Gradient-Based Empirical Risk Minimization using Local Polynomial Regression

Ali Jadbabaie  
Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, jadbabai@mit.edu

Anuran Makur  
Department of Computer Science and Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, amakur@purdue.edu

Devavrat Shah  
Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, devavrat@mit.edu

In this paper, we consider the widely studied problem of empirical risk minimization (ERM) of strongly convex and smooth loss functions using iterative gradient-based methods. A major goal of the existing literature has been to compare different prototypical algorithms, such as batch gradient descent (GD) or stochastic gradient descent (SGD), by analyzing their rates of convergence to $\epsilon$-approximate solutions with respect to the number of gradient computations, which is also known as the oracle complexity. For example, the oracle complexity of GD is $O(n \log(p\epsilon^{-1}))$, where $n$ is the number of training samples and $p$ is the parameter space dimension. When $n$ is large, this can be prohibitively expensive in practice, and SGD is preferred due to its oracle complexity of $O(p\epsilon^{-1})$. Such standard analyses only utilize the smoothness of the loss function in the parameter being optimized. In contrast, we demonstrate that when the loss function is smooth in the data, we can learn the oracle at every iteration and beat the oracle complexities of both GD and SGD in important regimes. Specifically, at every iteration, our proposed algorithm, Local Polynomial Interpolation based Gradient Descent (LPI-GD), first performs local polynomial regression with a virtual batch of data points to learn the gradient of the loss function, and then estimates the true gradient of the ERM objective function. We establish that the oracle complexity of LPI-GD is $\tilde{O}(p\epsilon^{-1})$, where $d$ is the data space dimension and the gradient of the loss function is assumed to belong to an $\eta$-Hölder class with respect to the data. Our proof extends the analysis of local polynomial regression in non-parametric statistics to provide supremum norm guarantees for interpolation in multivariate settings, and also exploits tools from the inexact GD literature. Unlike the complexities of GD and SGD, the complexity of our method depends on $d$. However, our algorithm outperforms GD and SGD in oracle complexity for a broad range of settings. For example, with typical loss functions (such as squared or cross-entropy loss), when $p = O(n^\beta)$ for any $\beta > 0$ and $\epsilon = \Theta(n^{-1/2})$ is at the statistical limit, our method can be made to require $O(n^\delta)$ oracle calls for any $\delta > 0$, while SGD and GD require $O(n^{\beta+1/2})$ and $O(n \log(n))$ oracle calls, respectively.

Key words: gradient descent, empirical risk minimization, local polynomial regression, oracle complexity

1. Introduction

Empirical risk minimization (ERM) is one of the mainstays of contemporary machine learning. Indeed, training tasks such as classification, regression, or representation learning using deep neural networks, can all be formulated as specific instances of ERM. In this paper, we consider gradient-based iterative optimization methods that are used to perform ERM, such as batch gradient descent (GD) (Nesterov 2004), stochastic gradient descent (SGD) (Nemirovski et al. 2009), and their refinements (e.g., Nesterov 1983, Dekel et al. 2012, Johnson and Zhang 2013). Since the primary computational bottleneck in such settings is gradient computation, a significant portion of this literature is concerned with analyzing the convergence rates of different iterative algorithms and determining which ones have smaller oracle complexities, i.e., smaller number of gradient queries.
required to obtain approximate minimizers, under various levels of convexity, smoothness, sampling, and other assumptions (see, e.g., Bubeck 2015, Jain and Kar 2017, Netrapalli 2019, and the references therein). We focus on the widely studied setting of strongly convex loss functions. In this case, neglecting any implicit dependence on dimension, the (first order) oracle complexity of GD is known to be $O(n \log(\varepsilon^{-1}))$ (Nesterov 2004, Theorem 2.1.15) where $n$ is the number of training samples in the ERM objective function, and $\varepsilon > 0$ is the desired approximation accuracy of the solution. In comparison, it is well-known that SGD has an oracle complexity of $O(\varepsilon^{-1})$ (Nemirovski et al. 2009, Bottou et al. 2018), again neglecting any implicit dependence on dimension. It turns out that by carefully accounting for dependence on the dimension $p$ of the parameter of interest in such an ERM, the oracle complexities of GD and SGD are $O(n \log(p \varepsilon^{-1}))$ and $O(p \varepsilon^{-1})$, respectively (see Propositions 4 and 5 in Appendix A). Typically, when performing ERM, the objective is to learn the true risk minimizer as accurately as possible. However, the empirical risk is statistically inaccurate, generally, up to a scaling of order $n^{-1/2}$. Hence, the typical regime of interest is $\varepsilon = \Omega(n^{-1/2})$ when $n$ is large. For small $p$ (i.e., $p \ll \sqrt{n}$), this makes SGD, at least theoretically, preferable to GD. This is corroborated by the ubiquitous use of SGD and its variants for ERM in practice. Our goal is to improve upon the oracle complexities of GD, SGD, and their variants by exploiting additional information about the loss functions that has heretofore been neglected in the literature.

Recall that the general unconstrained ERM problem is usually abstracted as (cf. Bubeck 2015, Section 1, Hastie et al. 2009):

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(\theta),$$  \hspace{1cm} (1)

where each $f_i : \mathbb{R}^p \to \mathbb{R}$ represents the loss function corresponding to a different sample of training data. (Note that this formulation, modulo the extra scaling $n^{-1}$, is also utilized in distributed optimization, where each $f_i$ represents the local objective function corresponding to an individual agent in a network of $n$ agents (Nedić 2015).) As suggested by the form of (1), typical analysis of iterative gradient-based algorithms, such as GD or SGD, makes no assumptions about $f_i$ as a function of $i$. However, the loss function often varies quite smoothly with the data in ERM. For example, given training data $\{(x^{(i)}, y_i) \in \mathbb{R}^p \times \mathbb{R} : i \in [n]\}$ (with $[n] \triangleq \{1, \ldots, n\}$), the classical least squares formulation of linear regression with regularization is (see, e.g., Hastie et al. 2009, Chapter 3):

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x^{(i)})^2 + R(\theta),$$  \hspace{1cm} (2)

where the objective is to find the optimal coefficients $\theta \in \mathbb{R}^p$, and $R(\theta)$ is some regularization term, e.g., ridge or lasso. Letting $f(x, y; \theta) = (y - \theta^T x)^2 + R(\theta)$ for $x, \theta \in \mathbb{R}^p$ and $y \in \mathbb{R}$, we see that each $f_i(\theta) = f(x^{(i)}, y_i; \theta)$ in formulation (1) for this problem. Clearly, $f$ is very smooth as a function of the data $(x, y)$, regardless of the regularization $R(\theta)$. This observation that loss functions are often smooth in the data holds in more complex learning scenarios too. For instance, when training a deep neural network with a smooth loss, e.g., cross-entropy loss, and smooth activation functions, e.g., logistic (sigmoid) functions, the induced loss function for the ERM objective function is infinitely differentiable in both the data and the network’s weight parameters (Bishop 2006, Chapter 5). Since such smoothness in the data has not been exploited by modern analysis of iterative gradient-based optimization algorithms for (1) (see Section 1.1), the main thrust of this work is to speed up algorithms for ERM using this additional smoothness in the data.

To this end, in the remainder of this work, we consider the following problem formulation of ERM. For $d, n \in \mathbb{N}$, assume that we are given $n$ samples of $d$-dimensional training data $\mathcal{D} = \{x^{(i)} \in [h', 1 - h']^d : i \in [n]\}$, where $h' > 0$ is an arbitrarily small parameter (to be determined later; see (19)), and each training sample belongs to the hypercube $[h', 1 - h']^d$ without loss of generality.\footnote{This compactness assumption affords us some analytical tractability in the sequel.}
vector. Now define the ERM objective function $p$ valued label (as in the linear regression example above). For a given feature vector, and the last element of the sample can be construed as the corresponding continuous-

Note that when $d \geq 2$, the first $d-1$ elements of a sample can be perceived as a $(d-1)$-dimensional feature vector, and the last element of the sample can be construed as the corresponding continuous-valued label (as in the linear regression example above). For a given $p \in \mathbb{N}$, fix the (parametrized) loss function $f : [0, 1]^d \times \mathbb{R}^p \rightarrow \mathbb{R}$, where $\theta = (\theta_1, \ldots, \theta_p) \in \mathbb{R}^p$ is a $p$-dimensional parameter vector. Now define the ERM objective function $F : \mathbb{R}^p \rightarrow \mathbb{R}$ as follows:

$$\forall \theta \in \mathbb{R}^p, \quad F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)}; \theta)$$

(3)

which is the empirical risk, i.e., the empirical expectation of the loss function with respect to the training data $\mathcal{D}$. With this objective function, we consider the unconstrained minimization problem:

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

(4)

where $F_\ast \in \mathbb{R}$ denotes the minimum value. In contrast to the canonical formulation (1), our formulation in (3) and (4) explicitly states the dependence of the loss function on the data.

Under the smoothness and strong convexity assumptions delineated in Section 1.3, we make the following main contributions:

1. We propose a new inexact gradient descent algorithm, Local Polynomial Interpolation based Gradient Descent (LPI-GD), to compute approximate solutions of (4) in Section 2.2.1 (see Algorithm 1). The main innovation of this algorithm is to use the smoothness of the loss function $f$ in the data to learn the gradient oracle of $f$ at every iteration by performing local polynomial regression based on oracle queries over a set of virtual data points.

2. We derive the iteration and oracle complexities of this algorithm in Proposition 2 and Theorem 2, respectively, in Section 2.2.2. In particular, we show that the oracle complexity scales like $O((pe^{-1})^{d/2n})$ (neglecting sub-dominant factors), where $\eta > 0$ denotes the Hölder exponent determining the smoothness of the gradient of the loss function with respect to the data (see Section 1.3). Although we focus on strongly convex loss functions to leverage simple convergence results on inexact gradient descent from the literature, analogous analysis could be carried out for larger classes of loss functions, e.g., when $f(x; \cdot) : \mathbb{R}^p \rightarrow \mathbb{R}$ is non-convex.

3. Furthermore, we show in Proposition 3 in Section 2.2.2 that if the data dimension is sufficiently small, e.g., $d = O(\log \log(n))$ (as in healthcare data analytics or control applications; see Section 1.1), and the loss function $f$ exhibits modest smoothness in the data, i.e., $\eta = \Theta(d)$, then the oracle complexity for our method beats the oracle complexities (or more precisely, the oracle complexity bounds) of both GD and SGD for any $p = O(\text{poly}(n))$ and any $\epsilon = \Theta(\text{poly}(n^{-1/2}))$. In particular, when we let $\epsilon = \Theta(n^{-1/2})$ have the smallest scaling of interest (as explained earlier) and $p = O(n^{\beta})$ for any $\beta > 0$, Table 1 depicts that the oracle complexities of GD, SGD, and LPI-GD are $O(n \log(n))$, $O(n^{\beta + 1/2})$, and $O(n^{\delta})$ for any small $\delta > 0$ (when the loss function is smooth enough), respectively. This demonstrates that smoothness of loss functions in the data can be successfully exploited in gradient-based algorithms for ERM to significantly lower their oracle complexity.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$O(n \log(n))$ (Nesterov 2004)</td>
</tr>
<tr>
<td>SGD</td>
<td>$O(n^{3/2})$ (Bottou et al. 2018)</td>
</tr>
<tr>
<td>LPI-GD</td>
<td>$O(n^3)$ (Theorem 2)</td>
</tr>
</tbody>
</table>

Table 1 Oracle complexities of GD, SGD, and our LPI-GD algorithm for strongly convex loss functions: Here, we assume that the approximation accuracy $\epsilon = \Theta(n^{-1/2})$, the parameter dimension $p = O(n^3)$ for any constant $\beta > 0$, and the data dimension $d = O(\log \log(n))$, where $n$ is the number of training samples. Moreover, $\delta > 0$ is a very small constant of our choice (when the loss function is commensurately smooth with respect to the data).
Finally, in order to analyze the convergence rate of our algorithm, we also generalize the technique of local polynomial regression, or more precisely, local polynomial interpolation (since there is no noise in our problem), in non-parametric statistics to the multivariate setting in Theorem 1 in Section 2.1. Although it is implied in many standard expositions that univariate local polynomial regression can be generalized to multiple variables (see, e.g., Fan and Gijbels 1996, Tsybakov 2009, or Wasserman 2019), the multivariate case requires significantly more care than the univariate case. Consequently, a detailed and rigorous analysis of local polynomial interpolation is difficult to find in the literature. Therefore, we believe that Theorem 1 may be of independent interest in statistics.

The outline of the remainder of this work is as follows. We discuss relevant related literature in Section 1.1 that compares and contrasts contributions of this work. We introduce required notation and definitions, formally state key assumptions, and briefly elaborate on the standard framework of oracle complexity in optimization theory in Sections 1.2 to 1.4, respectively. In Section 2, we introduce the LPI-GD algorithm and present our main results. We prove our results on local polynomial regression in Section 3, and carry out the convergence analysis of LPI-GD in Section 4. Finally, we conclude our discussion and propose future research directions in Section 5. Furthermore, we present known oracle complexity results for GD and SGD in Appendix A.

1.1. Related Literature

There is an enormous literature concerning the analysis of convergence rates of exact and inexact gradient-based algorithms that perform the optimization in (1). It is not feasible to survey this entire literature here, and instead we only mention some closely related works. For example, in our strongly convex setting of interest, the convergence analysis of GD can be found in the standard text (Nesterov 2004), and corresponding convergence analysis of SGD can be found in (Nemirovski et al. 2009) (also see Jain and Kar 2017, Bottou et al. 2018, Netrapalli 2019, and the references therein). Moreover, various refinements and improvements of these basic iterative algorithms have been analyzed in the literature. Notable examples include the accelerated or momentum based GD (Nesterov 1983), mini-batch SGD methods (Dekel et al. 2012), and variance reduced SGD methods (Johnson and Zhang 2013) (also see the references therein). We also refer readers to (Bubeck 2015) for a unified treatment of many of the aforementioned approaches. More generally, inexact gradient descent methods for ERM have been analyzed extensively as well—see, e.g., Friedlander and Schmidt 2012, So and Zhou 2017, and the references therein. As mentioned earlier, to our knowledge, all of this literature concerns the formulation in (1). Hence, these methods do not exploit smoothness of loss functions in data by learning gradients in every iteration as in our proposed method, LPI-GD.

On the other hand, motivated by the utility of gradient information for various tasks like variable selection and determining coordinate covariations (cf. Mukherjee and Zhou 2006, Mukherjee and Wu 2006), the problem of learning gradients in the context of supervised learning problems, such as classification and regression, has received a lot of attention from the machine learning community. Specifically, most of this literature aims to simultaneously learn a classification or regression function along with its gradient from training data. For example, the text (Fan and Gijbels 1996) presents the classical theory of gradient estimation using local polynomial fitting, and De Brabanter et al. (2013) present a gradient estimation approach that uses local polynomial regression, but does not estimate the regression function. In a related vein, Delecroix and Rosa (1996) study kernel methods for gradient estimation. Various other methods for learning gradients have also been analyzed, such as reproducing kernel Hilbert space methods (Mukherjee and Zhou 2006, Mukherjee and Wu 2006), regression splines (Zhou and Wolfe 2000), and nearest neighbor methods (Ausset
et al. 2020). However, the vast majority of these techniques have not been considered in the context of optimization for machine learning.

