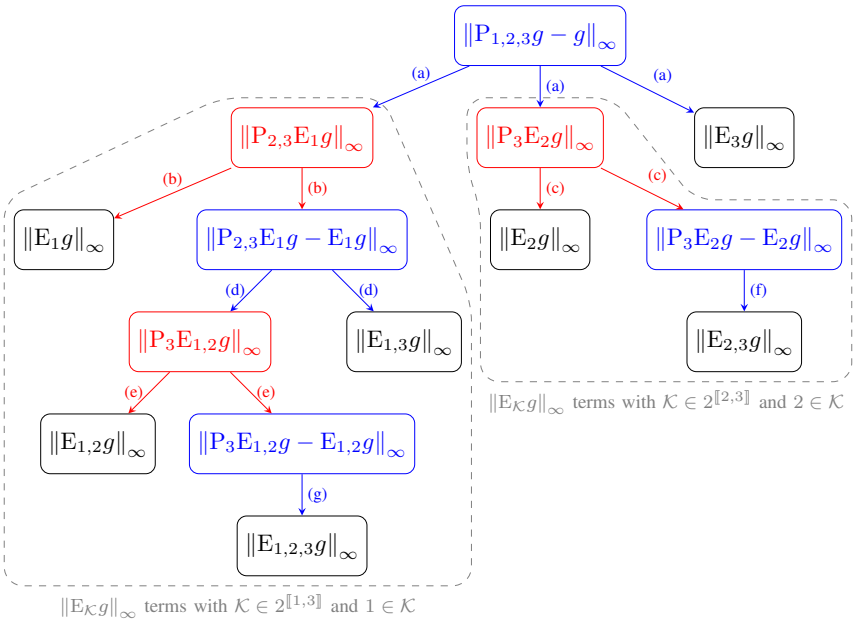
Proposition 8. *Under the same assumptions as Theorem 7, if $p/\varepsilon = O(n^\beta)$ for some $\beta > 0$, GD, SGD, SVRG, MBGD, and PPI-GD exhibit the following oracle complexity (expressed in Landau notation):*

Algorithm	*Oracle complexity*
GD	$\tilde{O}(n)$
SGD	$\tilde{O}(n^\beta)$
SVRG	$\tilde{O}(n)$
MBGD	$\tilde{O}(n^\beta)$
PPI-GD	$\tilde{O}(n^{(d\beta)/(2\ell)})$

Proposition 8 is proved in Section V. The regime $p/\varepsilon = O(n^\beta)$ studied here encompasses a wide range of common settings, including the case where p and $1/\varepsilon$ are some positive power of n. For example, in machine learning contexts where overparameterization ($p > n$) can often be beneficial [72], it is natural to consider a regime of $p = O(n^b)$ (where $b \geq 1$) and $\varepsilon = \Theta(1/\sqrt{n})$, with the error bound matching the scaling of the standard error.

Note that Proposition 8 implies the oracle complexity of PPI-GD to be $\tilde{O}(n^{\beta/2})$ since $\ell \geq d$ by assumption. Thus, in

Fig. 1: Recursion tree for the three-variable case of Proposition 2, showing how we decompose total interpolation error $\|\mathrm{P}g - g\|_\infty = \|\mathrm{P}_{1,2,3}g - g\|_\infty$ into a sum of iterated error terms $\|\mathrm{E}_\mathcal{K} g\|_\infty$, with one such term for each non-empty subset of variables. Letters along edges correspond to labeled steps from the proof sketch in Section III-B. At blue nodes (e.g., (13)), the decomposition process produces partial interpolants of iterated errors. At red nodes (e.g., (14)), the process produces an iterated error term (black child) and another term to be recursively decomposed (blue child). The partial interpolation operator P in blue nodes is subscripted with a suffix of the variables $[x_1, x_2, x_3]$, and blue nodes with $\mathrm{P}_{[\![t,3]\!]}$ have $3-t$ red children and 1 black child.



this regime, PPI-GD always outperforms the other methods in terms of asymptotic oracle complexity when $\beta < 2$. In addition, increasing ℓ can reduce the exponent $(d\beta)/(2\ell)$ to any arbitrary positive value, which implies that sufficient smoothness in data allows PPI-GD to outperform the other methods for *any* $\beta > 0$.

Although the strongly convex setting is important and well studied in the optimization literature, strong convexity can be a stringent requirement. Thus, we now shift our attention to the non-convex case. Note that the step size must be reduced from $1/L_l$ to $1/(4L_l)$ in this setting.

Theorem 9 (Oracle complexity of PPI-GD in non-convex setting)**.** *Given some (small) $\varepsilon > 0$, let m be the smallest positive integer that satisfies inequality (15) in Lemma 6 with $\delta = \varepsilon/(2\sqrt{p})$. Let the step size be $\alpha = 1/(4L_l)$. If $\ell \geq d \geq 2$ and $d \leq \log^{-1/2}(2)\log^{\gamma}(n) - 1/2$ for some constant $0 < \gamma < 1/2$, the first-order oracle complexity of PPI-GD (to reach an ε-stationary point) is bounded by*

$$\Gamma(\text{PPI-GD}) \leq C_2 \exp\left(\log^{2\gamma}(n)\right)\left(\frac{p}{\varepsilon^2}\right)^{1+d/(2\ell)}$$

where the constant C_2 only depends on L_l, ℓ, and L_h.

An analog to Proposition 8 immediately follows.

Proposition 10. *Under the same assumptions as Theorem 9, if $p/\varepsilon^2 = O(n^\beta)$ for some $\beta > 0$, GD, SGD, and PPI-GD exhibit the following oracle complexity (expressed in Landau notation):*

Algorithm	*Oracle complexity*
GD	$\tilde{O}(n^{1+\beta})$
SGD	$\tilde{O}(n^{2\beta})$
PPI-GD	$\tilde{O}(n^{(1+d/(2\ell))\beta})$

Theorem 9 and Proposition 10 are proved in Section V. All our remarks regarding Proposition 8 also apply to Proposition 10: under our assumptions, PPI-GD beats the other methods when $\beta < 2$, and sufficient smoothness (i.e., a large enough ℓ) can arbitrarily relax this bound.

We conclude this section with some remarks on the implications of our analysis by reexamining the oracle complexities summarized in Table I. Looking at each factor in isolation, we see that PPI-GD's oracle complexity $O(\exp(\log^{2\gamma}(n))(p/\varepsilon)^{d/(2\ell)}\log(p/\varepsilon))$ for strongly convex loss has better scaling in n than GD's $O(n\log(p/\varepsilon))$ (neglecting other factors like p and ε), and better scaling in p/ε than SGD's $O(p/\varepsilon)$. Likewise, in the non-convex setting, PPI-GD's $O(\exp(\log^{2\gamma}(n))(p/\varepsilon^2)^{1+d/(2\ell)})$ scales better in n than GD's $O(n(p/\varepsilon^2))$, and scales better in p/ε^2 than SGD's $O((p/\varepsilon^2)^2)$. This observation motivates Propositions 8 and 10, which show that PPI-GD outperforms the other methods in the important regimes of $p/\varepsilon = O(n^\beta)$ (for strongly convex loss) or $p/\varepsilon^2 = O(n^\beta)$ (for non-convex loss) for any $\beta > 0$. While this holds true for LPI-GD as well, our algorithm does not require as strong a bound on d, relaxing the curse of dimensionality effect in LPI-GD significantly.