Two recent exceptions where gradient estimation has been applied to optimization are (Wang et al. 2018) and (Ausset et al. 2020). In particular, Wang et al. (2018) illustrate that learning gradients leads to desirable convergence rates for derivative-free optimization algorithms, which only use zeroth order oracles (i.e., these oracles only output function values, not gradient values). Similarly, Ausset et al. (2020) portray the effectiveness of their nearest neighbor based gradient estimator for derivative-free optimization. It is worth noting that both these works analyze optimization of a single objective function (rather than a sum as in (1)). Hence, the results derived in these papers are not particularly insightful in our ERM setting in (3) and (4), where we seek to learn gradients based on a first order oracle by exploiting the smoothness of $f(x; \theta)$ in the data $x$.

Lastly, as we remarked earlier, we establish in Proposition 3 that the oracle complexity of our algorithm beats GD and SGD for a broad range of values of $p$ and $\epsilon$, if the data dimension $d$ is sufficiently small, e.g., $d = O(\log(\log(n)))$. Since we utilize multivariate local polynomial regression in our algorithm, a “small $d$ assumption” is theoretically unavoidable due to the curse of dimensionality (cf. Fan and Gijbels 1996, Section 7.1). We close this discussion of related literature by further explaining why this assumption is also very reasonable in applications. Indeed, many application domains have problems where the number of data attributes is small. For example, in the healthcare data analytics space, electronic health records often have very few features per patient, e.g., temperature, blood pressure, initial diagnosis, and medications, but machine learning techniques are still widely used to build models for diagnosing diseases or predicting risk (cf. Choi et al. 2018, and its references). As another example, in the problem of system identification for robotics and control applications, neural networks have been used to build non-linear state-space models (cf. Nechyba and Xu 1994, Nelles 2001, and their references). The number of inputs of these networks is often very small when the model order is known or assumed to be low. On the other hand, in many applications with high-dimensional “raw data,” data scientists believe that the useful information in the data lives in a low-dimensional manifold. This belief is the basis of a myriad of dimensionality reduction techniques, such as principal component analysis (Pearson 1901, Hotelling 1933), canonical correlation analysis (Hotelling 1936), Laplacian eigenmaps (Belkin and Niyogi 2001) or diffusion maps (Coifman and Lafon 2006), and modal decompositions (or correspondence analysis), cf. (Huang et al. 2019), (Makur 2019, Chapter 4), and the references therein. As a result, many high-dimensional data analysis pipelines proceed by first pre-processing the data using pertinent dimensionality reduction techniques, and then using the resulting low-dimensional features for other learning tasks. In such high-dimensional settings, our contributions in this paper pertain to ERM using the extracted low-dimensional features.

1.2. Notation
We briefly introduce some notation that is used throughout this work. Let $\mathbb{N} \triangleq \{1,2,3,\ldots\}$ and $\mathbb{Z}_+ \triangleq \{0,1,2,3,\ldots\}$ denote the sets of natural numbers and non-negative integers, respectively. As mentioned earlier, for any $n \in \mathbb{N}$, let $[n] \triangleq \{1,\ldots,n\}$ be the set of positive integers less than $n+1$. In the sequel, we analyze a $d$-variate problem setting with $d \in \mathbb{N}$. So, we briefly introduce the standard multi-index notation. For any $d$-tuple of non-negative integers $s = (s_1, \ldots, s_d) \in \mathbb{Z}_+^d$, we let $|s| = s_1 + \cdots + s_d$, $s! = s_1! \cdots s_d!$, and $x^s = x_1^{s_1} \cdots x_d^{s_d}$ for $x \in \mathbb{R}^d$. Moreover, given a continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, $f(x_1, \ldots, x_d)$, we write $\nabla_s f = \partial^{|s|} f / (\partial x_1^{s_1} \cdots \partial x_d^{s_d})$ to denote its $s$th partial derivative, and $\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_d} \end{bmatrix}^T$ to denote its gradient. We will sometimes find it convenient to index certain vectors or matrices using the set $\{s \in \mathbb{Z}_+^d : |s| \leq l\}$. In these cases, to help with calculations, we will assume that these vectors or matrices are written out with the indices in lexicographical order (i.e., for any $r, s \in \mathbb{Z}_+^d$ with $r \neq s$, we say that $r < s$ if $r_j < s_j$ for the first position $j \in [d]$ where $r_j \neq s_j$).

Furthermore, for any vector $x \in \mathbb{R}^n$ with $n \in \mathbb{N}$, we let $x_i$ be the $i$th entry of $x$, where $i \in [n]$ (or more generally, $i$ belongs to an appropriate index set), and we let $\|x\|_p$ be the usual $\ell^p$-norm of
Moreover, as additional notation, we let $\sigma \longmapsto R$ for every $g \in \mathcal{F}$ be any small constant representing the desired level of approximation accuracy. We begin by imposing the following assumptions on our loss function.

We make two pertinent remarks at this point. Firstly, it can be argued based on the above strong convexity assumption (as shown in Section 4.1) that $F^*$ is finite and achievable by a unique global

1.3. Smoothness Assumptions and Approximate Solutions

In order to perform tractable convergence analysis of gradient-based iterative algorithms that solve (4), we must formally define the problem class by specifying any smoothness conditions on the loss function, the appropriate notion of “approximate solutions,” and the oracle model of computational complexity used (Nesterov 2004, Section 1.1). This section and Section 1.4 precisely define our problem class.

Recall our formulation where we are given $n$ samples of $d$-dimensional training data $D$ and a loss function $f : [0,1]d \times \mathbb{R}^p \to \mathbb{R}$, $f(x, \theta)$, and we consider the ERM problem in (3) and (4). Let $\epsilon > 0$ be any small constant representing the desired level of approximation accuracy. We begin by imposing the following assumptions on our loss function:

1. **Smoothness in parameter**: There exists $L_1 > 0$ such that for all fixed $x \in [0,1]^d$, the gradient of $f$ with respect to $\theta$, denoted $\nabla_\theta f(x; \cdot) : \mathbb{R}^p \to \mathbb{R}^p$, exists and is $L_1$-Lipschitz continuous as a function of the parameter vector:

   $$\forall \theta_1, \theta_2 \in \mathbb{R}^p, \|\nabla_\theta f(x; \theta_1) - \nabla_\theta f(x; \theta_2)\|_2 \leq L_1 \|\theta_1 - \theta_2\|_2.$$

2. **Smoothness in data**: There exist $\eta > 0$ and $L_2 > 0$ such that for all fixed $\theta \in \mathbb{R}^p$ and for all $i \in [d]$, the $i$th partial derivative of $f$ (with respect to $\theta_i$) at $\theta$, denoted $g_i(\cdot; \theta) \triangleq \partial_\theta f(x; \theta) : [0,1]^d \to \mathbb{R}$, belongs to the $(\eta, L_2)$-Hölder class as a function of the data vector $x$ (cf. Tsybakov 2009, Definition 1.2). By definition, this means that each $g_i(\cdot; \theta) : [0,1]^d \to \mathbb{R}$ is $l = \lceil \eta \rceil - 1$ times differentiable, and for every $s = (s_1, \ldots, s_d) \in \mathbb{Z}_+^d$ such that $|s| = l$, we have

   $$\forall y_1, y_2 \in [0,1]^d, |\nabla^s g_i(y_1; \cdot) - \nabla^s g_i(y_2; \cdot)| \leq L_2 \|y_1 - y_2\|_1^{\eta - l}.$$

3. **Strong convexity**: There exists $\mu > 0$ such that for all fixed $x \in [0,1]^d$, the map $f(x; \cdot) : \mathbb{R}^p \to \mathbb{R}$ is $\mu$-strongly convex as a function of the parameter vector:

   $$\forall \theta_1, \theta_2 \in \mathbb{R}^p, f(x; \theta_1) \geq f(x; \theta_2) + \nabla_\theta f(x; \theta_2)^T(\theta_1 - \theta_2) + \frac{\mu}{2} \|\theta_1 - \theta_2\|^2_2.$$

Moreover, as additional notation, we let $\sigma \triangleq L_1 / \mu$ represent the condition number of the problem.

Under these assumptions, we define an $\epsilon$-approximate solution to the optimization problem (4) as any parameter vector $\theta^* \in \mathbb{R}^p$ that satisfies

$$F(\theta^*) - F_* \leq \epsilon.$$

We make two pertinent remarks at this point. Firstly, it can be argued based on the above strong convexity assumption (as shown in Section 4.1) that $F_*$ is finite and achievable by a unique global
minimizer of $F$; in particular, the infimum in (4) may be replaced by a minimum. Secondly, both the first and third assumptions are standard in the optimization literature (cf. Nesterov 2004, Bubeck 2015). As noted earlier, our main idea is the observation that the second assumption, that gradients of loss functions are also smooth in the data, holds in many machine learning applications. For example, in the regularized linear regression example in (2), $\nabla_\theta f$ is clearly smooth with respect to the data (assuming $\mathcal{R}$ is differentiable).

1.4. Oracle Complexity
Finally, we briefly introduce the formalism of “first order oracle complexity” upon which our analysis will be based. The notion of oracle complexity was first introduced in (Nemirovski˘ı and Yudin 1983) to circumvent the difficulties faced by vanilla computational complexity theory style analysis of convex programming (cf. Agarwal et al. 2009, Section 1). As elucidated in (Nesterov 2004, Section 1.1), although “arithmetical complexity,” which accounts for the running time of all basic arithmetic operations executed by an algorithm, is a more realistic measure of computational complexity, the idea of oracle (or “analytical”) complexity, i.e., the total number of oracle calls made by an algorithm, is a more convenient abstraction for the analysis of iterative optimization methods. Indeed, the running time of oracle queries is often the main bottleneck in arithmetical complexity.

Observe that any iterative algorithm that approximately solves optimization problems belonging to the class of ERM problems defined in (3) and (4), under the assumptions in Section 1.3, requires two inputs: a) the training data $\mathcal{D}$, and b) a “description” of the specific loss function $f$ under consideration. To fulfill the latter requirement, we will assume that such algorithms have access to a local first order oracle $O : [0,1]^d \times \mathbb{R}^p \to \mathbb{R}^p$ defined as:

$$\forall x \in [0,1]^d, \forall \vartheta \in \mathbb{R}^p, \quad O(x, \vartheta) \triangleq \nabla_\theta f(x; \vartheta)$$

(9)

which returns the value of the gradient $\nabla_\theta f$ with respect to $\theta$ for any input query pair $(x, \vartheta)$. (Note that sometimes the oracle is defined to produce values of both the loss function $f$ and its gradient $\nabla_\theta f$ at $(x, \vartheta)$ (cf. Nesterov 2004, Section 1.1.2).) This oracle $O$ can be construed as a black box “description” of the gradient of the loss function $f$ with respect to $\theta$.

Given any particular iterative algorithm, $\mathcal{A} = \mathcal{A} (\mathcal{D}, O)$, that solves our class of ERM problems, its first order oracle complexity (or just oracle complexity) $\Gamma (\mathcal{A})$ is formally defined as the minimum number of oracle calls made by the algorithm $\mathcal{A}$ to obtain an $\epsilon$-approximate solution in the sense of (8) (Nesterov 2004, Section 1.1). Thus, $\Gamma (\mathcal{A})$ is typically a function of the notable problem variables $n, d, p, \eta, \epsilon$, and the fixed constants $L_1, L_2, \mu$. In this paper, following the paradigm of the optimization literature, we will focus on determining the first order oracle complexity of our proposed optimization method.

2. Main Results and Discussion
In this section, we first present a generalization of local polynomial regression to the multivariate case in Section 2.1, and then describe our proposed algorithm LPI-GD and its convergence analysis in Section 2.2.

2.1. Local Polynomial Regression
We commence by introducing the essentials of local polynomial regression specialized to our noiseless setting. Since most references on the subject only present the univariate (i.e., $d = 1$) framework, e.g., (Fan and Gijbels 1996), (Tsybakov 2009, Section 1.6), we rigorously carry out the much more involved calculations for the multivariate case by carefully generalizing the univariate development in (Tsybakov 2009, Section 1.6). We also refer readers to (Cleveland and Loader 1996, Section 2) for classical references and a brief history of smoothing using local regression.

Consider a function $g : [0,1]^d \to \mathbb{R}$ that belongs to the $([\eta], L_2)$-Hölder class, namely, $g$ is $l = [\eta] - 1$ times differentiable and satisfies (6) for every $s \in \mathbb{Z}_+^d$ with $|s| = l$. In order to present the local
polynomial regression based approximation of this function, we first state some required definitions and notation. Let $K : [-1, 1] \to [b, c]$ be a bounded kernel function, where $c \geq b > 0$ are some fixed constants, and we will use the extension of this kernel (to the domain $\mathbb{R}$), $K : \mathbb{R} \to [0, c]$, in the sequel with the assumption that the support of $K$ is $[-1, 1]$. For any fixed $m \in \mathbb{N}$, define the uniform grid

$$G_m \triangleq \{ u \in [0, 1]^d : \forall i \in [d], u_i m \in [m] \},$$

which has cardinality $|G_m| = m^d$. Finally, define the vector(-valued function) $U : \mathbb{R}^d \to \mathbb{R}^D$ as:

$$\forall u \in \mathbb{R}^d, \quad U(u) \triangleq \left[ \frac{u^s}{s!} : s \in \mathbb{Z}_+, |s| \leq l \right]^T,$$

which has dimension $D \triangleq \dim(U) = \sum_{l=0}^{d} \binom{k+d-1}{d-1} = (l+1)^d$ (via the hockey-stick identity). Then, for any arbitrarily small value of the bandwidth parameter $h \in (0, \frac{1}{2})$, we can write the following weighted regression problem:

$$\forall x \in [h, 1-h]^d, \quad \Phi(x) \triangleq \arg \min_{\phi \in \mathbb{R}^D} \sum_{y \in G_m} \left( g(y) - \phi^T U \left( \frac{y-x}{h} \right) \right) \prod_{j=1}^{d} K \left( \frac{y_j-x_j}{h} \right),$$

where $\Phi(x) = [\Phi_1(x) \cdots \Phi_D(x)]^T$. We refer to the first entry $\hat{\phi}(x) \triangleq \Phi_1(x) = U(0)^T \Phi(x)$ of $\Phi(x)$ as the local polynomial interpolator of order $l$ for $g : [h, 1-h]^d \to \mathbb{R}$ (which is $g : [0, 1]^d \to \mathbb{R}$ restricted to the domain $[h, 1-h]^d$). As we shall see, $\hat{\phi}$ provides a “good” approximation of the function $g$ in the supremum (or Chebyshev) norm.

For any $x \in [h, 1-h]^d$, construct the $D \times D$ symmetric matrix

$$B(x) \triangleq \frac{1}{(mh)^d} \sum_{y \in G_m} U \left( \frac{y-x}{h} \right) U \left( \frac{y-x}{h} \right)^T \prod_{j=1}^{d} K \left( \frac{y_j-x_j}{h} \right),$$

which is also positive semidefinite due to the non-negativity of the kernel. We next present a key technical proposition which generalizes (Tsybakov 2009, Lemma 1.5) to the $d$-variate setting and demonstrates that $B(x)$ is actually positive definite.

**Proposition 1 (Löwner Lower Bound).** Suppose that $l \geq d \geq 19$. Then, for all $m \in \mathbb{N}$ and $h \in (0, \frac{1}{2})$ such that $mh \geq 4l(3e)^d / \Lambda(d,l)$ and all $x \in [h, 1-h]^d$, we have

$$B(x) \geq \frac{b^d \Lambda(d,l)}{2} I_D,$$

where the constant $\Lambda(d,l) > 0$ is defined as

$$\Lambda(d,l) \triangleq \left( \frac{\pi d}{8e^2} \right)^d \left( \left( \frac{l+d}{d} \right)^d - 1 \right)^{(l+d)^d-1} \frac{1}{(l+d)^{3l(l+d)/d} d^3}.$$

Proposition 1 is established in Section 3.1 by exploiting several basic ideas from the theory of orthogonal polynomials and matrix perturbation theory. The proof is much more sophisticated than (Tsybakov 2009, Lemma 1.5) because explicit quantitative estimates are derived to deal with general $d$. Since Proposition 1 shows that $B(x)$ is invertible, a straightforward exercise in calculus conveys that the unique solution $\hat{\Phi}(x)$ of the weighted regression problem in (12) can be written as:

$$\forall x \in [h, 1-h]^d, \quad \hat{\Phi}(x) = \sum_{y \in G_m} g(y) w_y(x),$$

where
where the optimal vector-valued weights $w_y(x) \in \mathbb{R}^D$ are given by:

$$\forall y \in G_m, \forall x \in [h, 1-h]^d, \quad w_y(x) = \frac{1}{(mh)^d} \left( \prod_{j=1}^{d} K \left( \frac{y_j - x_j}{h} \right) \right) B(x)^{-1} U \left( \frac{y-x}{h} \right).$$  \hspace{1cm} (16)

Then, defining the first coordinates of these vectors as the real-valued interpolation weights:

$$\forall y \in G_m, \forall x \in [h, 1-h]^d, \quad w^*_y(x) = [w_y(x)]_1,$$

the local polynomial interpolator of order $l$ for $g : [h, 1-h]^d \to \mathbb{R}$ can be written as the following weighted sum of the interpolation points $\{g(y) : y \in G_m\}$:

$$\forall x \in [h, 1-h]^d, \quad \hat{\phi}(x) = \sum_{y \in G_m} g(y) w^*_y(x).$$  \hspace{1cm} (18)

In Section 3.2, we present some important properties of the interpolation weights given in (17) based on the key estimate in Proposition 1.

The ensuing theorem conveys that local polynomial regression can be used to uniformly approximate a function $g$ (in the supremum norm sense) for sufficiently large uniform grids $G_m$.

**Theorem 1 (Supremum Norm Interpolation).** Fix any constant $\delta \in (0, 1)$, and consider the function $g : [0, 1]^d \to \mathbb{R}$ which belongs to the $(\eta, L_2)$-Hölder class. Suppose that $l = \lceil \eta \rceil - 1 \geq d \geq 19$, and that the grid size $|G_m| = m^d$ is sufficiently large so that

$$m \geq \frac{110(2L_2 + 1)c}{b} \left( \frac{d(3e)^d}{\Lambda(d,l)^2} \right) \left( \frac{1}{\delta} \right)^{1/\eta},$$

where $c \geq b > 0$ are the bounds on our kernel, and $\Lambda(d,l)$ is defined in (14). Then, the supremum norm approximation error of $g$ is bounded by $\delta$, viz.

$$\sup_{x \in [h, 1-h]^d} \left| \hat{\phi}(x) - g(x) \right| \leq \delta,$$

where $\hat{\phi}$ is the local polynomial interpolator of order $l$ for $g$ given in (18), and the bandwidth $h \in (0, \frac{1}{2})$ is chosen to be

$$h = \frac{4l(3e)^d}{m\Lambda(d,l)} \leq \frac{2b}{55(2L_2 + 1)c} \left( \frac{\Lambda(d,l)l}{d} \right).$$

Theorem 1 is proved in Section 3.3 using the pertinent properties of the interpolation weights in (17) derived in Section 3.2. We note that the lower bound on $m$ is expected to scale at least exponentially with the dimension $d$ due to the curse of dimensionality (cf. Fan and Gijbels 1996, Section 7.1). However, it may be possible to improve the dependence we have above using a different analysis. In the sequel, we will exploit Theorem 1 to perform inexact gradient descent.

### 2.2. Inexact Gradient Descent using Local Polynomial Interpolation

We now turn to describing and analyzing a new inexact gradient descent algorithm that performs local polynomial interpolation at every iteration. In particular, we delineate the algorithm in Section 2.2.1, and derive convergence rates in Section 2.2.2.
2.2.1. Description of Algorithm. Under the setup and assumptions described in Section 1, suppose we seek to solve the optimization problem (4) in the ε-approximate sense of (8) based on some training data $D$ with the parameter $h' > 0$ given by

$$h' = \frac{2b}{55(2L_2 + 1)c} \left( \frac{\Lambda(d, l)}{d} \right) = O\left( \frac{\Lambda(d, l)}{d} \right),$$  \hspace{1cm} (19)$$

which is the upper bound on the bandwidth $h$ in Theorem 1.\footnote{We remark that as long as $h' = O(\Lambda(d, l)/d)$, we can choose a kernel $K$ in the sequel such that (19) holds.} We first present our proposed algorithm, dubbed Local Polynomial Interpolation based Gradient Descent (LPI-GD), in general, and then specialize the values of different problem variables with judicious choices later on.

Let $\theta^{(t)} \in \mathbb{R}^p$ denote the “approximate solution” produced by LPI-GD at iteration $t \in \mathbb{Z}_+$, and let $\delta_t \in (0, 1)$ denote the maximum allowable supremum norm approximation error at iteration $t \in \mathbb{N}$. Fix any bounded kernel $K : [-1, 1] \rightarrow [b, c]$ with $c \geq b > 0$ as in Section 2.1, and corresponding to each $\delta_t$ at iteration $t \in \mathbb{N}$, consider the uniform grid $G_{m_t}$, where $m_t \in \mathbb{N}$ is given by Theorem 1:

$$m_t = \left\lceil \frac{110(2L_2 + 1)c}{b} \left( \frac{d(3e)^d}{\Lambda(d, l)} \right) \left( \frac{1}{\delta_t} \right)^{1/n} \right\rceil, \hspace{1cm} (20)$$

and the bandwidth $h_t \in (0, \frac{1}{2})$ is also given by Theorem 1:

$$h_t = \frac{4l(3e)^d}{m_t\Lambda(d, l)} \leq h'. \hspace{1cm} (21)$$

We initialize $\theta^{(0)} \in \mathbb{R}^p$ to be any arbitrary vector in $\mathbb{R}^p$ at iteration $t = 0$. Then, at iteration $t \in \mathbb{N}$, we make $|G_{m_t}| = m_t^d$ first order oracle calls at all virtual data points in the uniform grid $G_{m_t}$ with the approximate solution at iteration $t - 1$:

$$\forall y \in G_{m_t}, \hspace{0.5cm} O(y, \theta^{(t-1)}) = \nabla_{\theta} f(y; \theta^{(t-1)}). \hspace{1cm} (22)$$

For every $i \in [p]$, these oracle calls give us information about the partial derivatives $\{g_i(y; \theta^{(t-1)}) : y \in G_{m_t}\}$ with respect to $\theta_i$. Hence, akin to (12) and (18) in Section 2.1, we can construct the local polynomial interpolator $\hat{\phi}^{(t)} : [h', 1 - h']^d \rightarrow \mathbb{R}$ of order $l = \lceil \eta \rceil - 1$ for the $(\eta, L_2)$-Hölder class function $g_i(\cdot; \theta^{(t-1)}) = \frac{\partial f}{\partial \theta_i}(\cdot; \theta^{(t-1)}} : [h', 1 - h']^d \rightarrow \mathbb{R}$ as follows:

$$\forall x \in [h', 1 - h']^d, \hspace{0.5cm} \hat{\phi}^{(t)}_i(x) = \sum_{y \in G_{m_t}} g_i(y; \theta^{(t-1)}) w^{*}_{y, i}(x), \hspace{1cm} (23)$$

where the interpolation weights $w^{*}_{y, i}(x) \in \mathbb{R}$ are defined analogously to (17); in particular, they are the first elements of the weight vectors $w_{y, i}(x) \in \mathbb{R}^D$, which are defined as in (16), but with $G_m$ replaced by the grid $G_{m_t}$ and $h$ replaced by the bandwidth $h_t$. Using the learned interpolators in (23) at iteration $t$, we next approximate the true gradient of the ERM objective function $\nabla_{\theta} F(\theta^{(t-1)})$ at $\theta^{(t-1)}$ as follows. For every $i \in [p]$ and every $j \in [n]$, we first estimate the partial derivative $g_i(x^{(j)}; \theta^{(t-1)})$ at the data sample $x^{(j)}$ by computing $\hat{\phi}^{(t)}_i(x^{(j)})$. Then, since we have $\nabla_{\theta} F(\theta^{(t-1)}) = \frac{1}{n} \sum_{j=1}^{n} \nabla_{\theta} f(x^{(j)}; \theta^{(t-1)})$ due to (3), we can estimate $\nabla_{\theta} F(\theta^{(t-1)})$ by computing

$$\hat{\nabla} F^{(t)} = \frac{1}{n} \sum_{j=1}^{n} \left[ \hat{\phi}^{(t)}_i(x^{(j)}) : i \in [p] \right]^T, \hspace{1cm} (24)$$
where \([\hat{g}^{(t)}_i(x^{(j)}): i \in [p]]^T \in \mathbb{R}^p\) is our approximation of \(\nabla \theta f(x^{(j)}; \theta^{(t-1)})\) at the data sample \(x^{(j)}\). Finally, we perform the inexact gradient descent update

\[
\hat{\theta}^{(t)} = \hat{\theta}^{(t-1)} - \frac{1}{L_1} \hat{F}^{(t)},
\]

which produces the approximate solution for iteration \(t\). We run the above procedure for \(T \in \mathbb{N}\) iterations, and the LPI-GD algorithm returns \(\hat{\theta}^{(T)} \in \mathbb{R}^p\) as the \(\epsilon\)-approximate solution for the optimization problem (4). The analysis in Section 2.2.2 elucidates the specific values taken by the constants \(\delta\), and the total number of iterations \(T\) in terms of the other problem parameters. In particular, we will see that the approximation errors \(\delta\), grid sizes \(m\), bandwidths \(h\), and uniform grids \(G_{m, h}\) all will not change over iterations. A pseudocode summary of the LPI-GD algorithm with pertinent \(\delta\) and \(T\) (selected as in Section 2.2.2) is presented in Algorithm 1; the algorithm uses the training data \(D\) and a first order oracle \(O\) to generate an \(\epsilon\)-approximation solution that solves (4).

We remark that Algorithm 1 portrays a “theoretical” algorithm, because it assumes that problem parameters, such as the Lipschitz constant \(L_1\), the strong convexity parameter \(\mu\), the minimum value \(F_*\), and the Hölder exponent \(\eta\) and constant \(L_2\), are known to the practitioner. When some or all of these parameters are unknown, we can use appropriate bounds and estimates based on the loss function \(f: [0,1]^d \times \mathbb{R}^p \to \mathbb{R}\). For example, we may use any upper bound on \(L_1\), any lower bound on \(\mu\), and any universal lower bound on \(f\) (which yields a lower bound on \(F_*\)). Moreover, for smooth functions \(f\), we can choose any desired level of differentiability \(\eta\) and any upper bound on \(L_2\). When such bounds or estimates are also unavailable, the best we can do is to select \(T\), \(\delta\), \(m\), and \(h\) to have the correct scaling with respect to \(\epsilon\), \(d\), \(p\), and \(\eta\). It is worth mentioning that our objective in this paper is to demonstrate that the oracle complexities of GD and SGD can be theoretically beaten by the proposed algorithm. So, we do not delve further into considerations regarding its practical implementation here.

Before proceeding to our analysis, it is worth juxtaposing Algorithm 1 with standard batch GD and SGD. At each iteration, GD exactly computes gradients for all data samples using \(n\) oracle calls. Since this can be computationally expensive, inexact gradient descent methods attempt to reduce the number of oracle calls per iteration at the cost of increasing the number of iterations. For example, SGD makes only one oracle call per iteration for a randomly chosen data sample, and uses this queried gradient value as a proxy for the true gradient. In contrast, at each iteration, our approach evaluates the gradients at certain representative virtual points in the data space, and uses these oracle calls to learn a model for the gradient oracle. This learnt model is then used to approximate the true gradient. Intuitively, when the loss function is a sufficiently smooth function of the data, we can learn an accurate model with a reasonably small number of virtual points.

2.2.2. Analysis of Algorithm. To compute the oracle complexity of the LPI-GD algorithm, we must first compute the minimum number of iterations it takes to obtain an \(\epsilon\)-approximate solution to (4). The ensuing proposition chooses appropriate values of \(\delta\), which turn out to be constant with respect to \(t\), and develops an upper bound on the minimum number of iterations \(T\).

**Proposition 2 (Iteration Complexity of LPI-GD).** Suppose that \(l = [\eta] - 1 \geq d \geq 19\) and the assumptions of Section 1.3 hold. For any (small) accuracy \(\epsilon > 0\), define the number of iterations \(T \in \mathbb{N}\) as

\[
T = \left\lceil \left( \log \left( \frac{\sigma}{\sigma - 1} \right) \right)^{-1} \log \left( \frac{F(\theta(0)) - F_* + \frac{p}{2\mu}}{\epsilon} \right) \right\rceil,
\]

and the maximum allowable supremum norm approximation errors \(\{\delta_t \in (0,1) : t \in [T]\}\) via:

\[
\forall t \in [T], \quad \delta_t^2 = \left( 1 - \frac{1}{\sigma} \right)^T \Theta \left( \frac{\epsilon}{p} \right),
\]

where \([\hat{\theta}^{(t)}_i(x^{(j)}) : i \in [p]]^T \in \mathbb{R}^p\) is our approximation of \(\nabla \theta f(x^{(j)}; \theta^{(t-1)})\) at the data sample \(x^{(j)}\). Finally, we perform the inexact gradient descent update

\[
\hat{\theta}^{(t)} = \hat{\theta}^{(t-1)} - \frac{1}{L_1} \hat{F}^{(t)},
\]

which produces the approximate solution for iteration \(t\). We run the above procedure for \(T \in \mathbb{N}\) iterations, and the LPI-GD algorithm returns \(\hat{\theta}^{(T)} \in \mathbb{R}^p\) as the \(\epsilon\)-approximate solution for the optimization problem (4). The analysis in Section 2.2.2 elucidates the specific values taken by the constants \(\delta\), and the total number of iterations \(T\) in terms of the other problem parameters. In particular, we will see that the approximation errors \(\delta\), grid sizes \(m\), bandwidths \(h\), and uniform grids \(G_{m, h}\) all will not change over iterations. A pseudocode summary of the LPI-GD algorithm with pertinent \(\delta\) and \(T\) (selected as in Section 2.2.2) is presented in Algorithm 1; the algorithm uses the training data \(D\) and a first order oracle \(O\) to generate an \(\epsilon\)-approximation solution that solves (4).

We remark that Algorithm 1 portrays a “theoretical” algorithm, because it assumes that problem parameters, such as the Lipschitz constant \(L_1\), the strong convexity parameter \(\mu\), the minimum value \(F_*\), and the Hölder exponent \(\eta\) and constant \(L_2\), are known to the practitioner. When some or all of these parameters are unknown, we can use appropriate bounds and estimates based on the loss function \(f: [0,1]^d \times \mathbb{R}^p \to \mathbb{R}\). For example, we may use any upper bound on \(L_1\), any lower bound on \(\mu\), and any universal lower bound on \(f\) (which yields a lower bound on \(F_*\)). Moreover, for smooth functions \(f\), we can choose any desired level of differentiability \(\eta\) and any upper bound on \(L_2\). When such bounds or estimates are also unavailable, the best we can do is to select \(T\), \(\delta\), \(m\), and \(h\) to have the correct scaling with respect to \(\epsilon\), \(d\), \(p\), and \(\eta\). It is worth mentioning that our objective in this paper is to demonstrate that the oracle complexities of GD and SGD can be theoretically beaten by the proposed algorithm. So, we do not delve further into considerations regarding its practical implementation here.