IV. Proofs of Interpolation Error Bounds

In this section, we prove Theorem 1, which upper-bounds the uniform norm total interpolation error for d-variate functions. As discussed in Section III-B, we begin by proving Proposition 2, which decomposes the total error into a sum of iterated interpolation error terms.

Proof of Proposition 2. We will show by induction that for all $t \in [d]$,

$$\|\mathrm{P}_{t,\dots,d} g - g\|_\infty \le \sum_{\substack{\mathcal{K} \in 2^{[\![t,d]\!]}: \\ \mathcal{K} \neq \emptyset}} \|\mathrm{E}_{\mathcal{K}} g\|_\infty \ . \tag{16}$$

Base case: $t = d$. By inspection, we have

$$\|\mathrm{P}_d g - g\|_\infty \overset{(a)}{=} \|\mathrm{E}_d g\|_\infty = \sum_{\substack{\mathcal{K} \in 2^{\{d\}}: \\ \mathcal{K} \neq \emptyset}} \|\mathrm{E}_{\mathcal{K}} g\|_\infty \ ,$$

where (a) holds by the definition of E.

Inductive step: $t < d$. We have

$$\begin{aligned}
&\|\mathrm{P}_{t,\dots,d} g - g\|_\infty \\
&\overset{(a)}{=} \max_{(x_1,\dots,x_d)\in[a,b]^d} |\mathrm{P}_{t,\dots,d} g(x_1,\dots,x_d) - g(x_1,\dots,x_d)| \\
&\overset{(b)}{=} \max_{(x_1,\dots,x_d)\in[a,b]^d} \Bigg| \\
&\quad \sum_{i_t=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} g(x_1,\dots,x_{t-1},u_{i_t},\dots,u_{i_d}) \prod_{s=t}^{d} \phi_{i_s}(x_s) \\
&\quad - g(x_1,\dots,x_d) \Bigg| \\
&\overset{(c)}{=} \max_{(x_1,\dots,x_d)\in[a,b]^d} \Bigg| \sum_{r=t}^{d} \Bigg(\\
&\quad \sum_{i_r=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} g(x_1,\dots,x_{r-1},u_{i_r},\dots,u_{i_d}) \prod_{s=r}^{d} \phi_{i_s}(x_s) \\
&\quad - \sum_{i_{r+1}=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} g(x_1,\dots,x_r,u_{i_{r+1}},\dots,u_{i_d}) \prod_{s=r+1}^{d} \phi_{i_s}(x_s) \Bigg) \Bigg| ,
\end{aligned}$$

where (a) holds by definition of uniform norm and max may be used due to the continuity of g and the compactness of $[a,b]^d$, (b) holds by definition of P, we assume in (c) and all subsequent steps that the iterated sum $\sum_{i_{r+1}} \cdots \sum_{i_d}$ reduces to a single copy of the summand when $r = d$, and (c) holds because the sum over $r \in [\![t,d]\!]$ telescopes. Proceeding onwards, we have

$$\begin{aligned}
&\|\mathrm{P}_{t,\dots,d} g - g\|_\infty \\
&\overset{(d)}{\le} \max_{(x_1,\dots,x_d)\in[a,b]^d} \sum_{r=t}^{d} \Bigg| \\
&\quad \sum_{i_r=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} g(x_1,\dots,x_{r-1},u_{i_r},\dots,u_{i_d}) \prod_{s=r}^{d} \phi_{i_s}(x_s) \\
&\quad - \sum_{i_{r+1}=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} g(x_1,\dots,x_r,u_{i_{r+1}},\dots,u_{i_d}) \prod_{s=r+1}^{d} \phi_{i_s}(x_s) \Bigg| \\
&= \max_{(x_1,\dots,x_d)\in[a,b]^d} \sum_{r=t}^{d} \Bigg| \sum_{i_{r+1}=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} \\
&\quad \Bigg(\sum_{i_r=1}^{n+1} g(x_1,\dots,x_{r-1},u_{i_r},\dots,u_{i_d})\, \phi_{i_r}(x_r) \\
&\quad - g(x_1,\dots,x_r,u_{i_{r+1}},\dots,u_{i_d}) \Bigg) \prod_{s=r+1}^{d} \phi_{i_s}(x_s) \Bigg| \\
&\overset{(e)}{=} \max_{(x_1,\dots,x_d)\in[a,b]^d} \sum_{r=t}^{d} \Bigg| \sum_{i_{r+1}=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} \\
&\quad \Big(\mathrm{P}_r g(x_1,\dots,x_r,u_{i_{r+1}},\dots,u_{i_d}) \\
&\quad - g(x_1,\dots,x_r,u_{i_{r+1}},\dots,u_{i_d}) \Big) \prod_{s=r+1}^{d} \phi_{i_s}(x_s) \Bigg| \\
&\overset{(f)}{=} \max_{(x_1,\dots,x_d)\in[a,b]^d} \sum_{r=t}^{d} \Bigg| \sum_{i_{r+1}=1}^{n+1} \cdots \sum_{i_d=1}^{n+1} \\
&\quad \mathrm{E}_r g(x_1,\dots,x_r,u_{i_{r+1}},\dots,u_{i_d}) \prod_{s=r+1}^{d} \phi_{i_s}(x_s) \Bigg| \\
&\overset{(g)}{=} \max_{(x_1,\dots,x_d)\in[a,b]^d} \sum_{r=t}^{d} |\mathrm{P}_{r+1,\dots,d} \mathrm{E}_r g(x_1,\dots,x_d)| \ ,
\end{aligned}$$

where (d) holds by the triangle inequality, (e) holds by definition of P, (f) holds by definition of E, we assume in (g) and all subsequent steps that $\mathrm{P}_{r+1,\dots,d}$ is the identity operation when $r = d$, and (g) holds by definition of P. We note that steps (c) through (g) correspond to the decomposition of blue nodes in Figure 1 and, for example, (13) in the proof sketch in Section III-B. Proceeding onwards, we have

$$\begin{aligned}
&\|\mathrm{P}_{t,\dots,d} g - g\|_\infty \\
&\overset{(h)}{\le} \sum_{r=t}^{d} \max_{(x_1,\dots,x_d)\in[a,b]^d} |\mathrm{P}_{r+1,\dots,d} \mathrm{E}_r g(x_1,\dots,x_d)| \\
&\overset{(i)}{=} \sum_{r=t}^{d} \|\mathrm{P}_{r+1,\dots,d} \mathrm{E}_r g\|_\infty \\
&\overset{(j)}{\le} \sum_{r=t}^{d} \|\mathrm{E}_r g\|_\infty + \sum_{r=t}^{d-1} \|\mathrm{P}_{r+1,\dots,d} \mathrm{E}_r g - \mathrm{E}_r g\|_\infty \\
&\overset{(k)}{\le} \sum_{r=t}^{d} \|\mathrm{E}_r g\|_\infty + \sum_{r=t}^{d-1} \sum_{\substack{\mathcal{K} \in 2^{[\![r+1,d]\!]}: \\ \mathcal{K} \neq \emptyset}} \|\mathrm{E}_{\mathcal{K}} \mathrm{E}_r g\|_\infty \\
&\overset{(l)}{=} \sum_{r=t}^{d} \|\mathrm{E}_r g\|_\infty + \sum_{r=t}^{d-1} \sum_{\substack{\mathcal{K} \in 2^{[\![r,d]\!]}: \\ r \in \mathcal{K},\, |\mathcal{K}| \ge 2}} \|\mathrm{E}_{\mathcal{K}} g\|_\infty \\
&\overset{(m)}{=} \sum_{r=t}^{d} \sum_{\substack{\mathcal{K} \in 2^{[\![r,d]\!]}: \\ r \in \mathcal{K}}} \|\mathrm{E}_{\mathcal{K}} g\|_\infty \\
&\overset{(n)}{=} \sum_{\substack{\mathcal{K} \in 2^{[\![t,d]\!]}: \\ \mathcal{K} \neq \emptyset}} \|\mathrm{E}_{\mathcal{K}} g\|_\infty \ ,
\end{aligned}$$