Before proceeding to our analysis, it is worth juxtaposing Algorithm 1 with standard batch GD and SGD. At each iteration, GD exactly computes gradients for all data samples using \(n\) oracle calls. Since this can be computationally expensive, inexact gradient descent methods attempt to reduce the number of oracle calls per iteration at the cost of increasing the number of iterations. For example, SGD makes only one oracle call per iteration for a randomly chosen data sample, and uses this queried gradient value as a proxy for the true gradient. In contrast, at each iteration, our approach evaluates the gradients at certain representative virtual points in the data space, and uses these oracle calls to learn a model for the gradient oracle. This learnt model is then used to approximate the true gradient. Intuitively, when the loss function is a sufficiently smooth function of the data, we can learn an accurate model with a reasonably small number of virtual points.

2.2.2. Analysis of Algorithm. To compute the oracle complexity of the LPI-GD algorithm, we must first compute the minimum number of iterations it takes to obtain an \(\epsilon\)-approximate solution to (4). The ensuing proposition chooses appropriate values of \(\delta\), which turn out to be constant with respect to \(t\), and develops an upper bound on the minimum number of iterations \(T\).

**Proposition 2 (Iteration Complexity of LPI-GD).** Suppose that \(l = [\eta] - 1 \geq d \geq 19\) and the assumptions of Section 1.3 hold. For any (small) accuracy \(\epsilon > 0\), define the number of iterations \(T \in \mathbb{N}\) as

\[
T = \left\lceil \left( \log \left( \frac{\sigma}{\sigma - 1} \right) \right)^{-1} \log \left( \frac{F(\theta(0)) - F_* + \frac{p}{2\mu}}{\epsilon} \right) \right\rceil,
\]

and the maximum allowable supremum norm approximation errors \(\{\delta_t \in (0,1) : t \in [T]\}\) via:

\[
\forall t \in [T], \quad \delta_t^2 = \left( 1 - \frac{1}{\sigma} \right)^T \Theta \left( \frac{\epsilon}{p} \right),
\]
Algorithm 1: Local Polynomial Interpolation based Gradient Descent (LPI-GD) algorithm to approximately solve (4).

**Input:** \( n \) samples of training data \( \mathcal{D} = \{ x^{(i)} \in [h', 1 - h']^d : i \in [n] \} \)

**Input:** first order oracle \( \mathcal{O} : [0,1]^d \times \mathbb{R}^p \rightarrow \mathbb{R}^p \) defined in (9)

**Input:** approximation accuracy level \( \epsilon > 0 \)

**Output:** \( \epsilon \)-approximate solution \( \theta^* \in \mathbb{R}^p \) satisfying (8)

**Step 1: Initialization**
1. Set arbitrary initial parameter vector \( \theta^{(0)} \in \mathbb{R}^p \)
2. Set number of iterations \( T \in \mathbb{N} \) according to (26)
3. Set supremum norm approximation error \( \delta = (1 - \sigma^{-1})^{T/2} \quad \triangleright \) This is the same for all iterations; see (27)
4. Set grid size \( m \in \mathbb{N} \) according to (20) \quad \triangleright \) Substitute \( m_t = m \) and \( \delta_t = \delta \) in (20)
5. Set bandwidth \( h = 4l(3\epsilon)^d/(m\Lambda(d,l)) \) \quad \triangleright \) See (21)

**Step 2: Calculation of interpolation weights**
6. Construct uniform grid \( \mathcal{G}_m \) according to (10)
7. for \( j = 1 \) to \( n \) do \quad \triangleright \) Iterate over training data
8. \( \quad \quad \) Compute interpolation weights \( \{ w_j^*(x^{(i)}) : y \in \mathcal{G}_m \} \) via (16) and (17) using kernel \( K : [-1,1] \rightarrow [b,c] \)
9. end for

**Step 3: Inexact gradient descent**
10. for \( t = 1 \) to \( T \) do \quad \triangleright \) Iterate over training data
11. \( \quad \quad \) Make oracle calls \( \{ \mathcal{O}(y, \theta^{(t-1)}) : y \in \mathcal{G}_m \} \) to get partial derivatives \( \{ \frac{\partial \ell}{\partial \theta}(y; \theta^{(t-1)}) : y \in \mathcal{G}_m, i \in [p] \} \)
12. \( \quad \quad \) Compute local polynomial interpolators \( \{ \phi_j^{(t)}(x^{(i)}) : i \in [p] \} \) according to (23) using the oracle queries \( \{ \frac{\partial \ell}{\partial \theta}(y; \theta^{(t-1)}) : y \in \mathcal{G}_m, i \in [p] \} \) and precomputed interpolation weights \( \{ w_j^*(x^{(i)}) : y \in \mathcal{G}_m \} \)
13. end for
14. Construct estimate of gradient \( \nabla \mathcal{F}^{(t)} \) using interpolators \( \{ \hat{\phi}_j^{(t)}(x^{(i)}) : i \in [p], j \in [n] \} \) as shown in (24)
15. Update: \( \theta^{(t)} \leftarrow \theta^{(t-1)} - \frac{1}{t \epsilon} \nabla \mathcal{F}^{(t)} \) \quad \triangleright \) See (25)
16. end for
17. return \( \theta^* = \theta^{(T)} \)

where \( F : \mathbb{R}^p \rightarrow \mathbb{R} \) is the ERM objective function in (3), and \( \sigma = L_1/\mu > 1 \) is the condition number defined in Section 1.3. Then, after \( T \) iterations of the LPI-GD algorithm described in Algorithm 1 in Section 2.2.1 with arbitrary initialization \( \theta^{(0)} \in \mathbb{R}^p \) and updates (25), the parameter vector \( \theta^{(T)} \in \mathbb{R}^p \) is an \( \epsilon \)-approximate solution to the optimization problem in (4) in the sense of (8):

\[
F(\theta^{(T)}) - F_* \leq \epsilon.
\]

Proposition 2 is proved in Section 4.1 using Theorem 1 and some known ideas from the analysis of inexact gradient descent methods in Friedlander and Schmidt (2012). Using Proposition 2, the ensuing theorem bounds the first order oracle complexity of the LPI-GD algorithm in Algorithm 1, denoted \( \Gamma(\text{LPI-GD}) \), which is the minimum number of oracle calls made by Algorithm 1 to obtain an \( \epsilon \)-approximate solution (as defined in Section 1.4).

**Theorem 2 (Oracle Complexity of LPI-GD).** Suppose that \( l = \lceil \eta \rceil - 1 \geq d \geq 19 \) and the assumptions of Section 1.3 hold. For any (small) accuracy \( 0 < \epsilon \leq (L_2/\mu)^d \), consider the LPI-GD algorithm described in Algorithm 1 in Section 2.2.1 with arbitrary initialization \( \theta^{(0)} \in \mathbb{R}^p \), number of iterations \( T \in \mathbb{N} \) as in (26), maximum allowable approximation errors \( \{ \delta_t \in (0,1) : t \in [T] \} \) as in (27), and updates (25). Then, the first order oracle complexity of this algorithm (to obtain an \( \epsilon \)-approximate solution in the sense of (8)) is upper bounded by

\[
\Gamma(\text{LPI-GD}) \leq C_{\mu, L_1, L_2, b, c}(d,l) \left( \frac{p + 2\mu(F(\theta^{(0)}) - F_*)}{\epsilon} \right)^{d/(2m)} \log \left( \frac{p + 2\mu(F(\theta^{(0)}) - F_*)}{2\mu\epsilon} \right)
\]
where we let
\[
C_{\mu,L_1,L_2,b,c}(d,l) \triangleq \left( \frac{\sigma + 2(L_1 - \mu)}{(L_1 - \mu) \log \left( \frac{\sigma}{\sigma - 1} \right)} \right) \left( \frac{220(2L_2 + 1)c}{b} \right)^d \left( \frac{d^4(3c)^d}{\Lambda(d,l)^{2d}} \right),
\] (28)
where \( \Lambda(d,l) \) is defined in (14), and \( \mu, L_1, L_2, b, c \) are constants defined in Sections 1.3 and 2.1.

Theorem 2 is established in Section 4.2. Unlike the “dimension-free” oracle complexity bounds for batch GD and SGD, our bound in Theorem 2 explicitly depends on \( d \) and \( p \). However, even though it may not be immediately obvious, the oracle complexity of LPI-GD scales much better than the oracle complexities of GD or SGD in many important regimes. We next elucidate a scaling law for the bound in Theorem 2 in the setting where \( n, d, p, \eta, \epsilon \rightarrow \infty \) and all other problem parameters are constant.

As in Appendix A, let \( \theta^* \in \mathbb{R}^p \) be the unique global minimizer of the empirical risk \( F : \mathbb{R}^p \rightarrow \mathbb{R} \) in (3) so that \( F^* = F(\theta^*) \). Assume that the parameter \( \theta \) belongs to an \( \ell^\infty \)-ball with constant radius (as in many applications). Then, in the aforementioned setting, we have using (Bubeck 2015, Lemma 3.4) that
\[
F(\theta^{(0)}) - F^* \leq \frac{L_1}{2} \| \theta^{(0)} - \theta^* \|_2^2 = O(p),
\] (29)
where the inequality uses the facts that \( \nabla_\theta F : \mathbb{R}^p \rightarrow \mathbb{R}^p \) is \( L_1 \)-Lipschitz continuous (see the proof of Proposition 2 in Section 4.1) and \( \nabla_\theta F(\theta^*) = 0 \) (which is the zero vector), and the equality holds because \( \| \theta^{(0)} - \theta^* \|_2^2 = O(p) \) (see Appendix A). Hence, the first order oracle complexity of the LPI-GD algorithm in Theorem 2 scales as
\[
\Gamma(\text{LPI-GD}) = O \left( C_{\mu,L_1,L_2,b,c}(d,l) \left( \frac{p}{\epsilon} \right)^{d/(2\eta)} \log \left( \frac{p}{\epsilon} \right) \right).
\] (30)

For comparison, recall from Proposition 4 in Appendix A (cf. Nesterov 2004, Theorem 2.1.15) that when the empirical risk is strongly convex and has Lipschitz continuous gradient, the first order oracle complexity of GD is bounded by
\[
\Gamma(\text{GD}) \leq \Gamma_*(\text{GD}) \triangleq \Theta \left( n \log \left( \frac{p}{\epsilon} \right) \right).
\] (31)
Similarly, recall from Proposition 5 in Appendix A (cf. Nemirovski et al. 2009 and Bottou et al. 2018, Theorem 4.7) that when the empirical risk is strongly convex and has Lipschitz continuous gradient, and certain universal second moment bounds are also satisfied on stochastic gradients, the first order oracle complexity of SGD is bounded by
\[
\Gamma(\text{SGD}) \leq \Gamma_*(\text{SGD}) \triangleq \Theta \left( \frac{p}{\epsilon^2} \right).
\] (32)
Note that the expressions in (31) and (32) do not depend on \( d, \eta, \) and neglect all other constant parameters.

To compare the first order oracle complexities in (30), (31), and (32), we will assume for simplicity that the quantities \( d, p, \eta, \epsilon^{-1} \) are all functions of the number of training samples \( n \), and study the asymptotics as \( n \rightarrow \infty \). The ensuing proposition presents a broad scaling regime where the oracle complexity of LPI-GD beats the oracle complexities (or more precisely, the oracle complexity bounds) of both GD and SGD shown above.

**Proposition 3 (Comparison to GD and SGD).** Fix any constants \( \alpha > 0, \beta > 0, \tau > \max\{1,(\alpha + \beta)^{-1}\}, \) and \( \gamma > \max\{1,\tau(\alpha + \beta)/2\} \). Suppose that \( \epsilon = \Theta(n^{-\alpha}), \ p = O(n^\beta), \ d \leq \)
\[ \log \log(n) \left( \frac{c(\gamma+1)}{4 \log(e(\gamma+1))} \right), \text{ and } \eta = \Theta(d) \text{ such that } \eta \geq \tau(\alpha + \beta)d/2 \text{ and } d \leq \lfloor \eta \rfloor - 1 \leq \gamma d. \] Furthermore, suppose that \( \mu, L_1, L_2, b, c \) are \( \Theta(1) \) with respect to \( n \). Then, we have

\[ \Gamma(LPI-GD) = O\left( \log(n) \exp\left(2\sqrt{\log(n)}\right) n^{1/\tau} \right), \]

\[ \Gamma^*(GD) = \Theta(n \log(n)), \]

\[ \Gamma^*(SGD) = \Theta(n^{\alpha + \beta}). \]

This implies that

\[ \lim_{n \to \infty} \frac{\Gamma(LPI-GD)}{\Gamma^*(GD)} = \lim_{n \to \infty} \frac{\Gamma(LPI-GD)}{\Gamma^*(SGD)} = 0. \]

Proposition 3 is proved in Section 4.3. Under its assumptions, the LPI-GD algorithm beats any \( \Theta(\text{poly}(n)) \) oracle complexity when \( \tau \) is chosen to be large enough. Hence, one can also calculate interesting regimes where LPI-GD scales better than more refined gradient-based algorithms such as accelerated GD (Nesterov 1983), mini-batch SGD (Dekel et al. 2012), variance reduced SGD (Johnson and Zhang 2013), etc. We do not explicitly carry out these calculations here for brevity. Furthermore, as explained in Appendix A, comparisons of oracle complexities of GD and SGD in the literature typically neglect the factors of \( p \) in (31) and (32) (see our discussion at the outset of Section 1). In contrast, Proposition 3 performs comparisons with these additional factors, because they stem from a reasonable assumption that the parameter lives in an \( \ell^\infty \)-ball. However, since the LPI-GD algorithm can beat any \( \Theta(\text{poly}(n)) \) oracle complexity, the oracle complexity of LPI-GD would scale better than the oracle complexities of GD or SGD in the broad regime outlined in Proposition 3 even if we neglected the factors of \( p \) in GD and SGD.

We close this section by making several remarks. Firstly, many of the refined gradient-based methods mentioned above lead to improvements in the dependence of oracle complexity bounds on the condition number, e.g., variance reduced SGD (Johnson and Zhang 2013) (which has a similar oracle complexity expression as GD, but with better dependence on condition number). We do not consider or optimize the trade-off with condition number in our analysis; this is in many ways an orthogonal question. As indicated at the outset of this paper, our focus has been on beating GD and SGD’s oracle complexities with regard to their dependence on \( n, \epsilon, d, \) and \( p \).

Secondly, while methods like SGD or mini-batch SGD improve upon GD’s oracle complexity (in terms of its trade-off between \( n \) and \( \epsilon \), neglecting implicit dependencies on \( p \)), they are stochastic optimization algorithms that only generate \( \epsilon \)-approximate solutions on expectation. In contrast, the LPI-GD algorithm can beat GD and is completely deterministic.

Thirdly, since the LPI-GD algorithm has to learn the oracle at every iteration, the computational (or “arithmetical”) complexity of its iterations may seem to be higher than usual iterative methods. However, the motivation for focusing on the concept of oracle complexity in optimization theory is partly that computing gradients can be extremely hard for certain functions. So, the computational complexity of the oracle usually dominates any other reasonable computation done during iterations. We, therefore, adhere to this canonical paradigm in this work (Nesterov 2004). If the precise running time of the first order oracle \( O \) is known, then the complete computational complexity of the LPI-GD algorithm can be derived from Proposition 2 and Theorem 2 by accounting for the running time of the \( D \times D \) matrix inversions used to compute the interpolation weights. Note that as shown in Algorithm 1, these weights only need to be pre-computed once before running iterations of inexact GD.

Fourthly, we emphasize that Proposition 3 holds in the important regime where \( \alpha = \frac{1}{2} \) and \( \beta > 1 \). As noted in Section 1, \( \alpha = \frac{1}{2} \) is typically the smallest accuracy of interest since it produces the scaling of the statistical error of the empirical risk. (Table 1 presents the oracle complexities computed in Proposition 3 in this regime.) The latter condition, \( \beta > 1 \), ensures that the number of parameters \( p \) can be much larger than the number of training samples \( n \), which corresponds to the popular overparametrized regime considered in the context of deep neural networks (cf. Poggio et al. 2020).
Finally, we note that the dependence of the bounds in Theorem 2 and Proposition 3 on both \(p\) and \(d\) could potentially be improved. The dependence on \(p\) arises because Theorem 1 holds for functions with codomain \(\mathbb{R}\). However, the proof of Proposition 2 in Section 4.1 needs to bound the \(\ell^2\)-norm error between (24) and the true gradient \(\nabla_\theta F(\theta^{(t-1)})\) for \(t \in \mathbb{N}\). Translating the coordinatewise guarantee to an \(\ell^2\)-norm guarantee in this proof introduces the dependence on \(p\) (see (59)). If some version of Theorem 1 could be proved for functions with codomain \(\mathbb{R}^p\), then we could possibly improve the (explicit) dependence of Theorem 2 on \(p\).

On the other hand, the dependence on \(d\) could be improved by tightening the minimum eigenvalue lower bound in Lemma 1 as explained in Section 3.1. Nevertheless, this eigenvalue lower bound must decay at least exponentially (and correspondingly, the grid size parameter \(m\) must grow at least exponentially) in \(d\), as noted after Theorem 1 in Section 2.1. Thus, it is intuitively straightforward to see that the best scaling of \(d\) we can hope for, so that the conclusion of Proposition 3 remains true, is \(d = O(\log(n))\). This logarithmic scaling of data dimension \(d\) with number of training samples \(n\) is significant. Indeed, for many data science applications of interest, the useful information in a dataset of \(n\) high-dimensional feature vectors is contained within the pairwise (Euclidean) distances between different feature vectors. In these scenarios, standard low-dimensional approximate isometric embedding theorems, e.g., the Johnson-Lindenstrauss lemma (cf. Dasgupta and Gupta 2003, and references therein), ensure that the feature vectors can be mapped to a \(d = O(\log(n))\) dimensional space such that the pairwise distances are preserved. We leave the development of our results along these directions as future work.

3. Proofs for Local Polynomial Interpolation

In this section, we provide proofs for the results presented in Section 2.1. Specifically, we first establish Proposition 1, then portray some properties of the interpolation weights in (17), and finally prove Theorem 1.

3.1. Proof of Proposition 1

To prove Proposition 1, we require the following intermediate lemma which generalizes (Tsybakov 2009, Lemma 1.4) to our multivariate setting.

**Lemma 1 (Auxiliary Löwner Lower Bound).** When \(l \geq d \geq 19\), the \(D \times D\) symmetric matrix

\[
B \triangleq \int_{[-1,1]^d} U(u)U(u)^T \, du
\]

satisfies the Löwner lower bound

\[
B \succeq \Lambda(d,l) I_D,
\]

where \(U : \mathbb{R}^d \to \mathbb{R}^D\) is defined in (11), and the constant \(\Lambda(d,l) > 0\) is defined in (14).

**Proof.** We begin by establishing some simple facts about \(B\). Following the proof of (Tsybakov 2009, Lemma 1.4) mutatis mutandis, note that \(B\) is positive semidefinite:

\[
\forall v \in \mathbb{R}^D, \quad v^T B v = \int_{[-1,1]^d} (v^T U(u))^2 \, du \geq 0.
\]

Suppose for the sake of contradiction that there exists a non-zero vector \(v \in \mathbb{R}^D\) such that

\[
v^T B v = \int_{[-1,1]^d} (v^T U(u))^2 \, du = 0.
\]

Then, we must have \(v^T U(u) = 0\) almost everywhere (with respect to the Lebesgue measure) on \([-1,1]^d\). However, since the non-zero polynomial \(u \mapsto v^T U(u)\) is analytic, its set of roots has Lebesgue measure zero (see, e.g., Krantz and Parks 2002, Section 4.1). This is a contradiction.
So, we actually have $v^T B v > 0$ for all non-zero vectors $v \in \mathbb{R}^D$. Thus, $B$ is positive definite. From hereon, our proof deviates greatly from that of (Tsybakov 2009, Lemma 1.4).

For convenience in the sequel, we compute the entries of $B$. Since the rows and columns of $B$ are indexed by \( \{ s \in \mathbb{Z}^d_+ : |s| \leq l \} \), for any $r,s \in \mathbb{Z}^d_+$ with $|r|,|s| \leq l$, we have

\[
B_{r,s} = \int_{[-1,1]^d} \frac{u^{r+s}}{r!s!} \, du \\
= \frac{2^d}{r!s!} \int_{[-1,1]^d} \frac{1}{2^d} \prod_{j=1}^d u_j^{r_j+s_j} \, du \\
= \frac{2^d}{r!s!} \prod_{j=1}^d \mathbb{E} \left[ X_j^{r_j+s_j} \right] \\
= \{ r + s \text{ is even entry-wise} \} \frac{2^d}{r!s!} \prod_{j=1}^d \frac{1}{r_j + s_j + 1} \\
= \{ r + s \text{ is even entry-wise} \} \frac{2^d}{r!s!} \prod_{j=1}^d \frac{1}{r_j + s_j + 1} \\
\tag{33}
\]

where $X_1, \ldots, X_d \in [-1,1]$ are independent and identically distributed (i.i.d.) uniform random variables, and the $k$th moment of $X_1$ is $\mathbb{E}[X_1^k] = 0$ for all odd $k \in \mathbb{Z}_+$ and $\mathbb{E}[X_1^k] = (k+1)^{-1}$ for all even $k \in \mathbb{Z}_+$ (cf. Forbes et al. 2011, Chapter 40). Thus, $B$ is a doubly non-negative matrix with all rational entries.

We next lower bound the strictly positive minimum eigenvalue of $B$. To this end, we first upper bound the trace of $B$, and then lower bound the determinant of $B$. To upper bound $\text{tr}(B)$, observe that

\[
\text{tr}(B) = \sum_{s \in \mathbb{Z}^d_+ : |s| \leq l} B_{s,s} \\
= 2^d \sum_{s \in \mathbb{Z}^d_+ : |s| \leq l} \frac{1}{(s!)^2} \prod_{j=1}^d \frac{1}{2s_j + 1} \\
\leq 2^d \sum_{k=0}^l \frac{1}{k!} \sum_{s \in \mathbb{Z}^d_+ : |s| = k} \frac{k!}{s!} \\
= 2^d \sum_{k=0}^l \frac{d^k}{k!} \\
\leq (2e)^d, \tag{34}
\]

where the second equality uses (33), the fourth equality follows from the multinomial theorem, and the fifth inequality uses the Maclaurin series of the exponential function. (Note that the exponential scaling of $\text{tr}(B)$ is tight, because $\text{tr}(B) \geq B_{(0,\ldots,0),(0,\ldots,0)} = 2^d$.)

To lower bound $\det(B)$, we will first compute it exactly. To do this, we introduce the Legendre polynomials, \( \{ L_k : [-1,1] \to \mathbb{R} \ | \ k \in \mathbb{Z}_+ \} \), which are defined by Rodrigues’ formula (Chihara 2011, Chapter V, Equation (2.1)):

\[
\forall t \in [-1,1], \quad L_k(t) \triangleq \sqrt{\frac{k+\frac{1}{2}}{2^k k!}} \frac{d^k}{dt^k} (t^2 - 1)^k, \tag{35}
\]
where \( L_k \) is a polynomial with degree \( k \). It is well-known that the family of Legendre polynomials satisfies the orthonormality property (Chihara 2011, Chapter I, Exercise 1.6):

\[
\forall j, k \in \mathbb{Z}_+, \quad \int_{-1}^{1} L_j(t)L_k(t) \, dt = \mathbb{1}\{j = k\}. \quad (36)
\]

Moreover, for each \( k \in \mathbb{Z}_+ \), the leading coefficient (of \( t^k \)) of \( L_k(t) \) is known to be (Chihara 2011, Chapter V, Equation (2.7)):

\[
\sqrt{\frac{k + \frac{1}{2}}{2k}} \binom{2k}{k}. \quad (37)
\]

Next, for any \( r \in \mathbb{Z}_+^d \), define the Legendre product polynomial:

\[
\forall u \in [-1, 1]^d, \quad L_r(u) \triangleq \prod_{j=1}^d L_{r_j}(u_j) \quad (38)
\]

which has degree \( r \) (with abuse of notation). It is straightforward to verify that these product polynomials also satisfy an orthonormality condition:

\[
\forall r, s \in \mathbb{Z}_+^d, \quad \int_{[-1,1]^d} L_r(u)L_s(u) \, du = \prod_{j=1}^d \int_{-1}^{1} L_{r_j}(u_j)L_{s_j}(u_j) \, du_j = \mathbb{1}\{r = s\}, \quad (39)
\]

where we utilize (36). Moreover, for each \( r \in \mathbb{Z}_+^d \), the leading coefficient (of \( u^r \)) of \( L_r(u) \) is known to be

\[
\prod_{j=1}^d \sqrt{\frac{r_j + \frac{1}{2}}{2r_j}} \binom{2r_j}{r_j} = \frac{1}{2^{|r|}} \prod_{j=1}^d \sqrt{\frac{r_j + \frac{1}{2}}{2r_j}}, \quad (40)
\]

where we utilize (37). Due to (39), we know that for any \( r \in \mathbb{Z}_+^d \), the set \( \{L_s : s \in \mathbb{Z}_+^d, s \leq r \} \) is linearly independent and spans the set of all \( d \)-variate polynomials with degree at most \( r \). (Note that we use the lexicographical order over indices in \( \mathbb{Z}_+^d \) here. Furthermore, it is straightforward to verify that all \( d \)-variate monomials with non-zero coefficients that appear in \( L_s \) have degree at most \( s \).) As a result, defining a vector(-valued) function \( \mathcal{L} : \mathbb{R}^d \rightarrow \mathbb{R}^D \) as:

\[
\forall u \in [-1,1]^d, \quad \mathcal{L}(u) \triangleq \left[L_s(u) : s \in \mathbb{Z}_+^d, \, |s| \leq l \right]^T, \quad (41)
\]

which is indexed in lexicographical order, we have the linear relation:

\[
\forall u \in [-1,1]^d, \quad U(u) = R\mathcal{L}(u) \quad (42)
\]

for some fixed matrix \( R \in \mathbb{R}^{D \times D} \) (that does not depend on \( u \)). It is evident that this matrix \( R \) is lower triangular due to the lexicographical order we have imposed. Furthermore, it is easy to derive the diagonal entries of \( R \). Indeed, for any index \( s \in \mathbb{Z}_+^d \) with \( |s| \leq l \), equating the leading coefficients on both sides of (42) produces

\[
\frac{1}{s!} = R_{s,s} \frac{1}{2^{|s|}} \prod_{j=1}^d \sqrt{\frac{2s_j + 1}{2s_j}} \binom{2s_j}{s_j} \]

\( ^3 \) Alternatively, the Legendre polynomials can be obtained by applying the Gram-Schmidt process to the monomials \( 1, t, t^2, \ldots \) with respect to the standard inner product, using a three-term recurrence relation, solving Legendre’s differential equation, or expanding certain generating functions (Chihara 2011).
using (40). Rearranging this, we get:

$$\forall s \in \mathbb{Z}_+^d \text{ with } |s| \leq l, \quad R_{s,s} = \frac{2^{|s|} 2^{d/2}}{s!} \prod_{j=1}^{d} \frac{1}{\sqrt{2s_j + 1}} \left(\frac{2s_j}{s_j}\right)^{-1}. \quad (43)$$

Now notice that

$$B = \int_{[-1,1]^d} U(u)U(u)^T du$$

$$= R \left( \int_{[-1,1]^d} \mathcal{L}(u)\mathcal{L}(u)^T du \right) R^T$$

$$= R I_{d} R^T$$

$$= R R^T \quad (44)$$

where the second equality follows from (42), and the third equality follows from (39). We recognize (44) as the Cholesky decomposition of $B$ (Horn and Johnson 2013, Corollary 7.2.9). Thus, using (44), we can write

$$\det(B) = \prod_{s \in \mathbb{Z}_+^d : |s| \leq l} R_{s,s}^2$$

$$\geq \prod_{s \in \mathbb{Z}_+^d : |s| \leq l} \begin{cases} 4^{|s|} \prod_{i=1}^{d} \left( \frac{2s_i}{s_i} \right), & \text{if } |s| \leq 1 \\ \frac{2^d}{(s!)^2} \prod_{j=1}^{d} \frac{1}{2s_j + 1}, & \text{if } |s| > 1 \end{cases} = B_{s,s}$$

$$\geq \prod_{s \in \mathbb{Z}_+^d : |s| \leq l} \left( \frac{s_1! \cdots s_d!}{2^{|s|}|s|!} \right)^2$$

$$= \left( \frac{\pi}{2} \right)^{dd} \left( \prod_{s \in \mathbb{Z}_+^d : |s| \leq l} \frac{1}{2^{|s|}|s|!} \right)^2$$

$$= \left( \frac{\pi}{2} \right)^{dd} \frac{1}{4^{\sum_{l=0}^{d-1} \sum_{j=0}^{d-l} 1}} \left( \prod_{j=0}^{l} \prod_{s \in \mathbb{Z}_+^d : |s| = j} \frac{1}{s_1! \cdots s_d!} \right)^2$$

$$\geq \left( \frac{\pi}{4e} \right)^{dd} \frac{1}{4} \sum_{l=0}^{d-1} \sum_{j=0}^{d-l} \prod_{j=0}^{l} \prod_{s \in \mathbb{Z}_+^d : |s| = j} \frac{1}{(s_1 + 1)^{2s_1+1} \cdots (s_d + 1)^{2s_d+1}}$$

$$\geq \left( \frac{\pi}{4e} \right)^{dd} \prod_{j=0}^{l} \prod_{s \in \mathbb{Z}_+^d : |s| = j} \frac{1}{(s_1 + 1)^{2s_1+1} \cdots (s_d + 1)^{2s_d+1}}$$

$$\geq \left( \frac{\pi}{4e} \right)^{dd} \frac{1}{(j + d)^{d(j+1)^2}}$$

$$\geq \left( \frac{\pi d}{4e} \right)^{dd} \frac{1}{(l + d)^{3d}} \quad (45)$$
where the second equality follows from (43), the third inequality follows from the following bound (on Catalan numbers; see (Dutton and Brigham 1986, Theorem)):

$$\forall j \in \mathbb{Z}^+, \left(\frac{2j}{j}\right)^2 < \frac{4^{2j+1}}{\pi(4j+1)},$$

the fifth equality follows from adding the exponents $|s|$ of $2$ over all $s \in \mathbb{Z}^d_+$ with $|s| \leq l$ and then squaring, the sixth inequality follows from the Stirling’s formula bound (see, e.g., Feller 1968, Chapter II, Section 9, Equation (9.15)):

$$\forall j \in \mathbb{Z}^+, \ j! \leq (j+1)^{j+\frac{1}{2}}e^{-j+1},$$

the seventh inequality follows from the straightforward bound:

$$\sum_{i=0}^l i\left(\frac{i+d-1}{d-1}\right) \geq l\left(\frac{l+d-1}{d-1}\right) = \frac{dl}{l+d}\left(\frac{l+d}{d}\right) \geq \frac{d}{2}\left(\frac{l+d}{d}\right) = \frac{dD}{2}$$

which uses the fact that $l \geq d$, the eighth inequality uses the following basic facts and calculations:

1. The hypercube maximizes the volume over all $d$-orthotopes with an isoperimetric constraint:

$$\max_{s_1,\ldots,s_d \geq 0: \ s_1 + \cdots + s_d = j} (s_1 + 1) \cdots (s_d + 1) = \left(\frac{j}{d} + 1\right)^d$$

where $j \in \mathbb{Z}^+$ and the maximum is achieved by $s_1 = \cdots = s_d = \frac{j}{d}$.