where (h) holds by independently maximizing each term of

the sum $\sum_{r=t}^{d}$, (i) holds by definition of uniform norm, (j) holds by the triangle inequality, (k) holds by the induction hypothesis, (l) holds by definition of E, (m) combines the singleton and non-singleton sequences with lowest term r, and (n) combines the sequences in $2^{[\![t,d]\!]}$ grouped by their lowest term r. We note that step (j) corresponds to the decomposition of red nodes in Figure 1 and, for example, (14) in the proof sketch in Section III-B. Furthermore, each copy of the inner sum in step (m) is illustrated as a subtree in Figure 1.

Lastly, we obtain Proposition 2 as a specific case of the inductive argument above. We have

$$\left\|\mathrm{P}g-g\right\|_\infty = \left\|\mathrm{P}_{1,\ldots,d}g-g\right\|_\infty \overset{(a)}{\leq} \sum_{\substack{\mathcal{K}\in 2^{[d]}:\\ \mathcal{K}\neq\emptyset}} \left\|\mathrm{E}_{\mathcal{K}}g\right\|_\infty$$

as desired, where (a) holds by (16). □

Next, we prove Proposition 3, which bounds univariate interpolation errors.

Proof of Proposition 3. Given any $x \in [a,b]$ distinct from the abscissae $\mathcal{G}$, let $\mathrm{Q}_x h : [a,b] \to \mathbb{R}$ be the degree-$(n+1)$ polynomial which interpolates h at the points $\mathcal{G} \cup \{x\}$. We have

$$\begin{aligned}
\left\|\mathrm{P}h-h\right\|_\infty &\overset{(a)}{=} \max_{x\in[a,b]-\mathcal{G}} \left|\mathrm{P}h(x)-h(x)\right| \\
&\overset{(b)}{=} \max_{x\in[a,b]-\mathcal{G}} \left|\mathrm{Q}_x h(x)-\mathrm{P}h(x)\right| \\
&\overset{(c)}{=} \max_{x\in[a,b]-\mathcal{G}} \left| h[u_1,\ldots,u_{n+1},x]\prod_{i=1}^{n+1}(x-u_i)\right| \\
&\overset{(d)}{\leq} n!\,(\Delta u)^{n+1} \max_{x\in[a,b]-\mathcal{G}} \left|h[u_1,\ldots,u_{n+1},x]\right| ,
\end{aligned} \tag{17}$$

where it suffices to take the maximum in (a) over $x \notin \mathcal{G}$ because $\mathrm{P}h$ interpolates h at all points in $\mathcal{G}$, (b) holds because $\mathrm{Q}_x h(x) = h(x)$ by definition and $\mathrm{Q}_x h$ is well-defined since $x \notin \mathcal{G}$, (c) holds because the Newton basis representation of $\mathrm{Q}_x h$ is (cf. [58, p. 308])

$$\mathrm{Q}_x h(\tau) = \mathrm{P}h(\tau) + h[u_1,\ldots,u_{n+1},x]\prod_{i=1}^{n+1}(\tau-u_i)\,,$$

and (d) holds by the even spacing of the grid $\mathcal{G}$.

Next, we will show by induction that for any $t \geq q$ and any distinct (not necessarily sorted) points $\{z_i\}_{i=1}^{t+1} \subset [a,b]$,

$$\left|h[z_1,\ldots,z_{t+1}]\right| \leq \left(\frac{2}{b-a}\right)^{t-q} \frac{\left\|h^{(q)}\right\|_\infty}{q!}\,. \tag{18}$$

Base case: $t = q$. By inspection, we have

$$\begin{aligned}
\left|h[z_1,\ldots,z_{t+1}]\right| &= \left|h[z_1,\ldots,z_{q+1}]\right| \\
&\overset{(a)}{\leq} \frac{\left\|h^{(q)}\right\|_\infty}{q!} \\
&= \left(\frac{2}{b-a}\right)^{t-q} \frac{\left\|h^{(q)}\right\|_\infty}{q!}\,,
\end{aligned}$$

where (a) holds by the mean value theorem for divided differences [58, p. 312].

Inductive step: $t > q$. We have

$$\begin{aligned}
\left|h[z_1,\ldots,z_{t+1}]\right| &\overset{(a)}{=} \left|\frac{h[z_2,\ldots,z_{t+1}]-h[z_1,\ldots,z_t]}{z_{t+1}-z_1}\right| \\
&\overset{(b)}{\leq} \frac{1}{b-a}\left|h[z_2,\ldots,z_{t+1}]-h[z_1,\ldots,z_t]\right| \\
&\overset{(c)}{\leq} \frac{1}{b-a}\Big(\left|h[z_2,\ldots,z_{t+1}]\right|+\left|h[z_1,\ldots,z_t]\right|\Big) \\
&\overset{(d)}{\leq} \frac{2}{b-a}\left(\frac{2}{b-a}\right)^{t-1-q}\frac{\left\|h^{(q)}\right\|_\infty}{q!} \\
&= \left(\frac{2}{b-a}\right)^{t-q}\frac{\left\|h^{(q)}\right\|_\infty}{q!}
\end{aligned}$$

as desired, where (a) holds by the definition of divided differences (1), (b) holds because z_1 and z_{t+1} are points in $[a,b]$, (c) holds by the triangle inequality, and (d) holds by the induction hypothesis.

Proceeding from (17), we have

$$\begin{aligned}
\left\|\mathrm{P}h-h\right\|_\infty &\overset{(a)}{\leq} n!\,(\Delta u)^{n+1}\left(\frac{2}{b-a}\right)^{n+1-q}\frac{\left\|h^{(q)}\right\|_\infty}{q!} \\
&\overset{(b)}{=} n!\,(\Delta u)^{n+1}\left(\frac{2}{n\Delta u}\right)^{n+1-q}\frac{\left\|h^{(q)}\right\|_\infty}{q!} \\
&\leq \frac{2^{n+1-q}}{n}\,(\Delta u)^q\left\|h^{(q)}\right\|_\infty
\end{aligned}$$

as desired, where (a) holds by (18) because $n+1 \geq q$ and (b) holds by the even spacing of the grid $\mathcal{G}$. □

Next, we prove Proposition 4, which bounds the uniform norm of an iterated interpolation error in terms of the partial derivatives of the function being interpolated.

Proof of Proposition 4. We have

$$\begin{aligned}
&\left\|\mathrm{E}_{j_1,\ldots,j_r}g\right\|_\infty \\
&\overset{(a)}{=} \left\|\mathrm{P}_{j_r}\mathrm{E}_{j_1,\ldots,j_{r-1}}g-\mathrm{E}_{j_1,\ldots,j_{r-1}}g\right\|_\infty \\
&\overset{(b)}{\leq} \alpha_1\,(\Delta u)\left\|\frac{\partial \mathrm{E}_{j_1,\ldots,j_{r-1}}g}{\partial x_{j_r}}\right\|_\infty \\
&\overset{(c)}{=} \alpha_1\,(\Delta u)\left\|\frac{\partial}{\partial x_{j_r}}\big(\mathrm{P}_{j_{r-1}}\mathrm{E}_{j_1,\ldots,j_{r-2}}g-\mathrm{E}_{j_1,\ldots,j_{r-2}}g\big)\right\|_\infty \\
&\overset{(d)}{=} \alpha_1\,(\Delta u)\left\|\frac{\partial \mathrm{P}_{j_{r-1}}\mathrm{E}_{j_1,\ldots,j_{r-2}}g}{\partial x_{j_r}}-\frac{\partial \mathrm{E}_{j_1,\ldots,j_{r-2}}g}{\partial x_{j_r}}\right\|_\infty \\
&\overset{(e)}{=} \alpha_1\,(\Delta u)\left\|\mathrm{P}_{j_{r-1}}\left(\frac{\partial \mathrm{E}_{j_1,\ldots,j_{r-2}}g}{\partial x_{j_r}}\right)-\frac{\partial \mathrm{E}_{j_1,\ldots,j_{r-2}}g}{\partial x_{j_r}}\right\|_\infty \\
&\overset{(f)}{\leq} \alpha_1^2\,(\Delta u)^2\left\|\frac{\partial^2 \mathrm{E}_{j_1,\ldots,j_{r-2}}g}{\partial x_{j_{r-1}}\partial x_{j_r}}\right\|_\infty ,
\end{aligned}$$

where (a) holds by definition of E, (b) holds by Proposition 3, (c) holds by definition of E, (d) holds by linearity of differentiation, (e) holds by Lemma 5, and (f) holds by Proposition 3. Repeating steps (c) through (f), we have

$$\begin{aligned}
&\left\|\mathrm{E}_{j_1,\ldots,j_r}g\right\|_\infty \\
&\leq \alpha_1^{r-1}\,(\Delta u)^{r-1}\left\|\frac{\partial^{r-1}\mathrm{E}_{j_1}g}{\partial x_{j_2}\cdots\partial x_{j_r}}\right\|_\infty \\
&\overset{(g)}{=} \alpha_1^{r-1}\,(\Delta u)^{r-1}\left\|\frac{\partial^{r-1}}{\partial x_{j_2}\cdots\partial x_{j_r}}(\mathrm{P}_{j_1}g-g)\right\|_\infty
\end{aligned}$$

$$\overset{(h)}{=} \alpha_1^{r-1}(\Delta u)^{r-1}\left\|\frac{\partial^{r-1}\mathrm{P}_{j_1}g}{\partial x_{j_2}\cdots\partial x_{j_r}}-\frac{\partial^{r-1}g}{\partial x_{j_2}\cdots\partial x_{j_r}}\right\|_\infty$$
$$\overset{(i)}{=} \alpha_1^{r-1}(\Delta u)^{r-1}\left\|\mathrm{P}_{j_1}\left(\frac{\partial^{r-1}g}{\partial x_{j_2}\cdots\partial x_{j_r}}\right)-\frac{\partial^{r-1}g}{\partial x_{j_2}\cdots\partial x_{j_r}}\right\|_\infty$$
$$\overset{(j)}{\le} \alpha_{\ell-r+1}\,\alpha_1^{r-1}(\Delta u)^{\ell}\left\|\frac{\partial^{\ell}g}{\partial x_{j_1}^{\ell-r+1}\partial x_{j_2}\cdots\partial x_{j_r}}\right\|_\infty$$

as desired, where (g) holds by definition of E, (h) holds by linearity of differentiation, (i) holds by repeated application of Lemma 5, and (j) holds by Proposition 3 because $n+1\ge\ell\ge\ell-r+1$. □

Next, we prove Lemma 5, which states that interchanging the order of partial differentiation and partial interpolation with respect to distinct variables gives the same result.

Proof of Lemma 5. We have

$$\begin{aligned}
&\frac{\partial \mathrm{P}_{j_1,\ldots,j_r}g}{\partial x_k}(x_1,\ldots,x_d)\\
&\overset{(a)}{=}\frac{\partial}{\partial x_k}\sum_{i_1=1}^{n+1}\cdots\sum_{i_r=1}^{n+1}g(x_1,\ldots,u_{i_s},\ldots,x_d)\prod_{s=1}^{r}\phi_{i_s}(x_{j_s})\\
&\overset{(b)}{=}\sum_{i_1=1}^{n+1}\cdots\sum_{i_r=1}^{n+1}\frac{\partial g}{\partial x_k}(x_1,\ldots,u_{i_s},\ldots,x_d)\prod_{s=1}^{r}\phi_{i_s}(x_{j_s})\\
&\overset{(c)}{=}\mathrm{P}_{j_1,\ldots,j_r}\left(\frac{\partial g}{\partial x_k}\right)(x_1,\ldots,x_d),
\end{aligned}$$

where (a) holds by definition of P, (b) holds because k is distinct from $j_1,\ldots,j_r$, and (c) holds by definition of P. □

Finally, we prove Theorem 1 using the intermediate results established above.

Proof of Theorem 1. We have

$$\begin{aligned}
\|\mathrm{P}g-g\|_\infty &\overset{(a)}{\le}\sum_{\substack{\mathcal{K}\in 2^{[d]}:\\ \mathcal{K}\neq\emptyset}}\|\mathrm{E}_{\mathcal{K}}g\|_\infty\\
&\overset{(b)}{\le}\sum_{\substack{\mathcal{K}\in 2^{[d]}:\\ \mathcal{K}\neq\emptyset}}\left(\frac{2^{n-\ell+|\mathcal{K}|}}{n}\right)\left(\frac{2^n}{n}\right)^{|\mathcal{K}|-1}(\Delta u)^{\ell}\\
&\qquad\left\|\frac{\partial^{\ell}g}{\partial x_{j_1}^{\ell-|\mathcal{K}|+1}\partial x_{j_2}\cdots\partial x_{j_{|\mathcal{K}|}}}\right\|_\infty\\
&\overset{(c)}{\le}L_h(\Delta u)^{\ell}\sum_{\substack{\mathcal{K}\in 2^{[d]}:\\ \mathcal{K}\neq\emptyset}}\left(\frac{2^{n-\ell+|\mathcal{K}|}}{n}\right)\left(\frac{2^n}{n}\right)^{|\mathcal{K}|-1}\\
&\overset{(d)}{\le}L_h(\Delta u)^{\ell}\sum_{\substack{\mathcal{K}\in 2^{[d]}:\\ \mathcal{K}\neq\emptyset}}\left(\frac{2^n}{n}\right)^{|\mathcal{K}|}\\
&=L_h(\Delta u)^{\ell}\sum_{r=1}^{d}\binom{d}{r}\left(\frac{2^n}{n}\right)^{r}\\
&\overset{(e)}{\le}L_h(\Delta u)^{\ell}\left(\frac{2^n}{n}+1\right)^{d}
\end{aligned}$$

as desired, where (a) holds by Proposition 2, (b) holds by Proposition 4 because $\ell\ge d$ and $n+1\ge\ell$, (c) holds by (4), (d) holds because $|\mathcal{K}|\le d\le\ell$, and (e) holds by the binomial theorem. □

V. Proofs of Oracle Complexity

First, we prove Lemma 6.