2. For any $j \in \mathbb{Z}^+$ and any $s \in \mathbb{Z}^d_+$ with $|s| = j$, the vector $(j, 0, \ldots, 0) \in \mathbb{Z}^d_+$ majorizes the vector $s$, and since the map $0 \leq t \mapsto 2t \log(t+1)$ is convex, Karamata’s majorization inequality yields (cf. Marshall et al. 2011, Chapter 1, Section A)

$$\sum_{i=1}^d 2s_i \log(s_i + 1) \leq 2j \log(j+1),$$

which implies that

$$\prod_{i=1}^d (s_i + 1)^{2s_i} \leq (j+1)^{2j}$$

using the monotonicity of the exponential function;

and the tenth inequality follows from the simple bound:

$$\sum_{j=0}^l (2j+d)\left(\frac{j+d-1}{d-1}\right) \leq (2l+d)\sum_{j=0}^l \left(\frac{j+d-1}{d-1}\right) = (2l+d)D \leq 3lD$$

which uses the fact that $l \geq d$.

Finally, recall the following well-known result (cf. Merikoski and Virtanen 1997, Lemma 1):

$$\lambda_{\min}(\mathcal{B}) \geq \det(\mathcal{B}) \left(\frac{D-1}{\text{tr}(\mathcal{B})}\right)^{D-1},$$

(46)

Note that the usual upper bound of Stirling’s approximation is $j! \leq e\sqrt{j(j/e)^j}$ for all $j \in \mathbb{N}$. In order to include the case $j = 0$, we loosen this bound in our analysis.

This is a simple consequence of the arithmetic mean–geometric mean (AM–GM) inequality.
which utilizes the positive definiteness of $B$ established earlier. Combining (34), (45), and (46), we obtain

$$\lambda_{\min}(B) \geq \left(\frac{\pi d}{4e}\right)^{dD} \frac{(D-1)^{D-1}}{(l+d)^{3MD}(2e)^{d(D-1)}}$$

$$\geq \left(\frac{\pi d}{8e^2}\right)^{dD} \frac{(D-1)^{D-1}}{(l+d)^{3MD}}$$

$$\geq \left(\frac{\pi d}{8e^2}\right)^{d\left(\frac{l+d}{d}\right)^d} \left(\frac{l+d}{d}\right)^d \left(\frac{1}{d} \right)^{(l+d)^d} \frac{1}{(l+d)^d} \left(\frac{e(l+d)}{d}\right)^d$$

$$= \Lambda(d,l)$$

(47)

where the third inequality holds because $d \geq 19$, the map $1 \leq t \mapsto t^d$ is strictly increasing, and we utilize the ensuing standard bounds on binomial coefficients:

$$\left(\frac{l+d}{d}\right)^d \leq D = \left(\frac{l+d}{d}\right)^d \leq \left(\frac{e(l+d)}{d}\right)^d,$$

and the final equality follows from (14). Since $B \succeq \lambda_{\min}(B)I_D$, applying the lower bound in (47) completes the proof.

We make two pertinent remarks at this point. Firstly, we note that when $d \leq l \leq \gamma d$ for some constant $\gamma > 1$, one can verify that

$$\Lambda(d,l) \geq \left(\frac{\pi d}{8e^2}\right)^{d^2d} \frac{(2^d - 1)^{2^{d-1}}}{(\gamma + 1)d)^{2\gamma d(e(\gamma + 1))^d}}.$$

(48)

Hence, $\Lambda(d,l)^{-1}$ scales double exponentially in $d$:

$$\frac{1}{\Lambda(d,l)} = O\left(e^{\Theta(\text{poly}(d))}\right).$$

(49)

This observation will be used in the proof of Proposition 3 in Section 4.3. Secondly, we also conjecture that the scaling in (49) can be significantly improved.

**Conjecture 1 (Scaling of Minimum Eigenvalue of $B$).** If $l = \Theta(\text{poly}(d))$, then we have

$$\frac{1}{\lambda_{\min}(B)} = \Theta\left(e^{\Theta(\text{poly}(d))}\right).$$

Indeed, using the Courant-Fischer-Weyl min-max theorem (Horn and Johnson 2013, Theorem 4.2.6) (also see the Schur-Horn theorem (Horn and Johnson 2013, Theorem 4.3.45)), the minimum eigenvalue of $B$ is upper bounded by the minimum diagonal entry of $B$ as follows:

$$\lambda_{\min}(B) \leq \min_{s \in \mathbb{Z}_+^d, |s| \leq l} \frac{2^d}{(s!)^2} \prod_{j=1}^d \frac{1}{2s_j + 1}$$

$$\leq \frac{2^d}{(2l+1)(l!)^2}$$

$$\leq \frac{2^{d-1}e^{2l}}{\pi(2l+1)|l|^{2l+1}}$$

6 This bound can also be proved using the AM–GM inequality.
where the second inequality follows from setting $s_1 = l$ and $s_2 = \cdots = s_d = 0$ (which is reasonable because it asymptotically achieves the minimum), and the third inequality follows from the Stirling’s formula bound (cf. Feller 1968, Chapter II, Section 9, Equation (9.15)). Thus, $\lambda_{\min}(B)^{-1}$ has the potential to scale as
\begin{equation}
\frac{1}{\lambda_{\min}(B)} \geq \frac{\pi^{2l+2}}{2^{2l-2}e^{2l}} = \Omega(e^{\Theta(poly(d))})
\end{equation}
when $l = \Theta(poly(d))$. As shown in Conjecture 1, we believe that this scaling is likely to be the actual scaling of $\lambda_{\min}(B)^{-1}$. Since we exactly evaluate the trace and determinant in the proof of Lemma 1, the looseness of our minimum eigenvalue lower bound in Lemma 1 stems from the AM–GM based result in (46). Hence, a fruitful future direction would be to obtain tighter bounds on the minimum eigenvalue in terms of trace and determinant that are still sufficiently tractable so as to permit analytical evaluation.

We are now in a position to establish Proposition 1 using Lemma 1.

Proof of Proposition 1. We take inspiration from the proof technique of (Tsybakov 2009, Lemma 1.5), but the details of our analysis are more involved and quite different. Observe that for all $m \in \mathbb{N}$, for all $x \in [h, 1-h]^d$, and for all $v \in \mathbb{R}^d$ with $\|v\|_2 = 1$, we have
\begin{align*}
v^T B(x)v &= v^T \left( \frac{1}{(mh)^d} \sum_{y \in \mathcal{G}_m} U \left( \frac{y - x}{h} \right) U \left( \frac{y - x}{h} \right)^T \prod_{j=1}^d K \left( \frac{y_j - x_j}{h} \right) \right) v \\
&= v^T \left( \frac{1}{(mh)^d} \sum_{z \in [-1,1]^d : x+hz \in \mathcal{G}_m} U(z) U(z)^T \prod_{j=1}^d K(z_j) \right) v \\
&\geq b^d v^T \tilde{B}(x)v
\end{align*}
where the first equality follows from (13), the second equality follows from the substitution $y = x + h z$ and the fact that the kernel $K$ has support $[-1, 1]$, the third inequality holds because the kernel $K$ is lower bounded by $b > 0$ on $[-1, 1]$, and we define $\tilde{B}(x)$ as the $D \times D$ symmetric positive semidefinite matrix
\begin{equation}
\tilde{B}(x) \triangleq \sum_{z \in [-1,1]^d : x+hz \in \mathcal{G}_m} U(z) U(z)^T \frac{1}{(mh)^d}.
\end{equation}
Then, applying the Courant-Fischer-Weyl min-max theorem (Horn and Johnson 2013, Theorem 4.2.6), we obtain
\begin{equation}
\lambda_{\min}(B(x)) \geq b^d \lambda_{\min}(\tilde{B}(x)).
\end{equation}
We proceed to lower bounding $\lambda_{\min}(\tilde{B}(x))$. Fix any $x \in [h, 1-h]^d$, and any $r, s \in \mathbb{Z}_+^d$ with $|r|, |s| \leq l$, and consider the Riemann sum
\begin{equation}
\tilde{B}(x)_{r,s} = \frac{1}{r!s!} \sum_{z \in [-1,1]^d : x+hz \in \mathcal{G}_m} z^{r+s} \frac{1}{(mh)^d}.
\end{equation}
Since $x \in [h, 1-h]^d$, it is easy to verify that the points $\{z \in [-1,1]^d : x+hz \in \mathcal{G}_m\}$ partition the hypercube $[-1,1]^d$ into a grid of $d$-cells $\{I_z \subseteq [-1,1]^d : z \in [-1,1]^d, x+hz \in \mathcal{G}_m\}$ with side lengths $(mh)^{-1}$ and volume $(mh)^{-d}$ each, where $I_z$ is the $d$-cell with tag $z$. Moreover, since the polynomial function $\gamma : [-1,1]^d \to \mathbb{R}$, $\gamma(u) = u^{r+s}$ is Riemann integrable, if $h = \omega(m^{-1})$ with all other problem
parameters fixed, then the above Riemann sum converges to an entry of the matrix $\mathcal{B}$ defined in Lemma 1 (cf. Apostol 1974, Section 14.3):

$$\lim_{m \to \infty} \frac{1}{r!s!} \sum_{z \in [-1, 1]^d \cap \mathbb{G}_m} \frac{z^{r+s}}{(mh)^d} = \int_{[-1, 1]^d} \frac{u^{r+s}}{r!s!} \, du = \mathcal{B}_{r,s}.$$  

Therefore, we have shown that $\lim_{m \to \infty} \tilde{B}(x)_{r,s} = \mathcal{B}_{r,s}$ when $h = \omega(m^{-1})$. We will require an explicit upper bound on the rate of this convergence in the sequel. To this end, let us first compute the Lipschitz constant of $\gamma$ with respect to the $\ell^\infty$-norm. For any $i \in [d]$ and any fixed $(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_d) \in [-1, 1]^{d-1}$, notice that for every $u, \tilde{u} \in [-1, 1]$,  

$$|\gamma(u) - \gamma(u_{i-1}, \tilde{u}, u_{i+1}, \ldots, u_d)| = |u_i^{r_{i+1}} u_{i-1}^{r_{i-1}+s_i-1} \cdots u_d^{r_d+s_{d-1}} |u_i^{r_i+s_i} - \tilde{u}_i^{r_i+s_i}| 
\leq (r_i + s_i)|u_i - \tilde{u}_i|.$$  

Hence, via repeated application of the triangle inequality, we have

$$\forall u, \tilde{u} \in [-1, 1]^d, \quad |\gamma(u) - \gamma(\tilde{u})| = |u^{r+s} - \tilde{u}^{r+s}| \leq \sum_{i=1}^d (r_i + s_i)|u_i - \tilde{u}_i| 
\leq (|r| + |s|) \|u - \tilde{u}\|_{\ell^\infty}. \quad (53)$$

Now observe that for all $r, s \in \mathbb{Z}^d_+$ with $|r|, |s| \leq l$,

$$\left| \tilde{B}(x)_{r,s} - \mathcal{B}_{r,s} \right| = \frac{1}{r!s!} \left| \sum_{z \in [-1, 1]^d \cap \mathbb{G}_m} \frac{z^{r+s}}{(mh)^d} - \int_{I_z} \frac{u^{r+s}}{r!s!} \, du \right| 
\leq \frac{1}{r!s!} \left| \sum_{z \in [-1, 1]^d \cap \mathbb{G}_m} \frac{z^{r+s}}{(mh)^d} - \int_{I_z} \frac{u^{r+s}}{r!s!} \, du \right| 
\leq \frac{1}{r!s!} \sum_{z \in [-1, 1]^d \cap \mathbb{G}_m} \left( \sup_{u \in I_z} u^{r+s} - \inf_{\tilde{u} \in I_z} \tilde{u}^{r+s} \right) 
\leq \frac{|r| + |s|}{mh!} \sum_{z \in [-1, 1]^d \cap \mathbb{G}_m} \frac{1}{(mh)^d} 
\leq \frac{|r| + |s|}{mh!} \left( 2 + \frac{1}{mh} \right)^d 
\leq \frac{3^d(|r| + |s|)}{mh!}.$$  

where the second inequality follows from the triangle inequality, the fourth inequality follows from (53), the fifth inequality holds because $|z \in [-1, 1]^d \cap \mathbb{G}_m| \leq (2mh + 1)^d$ (since the interval $[-1, 1]$ can have at most $2mh + 1$ points at distance $(mh)^{-1}$ from each other), and the sixth inequality holds because we assume that $m \in \mathbb{N}$ and $h \in (0, \frac{1}{2})$ are such that $mh \geq 4l(3e)^d/\Lambda(d, l) \geq 1$. This implies that for all such $m$ and $h$,

$$\left\| \tilde{B}(x) - \mathcal{B} \right\|_F \leq \sqrt{\sum_{r,s \in \mathbb{Z}^d_+ : |r|, |s| \leq l} \frac{9^d(|r| + |s|)^2}{m^2h^2(r!s!)^2}}.$$
\[
\begin{align*}
    \sum_{r \in \mathbb{Z}^d_+ : |r| \leq l} \frac{1}{(r!)^2} & \leq 2l^3 \frac{1}{mh} \sum_{j=0}^{l} \frac{j!}{j!} \\
    & \leq 2l^3 \frac{1}{mh} \sum_{j=0}^{l} \frac{j!}{j!} \\
    & = 2l^3 \frac{1}{mh} \sum_{j=0}^{l} d^j \\
    & \leq 2l(3e)^d \frac{1}{mh}
\end{align*}
\] (54)

where the fourth equality follows from the multinomial theorem, and the fifth inequality uses
the Maclaurin series of the exponential function. Next, recall the following consequence of the
Wielandt-Hoffman-Mirsky inequality (cf. Horn and Johnson 2013, Corollary 6.3.8):

\[
||\lambda_{\min}(\tilde{B}(x)) - \lambda_{\min}(B)|| \leq \|\tilde{B}(x) - B\|_F.
\] (55)

Then, combining (54) and (55) produces the lower bound

\[
\lambda_{\min}(\tilde{B}(x)) \geq \lambda_{\min}(B) - \frac{2l(3e)^d}{mh},
\]

and employing Lemma 1 to this bound yields

\[
\lambda_{\min}(\tilde{B}(x)) \geq \Lambda(d,l) - \frac{2l(3e)^d}{mh},
\] (56)

where \(\Lambda(d,l)\) is defined in (14).

Finally, since \(m\) and \(h\) satisfy \(mh \geq 4l(3e)^d/\Lambda(d,l)\), or equivalently,

\[
\frac{2l(3e)^d}{mh} \leq \frac{\Lambda(d,l)}{2},
\]

using (52) and (56), we get

\[
\lambda_{\min}(B(x)) \geq b^d \lambda_{\min}(\tilde{B}(x)) \geq \frac{b^d \Lambda(d,l)}{2}
\]

for all \(x \in [h, 1-h]^d\). Since \(B(x) \succeq \lambda_{\min}(B(x))I_D\), the above lower bound on \(\lambda_{\min}(B(x))\) completes
the proof. \(\square\)

### 3.2. Properties of Local Polynomial Interpolation Weights

To prove Theorem 1, we will require two useful lemmata regarding the interpolation weights in
(17). The first lemma generalizes (Tsybakov 2009, Proposition 1.12) to the \(d\)-variate setting, and
demonstrates the intuitively pleasing property that polynomials are reproduced exactly by the
local polynomial interpolator given in (18).