Proof of Lemma 6. From Theorem 1, we have

$$\|Pg-g\|_\infty\le L_h\frac{1}{m^\ell}\left(\frac{2^\ell}{\ell}+1\right)^d.$$

Bounding the right hand side by δ implies that if

$$m\ge\left(\frac{L_h}{\delta}\left(\frac{2^\ell}{\ell}+1\right)^d\right)^{1/\ell},$$

then $\|Pg-g\|_\infty\le\delta$. □

Next, we prove Theorem 7.

Proof of Theorem 7. Let the condition number $\sigma=L_l/\mu>1$. From [10] (Proposition 2), the required number of iterations T in order for an inexact gradient descent algorithm to achieve an ε-approximate solution is

$$T=\left\lceil\left(\log\left(\frac{\sigma}{\sigma-1}\right)\right)^{-1}\log\left(\frac{F(\boldsymbol{\theta}^{(0)})-F^*+\frac{p}{2\mu}}{\varepsilon}\right)\right\rceil \tag{19}$$

when the gradient estimation error δ satisfies

$$\delta=\left(1-\frac{1}{\sigma}\right)^{T/2}. \tag{20}$$

Then, note that

$$\begin{aligned}
m^d&\le\left(\left(\frac{L_h}{\delta}\left(\frac{2^\ell}{\ell}+1\right)^d\right)^{1/\ell}+1\right)^d\\
&\overset{(a)}{\le}\left(L_h^{1/\ell}2^d\left(\frac{\sigma}{\sigma-1}\right)^{1/(2\ell)}\left(\frac{F(\boldsymbol{\theta}^{(0)})-F^*+\frac{p}{2\mu}}{\varepsilon}\right)^{1/\ell}+1\right)^d\\
&\le L_h^{d/\ell}2^{d(d+1)}\left(1+\frac{\sigma}{2(L_l-\mu)}\right)^{d/(2\ell)}\\
&\qquad\left(\frac{p+2\mu(F(\boldsymbol{\theta}^{(0)})-F^*)}{\varepsilon}\right)^{d/(2\ell)}\\
&\le L_h^{d/\ell}2^{d(d+1)}\left(1+\frac{\sigma}{2(L_l-\mu)}\right)\\
&\qquad\left(\frac{p+2\mu(F(\boldsymbol{\theta}^{(0)})-F^*)}{\varepsilon}\right)^{d/(2\ell)},
\end{aligned}$$

where (a) holds since $(2^\ell/\ell+1)^{1/\ell}\le 2$ when $\ell\ge 2$. Thus

$$\begin{aligned}
&\Gamma(\text{PPI-GD})\le Tm^d\\
&\le L_h^{d/\ell}2^{d(d+1)}\left(\frac{\sigma+2(L_l-\mu)}{(L_l-\mu)\log\left(\frac{\sigma}{\sigma-1}\right)}\right)\\
&\qquad\left(\frac{p+\Delta}{\varepsilon}\right)^{d/(2\ell)}\log\left(\frac{p+\Delta}{2\mu\varepsilon}\right),
\end{aligned}$$

where $\Delta=2\mu(F(\boldsymbol{\theta}^{(0)})-F^*)$.

Suppose $\ell \geq d$ and $d \leq \log^{-1/2}(2)\log^{\gamma}(n) - 1/2$ for some $0 < \gamma < 1/2$. Then, note that

$$\begin{aligned} 2^{d(d+1)} &= 2^{(d+(1/2))^2 - 1/4} \\ &\leq 2^{-1/4}\exp\left(\log(2)\left(\log^{-1/2}(2)\log^{\gamma}(n)\right)^2\right) \\ &= 2^{-1/4}\exp\left(\log^{2\gamma}(n)\right) \end{aligned}$$

for sufficiently large n. If we assume that the parameters $\boldsymbol{\theta}$ are bounded in some hypercube of constant edge length (which is usually the case in practice), $\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|_2^2 = O(p)$, which implies $\Delta = O(p)$ (due to Lipschitz continuity). Thus, we get

$$\begin{aligned} \Gamma(\text{PPI-GD}) &\leq L_h^{d/\ell} 2^{d(d+1)} \left(\frac{\sigma + 2(L_l - \mu)}{(L_l - \mu)\log\left(\frac{\sigma}{\sigma - 1}\right)}\right) \\ &\qquad \left(\frac{p+\Delta}{\varepsilon}\right)^{d/(2\ell)} \log\left(\frac{p+\Delta}{2\mu\varepsilon}\right) \\ &\leq C\exp\left(\log^{2\gamma}(n)\right)\left(\frac{p}{\varepsilon}\right)^{d/(2\ell)}\log\left(\frac{p}{\varepsilon}\right), \end{aligned}$$

where C depends on μ, L_l, ℓ, and L_h but not d, n, p, or ε. □

Next, we prove Proposition 8.

Proof of Proposition 8.

$$\begin{aligned} \Gamma(\text{PPI-GD}) &\overset{(a)}{\leq} C\exp\left(\log^{2\gamma}(n)\right)\left(\frac{p}{\varepsilon}\right)^{d/(2\ell)}\log\left(\frac{p}{\varepsilon}\right) \\ &\overset{(b)}{=} O(\mathsf{subpoly}(n)(n^{\beta})^{d/(2\ell)}) \\ &= \tilde{O}(n^{(d\beta)/(2\ell)}), \end{aligned}$$

where (a) is from Theorem 7 and (b) follows from the regime $p/\varepsilon = O(n^{\beta})$. A similar argument can be used to prove the oracle complexity results for the remaining methods based on known bounds presented in Table I for GD and SGD, [25] for SVRG, and [13] for MBGD (also see [10]). □

Now, we apply similar techniques to the non-convex setting. First, we prove Theorem 9.

Proof of Theorem 9. First, by setting $\delta = \frac{\varepsilon}{2\sqrt{p}}$ in Lemma 6, we get

$$\left\|\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}) - \boldsymbol{\nabla}F(\boldsymbol{\theta})\right\|_2^2 \leq \frac{\varepsilon^2}{4}.$$

Next, note that Assumption 1 (L_l-smoothness) implies

$$\begin{aligned} &F(\boldsymbol{\theta}^{(t+1)}) \\ &\leq F(\boldsymbol{\theta}^{(t)}) + \boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})^{\mathsf{T}}(\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}) \\ &\qquad + \frac{L_l}{2}\left\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\right\|_2^2 \\ &= F(\boldsymbol{\theta}^{(t)}) - \alpha\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})^{\mathsf{T}}\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}^{(t)}) \\ &\qquad + \frac{L_l\alpha^2}{2}\left\|\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 \\ &= F(\boldsymbol{\theta}^{(t)}) - \alpha\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 \\ &\qquad - \alpha\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})^{\mathsf{T}}\left(\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}^{(t)}) - \boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right) \\ &\qquad + \frac{L_l\alpha^2}{2}\left\|\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 \\ &\leq F(\boldsymbol{\theta}^{(t)}) - \alpha\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 \\ &\qquad + \frac{\alpha}{2}\left(\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 + \left\|\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}^{(t)}) - \boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2\right) \\ &\qquad + L_l\alpha^2\left(\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 + \left\|\widehat{\boldsymbol{\nabla}}F(\boldsymbol{\theta}^{(t)}) - \boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2\right) \\ &\leq F(\boldsymbol{\theta}^{(t)}) + \left(L_l\alpha^2 - \frac{\alpha}{2}\right)\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 + \frac{\varepsilon^2}{4}\left(L_l\alpha^2 + \frac{\alpha}{2}\right) \\ &= F(\boldsymbol{\theta}^{(t)}) - \frac{1}{16L_l}\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 + \frac{3\varepsilon^2}{64L_l}. \end{aligned}$$

(Recall that $\alpha = 1/(4L_l)$ is the step size.) Thus,

$$\sum_{t=0}^{T-1}\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 \leq 16L_l(F(\boldsymbol{\theta}^{(0)}) - F^*) + \frac{3}{4}T\varepsilon^2,$$

so if $T = 64L_l(F(\boldsymbol{\theta}^{(0)}) - F^*)/\varepsilon^2$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\left\|\boldsymbol{\nabla}F(\boldsymbol{\theta}^{(t)})\right\|_2^2 \leq \frac{16L_l}{T}(F(\boldsymbol{\theta}^{(0)}) - F^*) + \frac{3}{4}\varepsilon^2 = \varepsilon^2.$$

This implies that a ε-stationary point (as defined in (3)) is reached at least once. Therefore,

$$\begin{aligned} &\Gamma(\text{PPI-GD}) \leq Tm^d \\ &\leq \frac{64L_l(F(\boldsymbol{\theta}^{(0)}) - F^*)}{\varepsilon^2}\left(\left(\frac{2\sqrt{p}L_h}{\varepsilon}\left(\frac{2^{\ell}}{\ell} + 1\right)^d\right)^{1/\ell} + 1\right)^d \\ &\leq \frac{64L_l(F(\boldsymbol{\theta}^{(0)}) - F^*)}{\varepsilon^2}\left(\left(2^{d+(1/\ell)}{L_h}^{1/\ell}\left(\frac{p}{\varepsilon^2}\right)^{1/(2\ell)}\right) + 1\right)^d \\ &\leq \frac{64L_l(F(\boldsymbol{\theta}^{(0)}) - F^*)}{\varepsilon^2}2^{d^2+d+d/\ell}{L_h}^{d/\ell}\left(\frac{p}{\varepsilon^2}\right)^{d/(2\ell)} \\ &\overset{(a)}{\leq} C_2\exp\left(\log^{2\gamma}(n)\right)\left(\frac{p}{\varepsilon^2}\right)^{1+d/(2\ell)} \end{aligned}$$

where (a) follows from $2^{d(d+1)} \leq 2^{-1/4}\exp\left(\log^{2\gamma}(n)\right)$ and $F(\boldsymbol{\theta}^{(0)}) - F^* = O(p)$ as shown in the proof of Theorem 7.

We note that this flavor of analysis in the non-convex setting is standard in the literature (see, e.g., [22], [73]). □

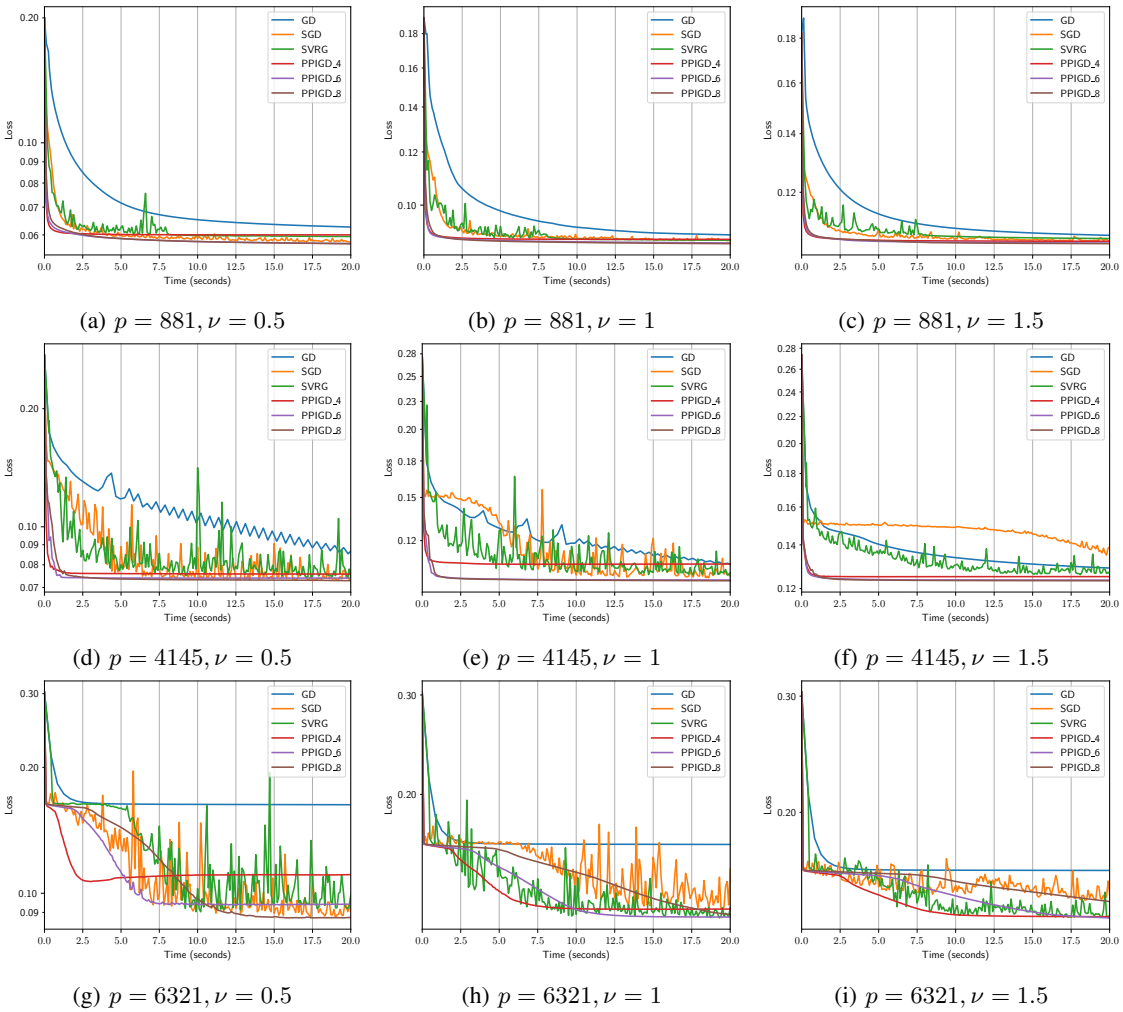
Finally, we prove Proposition 10.

Proof of Proposition 10. First, [12, Theorem 2.1] and [14, Corollary 2.2] imply the oracle complexities of GD and SGD to be $O(n(p/\varepsilon^2))$ and $O((p/\varepsilon^2)^2)$, respectively. Plugging in $p/\varepsilon^2 = O(n^{\beta})$ produces the desired result. PPI-GD's oracle complexity can be derived similarly from Theorem 9, noting that $O(\exp(\log^{2\gamma}(n)))$ is subpolynomial when $\gamma < 1/2$. □

VI. Experiments

Experiments were conducted to evaluate the empirical performance of PPI-GD compared to other GD methods when used to train a neural network classifier. The models and optimization algorithms were implemented with the NumPy Python library. All experiments were run on a machine with an Intel Core i7-8700K CPU and 16 GB of memory.