**Lemma 2 (Polynomial Reproduction (Tsybakov 2009)).** Consider any \(d\)-variate polynomial \(g : [0,1]^d \rightarrow \mathbb{R}\) with total degree at most \(l\) (i.e., for every monomial of \(g\) with non-zero coefficient, the sum of the exponents of the \(d\) variables is at most \(l\)). Then, for every \(m \in \mathbb{N}\) and every \(x \in [h, 1-h]^d\) such that \(B(x)\) is positive definite, we have

\[
\hat{\phi}(x) = \sum_{y \in \mathbb{Z}^d_+} g(y)w_m^*(x) = g(x),
\]

where the interpolation weights are given in (17).
where the second equality follows from (13), and the third equality holds because $B$ is positive definite. Therefore, by considering the first coordinate, we get

$$mh^{\ell} = \arg\min_{s \in \mathbb{Z}^d_+} |s| \leq l,$$

Then, for every $y \in G_m$, we have

$$g(y) = \sum_{s \in \mathbb{Z}^d_+: |s| \leq l} \frac{\nabla^s g(x)}{s!} (y - x)^s$$

$$= q(x)^T U \left( \frac{y - x}{h} \right),$$

where the first equality uses Taylor’s theorem (cf. Apostol 1974, Theorem 12.14) and the fact that $g$ is a polynomial with total degree at most $l$, and $U$ is defined in (11). This implies that the solution to the weighted regression problem in (12) satisfies

$$\hat{\Phi}(x) = \arg\min_{\phi \in \mathbb{R}^D} \sum_{y \in G_m} (q(x) - \phi)^T U \left( \frac{y - x}{h} \right) U \left( \frac{y - x}{h} \right)^T (q(x) - \phi) \prod_{j=1}^d K \left( \frac{y_j - x_j}{h} \right)$$

$$= \arg\min_{\phi \in \mathbb{R}^D} (q(x) - \phi)^T B(x)(q(x) - \phi) = q(x)$$

where the second equality follows from (13), and the third equality holds because $B(x)$ is positive definite. Therefore, by considering the first coordinate, we get

$$\hat{\phi}(x) = \sum_{y \in G_m} g(y) w^*_y(x) = g(x),$$

where the first equality uses (18). This completes the proof. □

The second lemma generalizes (Tsybakov 2009, Lemma 1.3) to the $d$-variate setting, and portrays an $\ell^1$-norm bound on the sequence of interpolation weights in (17).

**LEMMA 3 ($\ell^1$-Norm Bound).** Suppose that $l \geq d \geq 19$. Then, for all $m \in \mathbb{N}$ and $h \in (0, \frac{1}{2})$ such that $mh \geq 4l(3e)^d/\Lambda(d, l)$ and all $x \in [h, 1-h]^d$, we have

$$\sum_{y \in G_m} |w^*_y(x)| \leq \frac{2}{\Lambda(d, l)} \left( \frac{3\sqrt{ec}}{b} \right)^d,$$

where $c \geq b > 0$ are the bounds on our kernel, and $\Lambda(d, l)$ is defined in (14).

**Proof.** Once again, we follow the proof of (Tsybakov 2009, Lemma 1.3) mutatis mutandis. First, notice that $B(x)$ satisfies the Löwner lower bound in Proposition 1 since the conditions of the proposition are satisfied. Then, observe that for every $x \in [h, 1-h]^d$, we have

$$\sum_{y \in G_m} |w^*_y(x)| \leq \sum_{y \in G_m} \|w_y(x)\|_2$$

$$= \frac{1}{(mh)^d} \sum_{y \in G_m} \left\| B(x)^{-1} U \left( \frac{y - x}{h} \right) \prod_{j=1}^d K \left( \frac{y_j - x_j}{h} \right) \right\|_2 1 \{\forall j \in [d], |y_j - x_j| \leq h\}$$

$$\leq \frac{c^d}{(mh)^d} \sum_{y \in G_m} \left\| B(x)^{-1} U \left( \frac{y - x}{h} \right) \prod_{j=1}^d K \left( \frac{y_j - x_j}{h} \right) \right\|_2 1 \{\forall j \in [d], |y_j - x_j| \leq h\}.$$
$$\leq \frac{2e^d}{(mh)b^d} \sum_{y \in \mathcal{Y}_m} \left\| U\left( \frac{y - x}{h} \right) \right\|_2 1\{ \forall j \in [d], |y_j - x_j| \leq h \}$$
$$\leq \frac{2}{\Lambda(d,l)} \left( \frac{\sqrt{e} c}{mhb} \right)^d \sum_{y \in \mathcal{Y}_m} 1\{ \forall j \in [d], |y_j - x_j| \leq h \}$$
$$\leq \frac{2}{\Lambda(d,l)} \left( \frac{\sqrt{e} c}{b} \right)^d \left( \frac{2mh + 1}{mh} \right)^d$$
$$\leq \frac{2}{\Lambda(d,l)} \left( \frac{3\sqrt{e} c}{b} \right)^d$$

where the first inequality follows from (17) and the Cauchy-Schwarz inequality, the second equality follows from (16), the third inequality holds because the kernel $K$ is upper bounded by $c > 0$ on the support $[-1,1]$, the fourth inequality follows from Proposition 1, the fifth inequality follows from the bound

$$\left\| U\left( \frac{y - x}{h} \right) \right\|_2^2 = \sum_{s \in \mathbb{Z}_+^d : |s| \leq l} \left( \frac{\Gamma(s)}{s!} \right)^2$$
$$\leq \sum_{s \in \mathbb{Z}_+^d : |s| \leq l} \frac{1}{(s!)^2}$$
$$\leq \sum_{i=0}^{l} \frac{1}{i!} \sum_{s \in \mathbb{Z}_+^d : |s| = i} \frac{i!}{s!}$$
$$= \sum_{i=0}^{l} \frac{d^i}{i!}$$
$$\leq e^d,$$

which uses (11), the fact that $|y_j - x_j| \leq h$ for all $j \in [d]$, the multinomial theorem, and the Maclaurin series of the exponential function, the sixth inequality holds because the interval $[-h,h]$ can have at most $2mh + 1$ points at distance $m^{-1}$ from each other, and the seventh inequality follows from the assumption that $mh \geq 4l(3e)^d/\Lambda(d,l) \geq 1$. This completes the proof. \[\square\]

### 3.3. Proof of Theorem 1

Finally, we are in a position to prove the supremum norm interpolation guarantee in Theorem 1.

**Proof of Theorem 1.** Suppose that $m \in \mathbb{N}$ and $h \in (0, \frac{1}{2})$ are such that $mh \geq 4l(3e)^d/\Lambda(d,l)$. Then, observe that for every $x \in [h,1-h]^d$, we have

$$\hat{\phi}(x) - g(x) = \sum_{y \in \mathcal{Y}_m} g(y)w^*_y(x) - g(x)$$
$$= \sum_{y \in \mathcal{Y}_m} w^*_y(x)(g(y) - g(x))$$
$$= \sum_{y \in \mathcal{Y}_m} w^*_y(x) \sum_{s \in \mathbb{Z}_+^d : 1 \leq |s| \leq l} \frac{\nabla^s g(x)}{s!} (y - x)^s$$
$$+ \sum_{s \in \mathbb{Z}_+^d : |s| = l} \frac{\nabla^s g(x + \tau(y - x)) - \nabla^s g(x)}{s!} (y - x)^s$$
\[
\sum_{y \in \mathcal{Y}_m} w_y^*(x) \sum_{s \in \mathbb{Z}_+^d : |s| = l} \frac{\nabla^s g(x + \tau(y - x)) - \nabla^s g(x)(y - x)^s}{s!}
\]

where the first equality follows from (18), the second equality holds because

\[
\sum_{y \in \mathcal{Y}_m} w_y^*(x) = 1
\]

which follows from Lemma 2 applied to the constant unit polynomial, the third equality follows from Taylor’s theorem with Lagrange remainder term (Apostol 1974, Theorem 12.14):

\[
g(y) - g(x) = \sum_{s \in \mathbb{Z}_+^d : 1 \leq |s| \leq l} \frac{\nabla^s g(x)}{s!}(y - x)^s + \sum_{s \in \mathbb{Z}_+^d : |s| = l} \frac{\nabla^s g(x + \tau(y - x)) - \nabla^s g(x)(y - x)^s}{s!}
\]

which holds for some \( \tau \in (0, 1) \), and the fourth equality holds because

\[
\sum_{y \in \mathcal{Y}_m} (y - x)^s w_y^*(x) = 0
\]

for all \( s \in \mathbb{Z}_+^d \) with \( 1 \leq |s| \leq l \), which follows from Lemma 2 applied to the polynomials \( u \mapsto (u - x)^s \).

This implies that for every \( x \in [h, 1 - h]^d \),

\[
|\hat{\phi}(x) - g(x)| \leq \sum_{y \in \mathcal{Y}_m} |w_y^*(x)| \sum_{s \in \mathbb{Z}_+^d : |s| = l} \frac{|\nabla^s g(x + \tau(y - x)) - \nabla^s g(x)|}{s!} |(y - x)^s|
\]

\[
\leq L_2 \sum_{y \in \mathcal{Y}_m} |w_y^*(x)| \|y - x\|_1^{\eta - l} \sum_{s \in \mathbb{Z}_+^d : |s| = l} \frac{1}{s!} |(y - x)^s|
\]

\[
= \frac{L_2}{l!} \sum_{y \in \mathcal{Y}_m} |w_y^*(x)| \|y - x\|_1^\eta \mathbb{1}\{\forall j \in [d], |y_j - x_j| \leq h\}
\]

\[
\leq 2L_2 (dh)^\eta \left( \frac{3\sqrt{e}c}{b} \right)^d \frac{4dl(3e)^d}{m\Lambda(d,l)}
\]

where the first inequality follows from the triangle inequality, the second inequality follows from the Hölder class assumption (see (6)) and the fact that \( \tau \in (0, 1) \), the third equality follows from the multinomial theorem, the fourth equality follows from (16) and (17) since the kernel \( K \) has support \([-1, 1]\), the fifth inequality follows from Lemma 3, and the sixth equality holds because we set

\[
h = \frac{4dl(3e)^d}{m\Lambda(d,l)},
\]

which minimizes the upper bound in the fifth inequality and achieves equality in the condition imposed on \( m \) and \( h \) at the outset of the proof. Lastly, to ensure that \( \sup_{x \in [h, 1 - h]^d} |\hat{\phi}(x) - g(x)| \leq \delta \), it suffices to impose the condition

\[
\frac{2L_2}{l!\Lambda(d,l)} \left( \frac{3\sqrt{e}c}{b} \right)^d \left( \frac{4dl(3e)^d}{m\Lambda(d,l)} \right)^\eta \leq \delta,
\]
which is equivalent to
\[ m \geq \frac{4dl(3e)^d}{\Lambda(d,l)} \left( \frac{2L_2}{d!\Lambda(d,l)} \right)^{1/\eta} \left( \frac{3\sqrt{e}c}{b} \right)^{d/\eta} \left( \frac{1}{\delta} \right)^{1/\eta}, \]
which in turn is implied by the condition
\[ m \geq \frac{12\sqrt{e}(2L_2 + 1)c}{b} \left( \frac{d!l(3e)^d}{(d!\Lambda(d,l))^2} \right)^{1/\eta}, \tag{58} \]
where we use the facts that \( \eta > l \geq d \geq 19 \) (by assumption) and \( \Lambda(d,l)^{(\eta+1)/\eta} \geq \Lambda(d,l)^2 \) (since \( \Lambda(d,l) \geq 1 \)), and we slacken the \( \Lambda(d,l) \) term because this does not change its double exponential nature. Moreover, we may further simplify (58) by noting that
\[ l^{1/\eta} \geq \frac{(2\pi)^{1/(2\eta)}l!^{(1/\eta)+(1/(2\eta))}}{e^{l/\pi}} \geq \frac{l}{e^{l/\pi}} \geq \frac{l}{2e} \]
where we use the Stirling’s formula bound (cf. Feller 1968, Chapter II, Section 9, Equation (9.15)) and the fact that \( l^{1/(l+1)} \leq 2 \) for \( l \geq 19 \). Applying this bound, we obtain the sufficient condition for (58) presented in the theorem statement:
\[ m \geq \frac{110(2L_2 + 1)c}{b} \left( \frac{d!l(3e)^d}{\Lambda(d,l)^2} \right)^{1/\eta}, \]
where we use the fact that \( 24e\sqrt{e} \leq 110 \). Finally, plugging this bound into (57) yields the upper bound on \( h \) in the theorem statement. This completes the proof. \( \square \)

We note that this argument is inspired by the proof of (Tsybakov 2009, Proposition 1.13), but the details are quite different.

4. Proofs of Convergence Analysis
In this section, we prove Propositions 2 and 3 and Theorem 2 from Section 2.2.2.

4.1. Proof of Proposition 2
To establish Proposition 2, we require the following lemma from the literature (Friedlander and Schmidt 2012, Lemma 2.1) (also see So and Zhou 2017, Theorem 2, Equation (19)) that upper bounds the difference between the ERM objective function (3) evaluated at \( \theta(T) \), which is the output of our proposed algorithm, and the true infimum in (4). Although (Friedlander and Schmidt 2012, Lemma 2.1) holds for general inexact gradient descent methods, we unwind the recursion in (Friedlander and Schmidt 2012, Lemma 2.1) and adapt it to our setting below for convenience.

**Lemma 4 (Inexact Gradient Descent Bound (Friedlander and Schmidt 2012)).**
Suppose the ERM objective function \( F : \mathbb{R}^p \to \mathbb{R} \) is continuously differentiable and \( \mu \)-strongly convex, and its gradient \( \nabla \theta F : \mathbb{R}^p \to \mathbb{R}^p \) is \( L_1 \)-Lipschitz continuous. Then, for any \( T \in \mathbb{N} \), we have
\[ 0 \leq F(\theta(T)) - F_* \leq \left( 1 - \frac{1}{\sigma} \right)^T \left( F(\theta(0)) - F_* \right) + \frac{1}{2L_1} \sum_{t=1}^{T} \left( 1 - \frac{1}{\sigma} \right)^{T-t} \left\| \nabla \theta F(t) - \nabla \theta F(\theta(t-1)) \right\|_2^2, \]
where \( \theta(t) \) for \( t \in \mathbb{Z}_+ \) are the inexact gradient descent updates given by (25) with an arbitrary initialization \( \theta(0) \), \( \nabla \theta F(t) \) for \( t \in \mathbb{N} \) are our local polynomial interpolation based approximations of the true gradients \( \nabla \theta F(\theta(t-1)) \) shown in (24), and \( \sigma = L_1/\mu \geq 1 \) is the condition number defined in Section 1.3.
We remark that (Friedlander and Schmidt 2012, Lemma 2.1) additionally assumes that the infimum \( F^* \) in (4) can be achieved. However, this condition need not be imposed in Lemma 4, since if \( F \) is strongly convex on \( \mathbb{R}^m \), its infimum \( F^* \) is finite and achieved by some unique global minimizer \( \theta_* \in \mathbb{R}^p \). We next prove Proposition 2 using Lemma 4 and Theorem 1.