Three synthetic classification datasets were generated for the experiments. Each dataset is parameterized by the variable $\nu \in \{0.5, 1, 1.5\}$, which captures the "noisiness" of the dataset. Each dataset consists of $n = 10000$ randomly generated samples in $[0,1]^2 \times \{0,1\}$. Each data point is of the



(a) $p = 881, \nu = 0.5$ (b) $p = 881, \nu = 1$ (c) $p = 881, \nu = 1.5$

(d) $p = 4145, \nu = 0.5$ (e) $p = 4145, \nu = 1$ (f) $p = 4145, \nu = 1.5$

(g) $p = 6321, \nu = 0.5$ (h) $p = 6321, \nu = 1$ (i) $p = 6321, \nu = 1.5$

Fig. 2: Comparison of GD, SGD, SVRG, and PPI-GD (with $\ell = 1$ and $m \in \{4, 6, 8\}$) when training a neural network. $p \in \{881, 4145, 6321\}$ is the number of parameters, and $\nu \in \{0.5, 1, 1.5\}$ is the noise in data.

TABLE II: Lowest loss attained for each optimizer, rounded to four decimal places. The best result in each round is highlighted.

Optimizer	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
GD	0.0627	0.0903	0.1071	0.0853	0.1128	0.1290	0.1628	0.1631	0.1633
SGD	0.0577	0.0885	0.1055	**0.0728**	0.1068	0.1351	0.0877	0.1250	0.1472
SVRG	0.0595	0.0887	0.1062	0.0741	0.1081	0.1260	0.0903	**0.1209**	0.1385
PPIGD_4	0.0600	0.0889	0.1055	0.0758	0.1131	0.1251	0.1067	0.1252	0.1389
PPIGD_6	**0.0572**	0.0878	0.1050	0.0740	0.1059	0.1237	0.0937	0.1213	**0.1383**
PPIGD_8	0.0572	**0.0876**	**0.1048**	0.0729	**0.1055**	**0.1234**	**0.0873**	0.1226	0.1464

form $(x_1, x_2, \mathbb{1}\{\sin(5x_1) + \sin(5x_2) + \xi\})$, where $x_1, x_2 \sim \text{Unif}(0, 1)$ and $\xi \sim \mathcal{N}(0, \nu)$.

Fully connected leaky ReLU (rectified linear unit) neural networks were used to classify the datasets. Each neural network has $h \in \{4, 16, 24\}$ hidden layers with 16 nodes each. The number of trainable parameters in each model is $p = 272h - 207 \in \{881, 4145, 6321\}$.

In each round of simulations, the performance (in terms of wall-clock time) of GD, SGD, SVRG, and three separate runs of PPI-GD with different parameters was measured. Each algorithm was allowed to run for 20 seconds, and the training MSE loss was recorded every 0.1 seconds. For consistency, the same learning rate schedule (inverse square root with an initial value of 0.2) was used for every optimizer. The update frequency of SVRG was set to $n = 10000$. Hyperparameters $\ell = 1$ and $m \in \{4, 6, 8\}$ were chosen for PPI-GD (each denoted by PPIGD_4, PPIGD_6, and PPIGD_8 in the plots).

Figure 2 shows the learning curves for each combination of dataset and model, and Table II presents the lowest loss attained. The results demonstrate that PPI-GD can outperform GD and SGD not only theoretically but also in simulations in certain settings.

As expected, the optimizers converge more quickly when the model is simpler (i.e., when p is smaller) or when the data is less noisy (i.e., when ν is smaller). Notably, the performance of SGD degrades quickly in the face of noise. This is because

SGD performs well on datasets that can be easily generalized from a small subset, and noise makes this much more difficult. Indeed, it is evident from Figure 2 that the other algorithms are less affected by noise than SGD.

Figure 2(g) is especially noteworthy, as it clearly depicts the unique characteristics of PPI-GD. Although PPI-GD with $m = 4$ improves very quickly at first, it soon converges to a suboptimal point. This is because PPI-GD, unlike some other inexact gradient methods like SGD, is biased. As a result, unlike SGD, PPI-GD does not converge to the exact solution even when $T \to \infty$. When the grid size is small, each iteration is very computationally efficient, but the approximate gradient $\widehat{\boldsymbol{\nabla}} F(\boldsymbol{\theta}^{(t)})$ is less accurate. As m increases, the initial rate of convergence becomes slower, but the algorithm ultimately converges to a more accurate solution.

VII. Conclusion

In this paper, we introduced the PPI-GD algorithm for optimizing ERM objectives which satisfy Hölder smoothness assumptions in the training data, by using multivariate polynomial interpolation to approximate the true gradient oracle at each iteration. When the data space dimension satisfies $d = O(\log^{0.49}(n))$, we established that the oracle complexity of PPI-GD scales as $\tilde{O}((p/\varepsilon)^{d/(2\ell)})$ in the strongly convex setting and $\tilde{O}((p/\varepsilon^2)^{1+d/(2\ell)})$ in the non-convex setting, showing that our algorithm has lower complexity than GD, SGD, and variants for sufficiently smooth loss functions in the common regime where $p = O(\mathsf{poly}(n))$ and $\varepsilon = \Theta(\mathsf{poly}(1/n))$. PPI-GD improves upon former inexact gradient methods for smooth objectives by relaxing necessary conditions on the scaling of d, showing that an astute choice of gradient estimation method contributes substantially towards obtaining algorithms with theoretical gains in oracle complexity in a broad range of domains.

Future research in this vein may incorporate classic techniques such as momentum, Nesterov acceleration, and mini-batching, since the primary innovation of PPI-GD in leveraging smoothness in training data is orthogonal to these techniques in principle. Moreover, our present work builds upon the classical literature on bicubic spline interpolation to derive generalized error bounds on polynomial interpolants, and a fruitful future direction may involve analyzing natural spline interpolants in the same generalized manner. Overall, our main contributions suggest that interesting complexity results remain to be discovered at the intersection of inexact gradient methods and interpolation analysis.

References

[1] A. S. Nemirovskii and D. B. Yudin, *Problem complexity and method efficiency in optimization*. Wiley-Interscience, 1983.

[2] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Science & Business Media, 2004.

[3] M. Schmidt, "Convergence rate of stochastic gradient with constant step size," Sep 2014. [Online]. Available: https://open.library.ubc.ca/collections/facultyresearchandpublications/52383/items/1.0050992

[4] S. Lacoste-Julien, M. Schmidt, and F. R. Bach, "A simpler approach to obtaining an o(1/t) convergence rate for the projected stochastic subgradient method," *CoRR*, vol. abs/1212.2002, 2012. [Online]. Available: http://arxiv.org/abs/1212.2002

[5] A. Agarwal, M. J. Wainwright, P. L. Bartlett, and P. K. Ravikumar, "Information-theoretic lower bounds on the oracle complexity of convex optimization," in *Proceedings of the Advances in Neural Information Processing Systems 22*, Vancouver, BC, Canada, December 6-11 2009, pp. 1–9.

[6] A. d'Aspremont, "Smooth optimization with approximate gradient," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1171–1183, October 2008.

[7] 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, Jan. 2012.

[8] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods with inexact oracle: the strongly convex case," *CORE Discussion Papers 2013016*, vol. 2013, no. 16, March 2013.

[9] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Mathematical Programming, Series A*, vol. 146, pp. 37–75, August 2014.

[10] A. Jadbabaie, A. Makur, and D. Shah, "Gradient-based empirical risk minimization using local polynomial regression," *Stochastic Systems, INFORMS*, pp. 1–40, March 2024.

[11] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in Neural Information Processing Systems*, vol. 4. Morgan-Kaufmann, 1991.

[12] S. A. Vavasis, "Black-box complexity of local minimization," *SIAM Journal on Optimization*, vol. 3, no. 1, pp. 60–80, Feb. 1993.

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

[14] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[15] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, September 1951.

[16] 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.

[17] 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.

[18] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2880–2887.

[19] 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.

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

[21] R. Cosson, A. Jadbabaie, A. Makur, A. Reisizadeh, and D. Shah, "Gradient descent with low-rank objective functions," in *Proceedings of the 62nd IEEE Conference on Decision and Control (CDC)*, Singapore, December 13-15 2023, pp. 3309–3314.

[22] R. Cosson, A. Jadbabaie, A. Makur, A. Reisizadeh, and D. Shah, "Low-rank gradient descent," *IEEE Open Journal of Control Systems*, vol. 2, pp. 380–395, October 2023.

[23] A. Jadbabaie, A. Makur, and A. Reisizadeh, "Adaptive low-rank gradient descent," in *Proceedings of the 62nd IEEE Conference on Decision and Control (CDC)*, Singapore, December 13-15 2023, pp. 3315–3320.

[24] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, ser. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011.

[25] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, vol. 26. Curran Associates, Inc., 2013.

[26] N. Le Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proceedings of the Advances in Neural Information Processing Systems 25 (NeurIPS)*, Lake Tahoe, NV, USA, December 3-8 2012, pp. 1–9.

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

[28] 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.

[29] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, July 2011.

[30] K. Chakrabarti and N. Chopra, "Generalized adagrad (g-adagrad) and adam: A state-space perspective," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 1496–1501.

[31] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," 2012, cOURSERA: Neural Networks for Machine Learning.

[32] 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.

[33] 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.

[34] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York, NY, USA: Springer, 2006.

[35] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., ser. Springer Series in Statistics. New York, NY, USA: Springer, 2009.

[36] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, QC, Canada, December 2-8 2018, pp. 1–11.

[37] 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.

[38] Y. Arjevani, Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro, and B. Woodworth, "Lower bounds for non-convex stochastic optimization," *Mathematical Programming*, vol. 199, no. 1, pp. 165–214, May 2023.

[39] A. Ramaswamy and S. Bhatnagar, "Analysis of gradient descent methods with nondiminishing bounded errors," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1465–1471, 2017.

[40] I. Necoara and V. Nedelcu, "Rate analysis of inexact dual first-order methods application to dual decomposition," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1232–1243, 2013.

[41] S. Vlaski and A. H. Sayed, "Second-order guarantees of stochastic gradient descent in nonconvex optimization," *IEEE Transactions on Automatic Control*, vol. 67, no. 12, pp. 6489–6504, 2021.

[42] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent for convex and smooth functions," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2260–2267.

[43] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2566–2581, 2019.

[44] E. Trimbach, E. D. H. Nguyen, and C. A. Uribe, "On acceleration of gradient-based empirical risk minimization using local polynomial regression," in *2022 European Control Conference (ECC)*. IEEE, 2022, pp. 429–434.

[45] N. Bhavsar and L. Prashanth, "Nonasymptotic bounds for stochastic optimization with biased noisy gradient oracles," *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1628–1641, 2022.

[46] J. Fan and I. Gijbels, "Local polynomial modeling and its applications," *London: Chapmanand*, 1996.

[47] K. De Brabanter, J. De Brabanter, I. Gijbels, and B. De Moor, "Derivative estimation with local polynomial fitting," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 281–301, 2013.

[48] M. Delecroix and A. Rosa, "Nonparametric estimation of a regression function and its derivatives under an ergodic hypothesis," *Journal of Nonparametric Statistics*, vol. 6, no. 4, pp. 367–382, 1996.

[49] Y. Wang, S. Du, S. Balakrishnan, and A. Singh, "Stochastic zeroth-order optimization in high dimensions," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1356–1365.

[50] G. Ausset, S. Clémençon, and F. Portier, "Nearest neighbour based estimates of gradients: Sharp nonasymptotic bounds and applications," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 532–540.

[51] A. Jadbabaie, A. Makur, and D. Shah, "Federated optimization of smooth loss functions," *IEEE Transactions on Information Theory*, vol. 69, no. 12, pp. 7836–7866, December 2023.

[52] E. Choi, C. Xiao, W. Stewart, and J. Sun, "Mime: Multilevel medical embedding of electronic health records for predictive healthcare," *Advances in neural information processing systems*, vol. 31, 2018.

[53] M. C. Nechyba and Y. Xu, "Neural network approach to control system identification with variable activation functions," in *Proceedings of 1994 9th IEEE International Symposium on Intelligent Control*. IEEE, 1994, pp. 358–363.

[54] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[55] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in neural information processing systems*, vol. 14, 2001.

[56] S.-L. Huang, A. Makur, G. W. Wornell, and L. Zheng, *Universal Features for High-Dimensional Learning and Inference*, ser. Foundations and Trends in Communications and Information Theory, A. Barg, Ed. Hanover, MA, USA: now Publishers Inc., February 2024, vol. 21, no. 1-2.

[57] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of johnson and lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[58] U. M. Ascher and C. Greif, *A First Course on Numerical Methods*. SIAM, 2011.

[59] A. B. Tsybakov, *Introduction to Nonparametric Estimation*, ser. Springer Series in Statistics. New York, NY: Springer, 2009.

[60] V. Sadhanala, Y.-X. Wang, J. L. Sharpnack, and R. J. Tibshirani, "Higher-order total variation classes on grids: Minimax theory and trend filtering methods," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[61] G. H. Chen and D. Shah, "Explaining the success of nearest neighbor methods in prediction," *Foundations and Trends® in Machine Learning*, vol. 10, no. 5-6, pp. 337–588, 2018.

[62] J. Xu, "Rates of convergence of spectral methods for graphon estimation," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 5433–5442.

[63] A. Agarwal, D. Shah, D. Shen, and D. Song, "On robustness of principal component regression," *Journal of the American Statistical Association*, Oct. 2021.

[64] G. Gur-Ari, D. A. Roberts, and E. Dyer, "Gradient descent happens in a tiny subspace," Dec. 2018.

[65] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, "Empirical analysis of the Hessian of over-parametrized neural networks," in *International Conference on Learning Representations*, 2018.

[66] T. Le and S. Jegelka, "Training invariances and the low-rank phenomenon: beyond linear networks," in *International Conference on Learning Representations*, 2022.

[67] L. Van Der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," Tilburg University Technical Report, Tech. Rep. TiCC-TR 2009-005, 2009.

[68] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33.

[69] C. de Boor, "Bicubic spline interpolation," *Journal of Mathematics and Physics*, vol. 41, no. 3, pp. 212–218, 1962.

[70] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[71] R. E. Carlson and C. A. Hall, "Error bounds for bicubic spline interpolation," *Journal of Approximation Theory*, vol. 7, no. 1, pp. 41–47, 1973.

[72] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[73] H. Li, J. Qian, Y. Tian, A. Rakhlin, and A. Jadbabaie, "Convex and non-convex optimization under generalized smoothness," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 40 238–40 271.