**Proof of Proposition 2.** We first verify that the conditions of Lemma 4 hold. Recall the ERM objective function \( F : \mathbb{R}^p \to \mathbb{R} \) given in (3):

\[
\forall \theta \in \mathbb{R}^p, \quad F(\theta) = \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)}; \theta),
\]

where \( \{x^{(i)} \in [h', 1-h']^d : i \in [n] \} \) is some fixed training data, and \( f : [0, 1]^d \times \mathbb{R}^p \to \mathbb{R} \) is a given loss function satisfying the assumptions in Section 1.3. Specifically, since \( f(x^{(i)}; \cdot) : \mathbb{R}^p \to \mathbb{R} \) is \( \mu \)-strongly convex for all \( i \in [n] \), \( F \) is also \( \mu \)-strongly convex due to (Nesterov 2004, Lemma 2.1.4). Likewise, since \( \nabla f(x^{(i)}; \cdot) : \mathbb{R}^p \to \mathbb{R}^p \) is \( L_1 \)-Lipschitz continuous for all \( i \in [n] \), \( F \) is continuously differentiable and \( \nabla \theta F : \mathbb{R}^p \to \mathbb{R}^p \) is also \( L_1 \)-Lipschitz continuous. Indeed, observe that for all \( \theta_1, \theta_2 \in \mathbb{R}^p \),

\[
\|\nabla \theta F(\theta_1) - \nabla \theta F(\theta_2)\|_2 \leq \frac{1}{n} \sum_{i=1}^{n} \|\nabla \theta f(x^{(i)}; \theta_1) - \nabla \theta f(x^{(i)}; \theta_2)\|_2 \\
\leq \frac{L_1}{n} \sum_{i=1}^{n} \|\theta_1 - \theta_2\|_2 \\
= L_1 \|\theta_1 - \theta_2\|_2,
\]

where we use the triangle inequality.

Next, we apply Lemma 4 and get

\[
F(\theta^{(T)}) - F_* \leq \left(1 - \frac{1}{\sigma}\right)^T (F(\theta^{(0)}) - F_*) + \frac{1}{2L_1} \sum_{t=1}^{T} \left(1 - \frac{1}{\sigma}\right)^{T-t} \left\|\widehat{\nabla F}^{(t)} - \nabla \theta F(\theta^{(t-1)})\right\|_2^2,
\]

where \( \theta^{(t)} \) for \( t \in \mathbb{Z}_+ \) are our updates in (25), and \( \widehat{\nabla F}^{(t)} \) for \( t \in \mathbb{N} \) are our approximations of the true gradients in (24). To upper bound the squared \( \ell^2 \)-norms on the right hand side of this inequality, notice that for any \( t \in \mathbb{N} \), we have

\[
\left\|\widehat{\nabla F}^{(t)} - \nabla \theta F(\theta^{(t-1)})\right\|_2 \leq \frac{1}{n} \sum_{j=1}^{n} \sqrt{\sum_{i=1}^{p} \left(\frac{\partial f}{\partial \theta_i}(x^{(j)}; \theta^{(t-1)})\right)^2} \\
\leq \sqrt{p} \delta_t,
\]

where the first inequality follows from (24) and the triangle inequality, \( \hat{\phi}_i^{(t)}(x) \) denotes the output of our local polynomial interpolator for \( \frac{\partial f}{\partial \theta_i}(\cdot; \theta^{(t-1)}) \) when evaluated at \( x \) (cf. (23)), and the third inequality follows from Theorem 1 and our choices of grid sizes (20) and bandwidths (21) in Section 2.2.1 (where (19) ensures that \( h_t \leq h' \)). Hence, we obtain

\[
F(\theta^{(T)}) - F_* \leq \left(1 - \frac{1}{\sigma}\right)^T (F(\theta^{(0)}) - F_*) + \frac{p}{2L_1} \sum_{t=1}^{T} \left(1 - \frac{1}{\sigma}\right)^{T-t} \delta_t^2.
\]
We now select appropriate values for the constants \( \{ \delta_t \in (0, 1) : t \in \mathbb{N} \} \). For every \( t \in \mathbb{N} \), choose
\[
\delta_t = \left( 1 - \frac{1}{\sigma} \right)^T
\]
as indicated in (27). This produces
\[
F(\theta(T)) - F_* \leq \left( 1 - \frac{1}{\sigma} \right)^T \left( F(\theta(0)) - F_* + \frac{p}{2L_1} \sum_{t=1}^{T} \left( 1 - \frac{1}{\sigma} \right)^{T-t} \right)
\leq \left( 1 - \frac{1}{\sigma} \right)^T \left( F(\theta(0)) - F_* + \frac{p}{2L_1} \sum_{t=0}^{\infty} \left( 1 - \frac{1}{\sigma} \right)^t \right)
\leq \left( 1 - \frac{1}{\sigma} \right)^T \left( F(\theta(0)) - F_* + \frac{p\sigma}{2L_1} \right),
\]
where the last equality computes the value of the previous geometric series.

Finally, to ensure that \( \theta(T) \) is an \( \epsilon \)-approximate solution of (4), viz.
\[
F(\theta(T)) - F_* \leq \epsilon
\]
as defined in (8), it suffices to impose the condition
\[
\left( 1 - \frac{1}{\sigma} \right)^T \left( F(\theta(0)) - F_* + \frac{p\sigma}{2L_1} \right) \leq \epsilon.
\]
By taking logarithms of both sides, and rearranging and simplifying the resulting inequality, we obtain the equivalent inequality
\[
T \geq \left( \log \left( \frac{\sigma}{\sigma - 1} \right) \right)^{-1} \log \left( \frac{F(\theta(0)) - F_* + \frac{p}{2\mu \eta}}{\epsilon} \right),
\]
where we also use the fact that \( \sigma = L_1/\mu \). Therefore, after running the LPI-GD algorithm for \( T \in \mathbb{N} \) iterations with \( T \) given by (26), the updated parameter vector \( \theta(T) \) forms an \( \epsilon \)-approximate solution to (4). We also note that for every \( t \in \mathbb{N} \),
\[
\delta_t = \left( 1 - \frac{1}{\sigma} \right)^T = \Theta \left( \sqrt{\frac{\epsilon}{p}} \right),
\]
by substituting (26) into our choice of \( \delta_t \) above. This yields (27). Hence, the proof is complete. 

4.2. Proof of Theorem 2

We next establish the oracle complexity bound in Theorem 2 using Proposition 2.

Proof of Theorem 2. Since we choose the maximum allowable approximation errors \( \{ \delta_t \in (0, 1) : t \in [T] \} \) to be constant with respect to \( t \) as in (27), a single common uniform grid \( G_m \) is used in all iterations of the LPI-GD algorithm, where \( m = m_t \) is given by (20) (and is also constant with respect to \( t \)). The size of this grid is upper bounded by
\[
|G_m| = m^d
\leq \left( \frac{110(2L_2 + 1)c}{b} \left( \frac{d(3\epsilon)^d}{\Lambda(d,t)^2} \right) \left( \frac{\sigma}{\sigma - 1} \right)^{T/(2\eta)} \right)^d
\]
Lastly, we derive Proposition 3 from Theorem 2.

### 4.3. Proof of Proposition 3

We first establish the scaling of $C_{\mu,L_1,L_2,b,c}(d,l)$ in (28). Since $d = O(\log \log(n))$, observe that

$$\left(\frac{\sigma + 2(L_1 - \mu)}{(L_1 - \mu) \log \left(\frac{\sigma}{\sigma - 1}\right)}\right) \left(\frac{220(2L_2 + 1)c}{b}\right)^d = O(\text{polylog}(n)),$$

and since $l \geq d$, observe that

$$d^d(3e)^d \frac{\sigma - 1}{\Lambda(d,l)^{2d}} = O\left((l + d)^{6d}(\frac{\sigma(1 - d)}{d})^d\right)$$

as $d$ grows, where we use (14). To simplify the second scaling expression above, notice that for all sufficiently large $d \in \mathbb{N}$, we have

$$\log \log \left(\frac{d^d(3e)^d}{\Lambda(d,l)^{2d}}\right) \leq \log \log(l + d) + \log(ld) + d \log \left(\frac{\epsilon(l + d)}{d}\right) + \Theta(1)$$
\[
= \Theta(\log(d)) + d \log \left( \frac{e(l + d)}{d} \right)
\]
\[
\leq \Theta(\log(d)) + d \log(e(\gamma + 1))
\]
\[
\leq 2 \log(e(\gamma + 1))d
\]

where the second equality follows from the fact that \( \eta = \Theta(d) \), and the third inequality holds because \( l \leq \gamma d \). This implies that for all sufficiently large \( d \in \mathbb{N} \),
\[
\frac{d^4(3e)^2}{\lambda(d, l)^{2d}} \leq \exp(e^{2\log(e(\gamma + 1))}d).
\]

(62)

Hence, since \( d \leq \frac{\log \log(n)}{\log(e(\gamma + 1))} \), we obtain
\[
C_{\mu, L_1, L_2, b, c}(d, l) \leq O(\text{polylog}(n)) \cdot \exp\left(\sqrt{\log(n)}\right)
\]
\[
\leq \exp\left(2\sqrt{\log(n)}\right)
\]

for all sufficiently large \( n \in \mathbb{N} \), using (28), (61), and (62). Next, applying (30), which follows from Theorem 2, we have that for all sufficiently large \( n \in \mathbb{N} \),
\[
\Gamma(LPI-GD) \leq c_1 \exp\left(2\sqrt{\log(n)}\right) \left(\frac{p}{\epsilon}\right)^{d/(2n)} \log\left(\frac{p}{\epsilon}\right)
\]
\[
\leq c_1 \exp\left(2\sqrt{\log(n)}\right) \left(c_2 n^{\alpha + \beta}\right)^{d/(2n)} \log\left(c_2 n^{\alpha + \beta}\right)
\]
\[
\leq c_3 \log(n) \exp\left(2\sqrt{\log(n)}\right) n^{1/\tau}
\]

where \( c_1, c_2, c_3 > 0 \) are some positive constants, the second inequality follows from the assumptions that \( \epsilon = \Theta(n^{-\alpha}) \) and \( p = O(n^\beta) \), and the third inequality holds because \( \eta \geq \tau(\alpha + \beta)d/2 \). This proves the scaling of \( \Gamma(LPI-GD) \) with respect to \( n \).

Finally, from the relations (31) and (32) and the assumptions that \( \epsilon = \Theta(n^{-\alpha}) \) and \( p = O(n^\beta) \), we get \( \Gamma_*(GD) = \Theta(n \log(n)) \) and \( \Gamma_*(SGD) = \Theta(n^{\alpha + \beta}) \). Therefore, for some constants \( c_4, c_5 > 0 \), we obtain
\[
\lim_{n \to \infty} \frac{\Gamma(LPI-GD)}{\Gamma_*(GD)} \leq \lim_{n \to \infty} \frac{c_3 \exp\left(2\sqrt{\log(n)}\right) n^{1/\tau}}{c_4 n} = 0,
\]
\[
\lim_{n \to \infty} \frac{\Gamma(LPI-GD)}{\Gamma_*(SGD)} \leq \lim_{n \to \infty} \frac{c_3 \log(n) \exp\left(2\sqrt{\log(n)}\right) n^{1/\tau}}{c_5 n^{\alpha + \beta}} = 0,
\]

which use the fact that \( \tau > \max\{1, (\alpha + \beta)^{-1}\} \). This completes the proof. \( \square \)

5. Conclusion

We conclude by summarizing our main contributions and reiterating some directions of future research. In order to exploit the smoothness of loss functions in data for optimization purposes, we proposed the LPI-GD algorithm (see Algorithm 1) to compute approximate solutions of the ERM problem in (3) and (4), where the loss function is strongly convex and smooth in both the parameter and the data. We then derived the iteration and oracle complexities of this algorithm, and illustrated that its oracle complexity beats the oracle complexities of both GD and SGD for a very broad range of dependencies between the parameter dimension \( p \), the approximation accuracy \( \epsilon \), and the number of training samples \( n \), when the data dimension \( d \) is sufficiently small.
Finally, in the course of our convergence analysis, we also provided a careful and detailed analysis of multivariate local polynomial interpolation with supremum norm guarantees, which may be of independent interest in non-parametric statistics.

We close our discussion by stating three avenues for future work. First, as stated at the end of Section 2 and in Section 3.1, it would be very valuable to improve the dependence of Theorem 2 on the data dimension $d$ by sharpening the minimum eigenvalue lower bound in the key technical estimate in Lemma 1 (see Conjecture 1). Second, the explicit dependence of Theorem 2 on the parameter dimension $p$ could be improved by establishing new interpolation guarantees for approximating smooth $\mathbb{R}^p$-valued functions using local polynomial regression or other methods. Last, as noted in Section 1, while our work focuses on the strongly convex loss function setting, similar analyses could be carried out for other conventional scenarios, e.g., when $f(x; \cdot): \mathbb{R}^p \rightarrow \mathbb{R}$ is non-convex and $\epsilon$-approximate solutions to (4) are defined via approximate first order stationary points.

Appendix A: Complexities of Gradient Descent and Stochastic Gradient Descent

In this appendix, we present the well-known first order oracle complexities of GD and SGD from the literature. Recall the setup in Section 1.3. To illustrate the scaling of these complexities, we assume that $n, d, p, \epsilon^{-1} \rightarrow \infty$ and all other problem parameters are constant. As shown in the proof of Proposition 2 in Section 4.1, the strong convexity and Lipschitz continuous gradient assumptions on the loss function in Section 1.3 imply that the ERM objective function $F: \mathbb{R}^p \rightarrow \mathbb{R}$ in (3) is $\mu$-strongly convex and has $L_1$-Lipschitz continuous gradient. Let $\theta_0 \in \mathbb{R}^p$ be the unique global minimizer of the empirical risk $F: \mathbb{R}^p \rightarrow \mathbb{R}$ so that $F_* = F(\theta_0)$ and $\nabla \theta F(\theta_0) = 0$ (which is the zero vector). Furthermore, assume that the parameter $\theta$ belongs to an $\ell^\infty$-ball with constant radius, e.g., $[0, 1]^p$, and all iterates of GD, SGD, and LPI-GD also live in this $\ell^\infty$-ball. (This is very often the case in applications, because unbounded values cannot be easily represented in computers.) The $\ell^2$-diameter of such an $\ell^\infty$-ball scales as $O(\sqrt{p})$, which means that the $\ell^2$-distance between any two parameter vectors is $O(\sqrt{p})$. Under these assumptions, the next proposition presents the first order oracle complexity of GD with appropriately chosen constant step size (Nesterov 2004, Theorem 2.1.15).

**Proposition 4 (Oracle Complexity of GD (Nesterov 2004)).** Under the aforementioned assumptions, the first order oracle complexity of GD is bounded by

\[
\Gamma_{GD} \leq \frac{n}{2} \left( \log \left( \frac{\sigma + 1}{\sigma - 1} \right) \right)^{-1} \log \left( \frac{L_1 \| \theta^{(0)} - \theta_* \|^2_2}{2 \epsilon} \right) = O \left( n \log \left( \frac{p}{\epsilon} \right) \right),
\]

where $\theta^{(0)} \in \mathbb{R}^p$ is the initial parameter vector of GD and $\| \theta^{(0)} - \theta_* \|^2_2 = O(p)$.

Proposition 4 follows from the facts that: 1) Each iteration of GD makes $n$ first order oracle queries, and 2) It takes $O(\log(p\epsilon^{-1}))$ iterations for GD to produce an $\epsilon$-approximate solution in the sense of (8) (Nesterov 2004, Theorem 2.1.15). When certain universal second moment bounds are additionally satisfied on stochastic gradients, the next proposition conveys the first order oracle complexity of SGD with linearly diminishing step sizes (cf. Nemirovski et al. 2009 and Bottou et al. 2018, Theorem 4.7).

**Proposition 5 (Oracle Complexity of SGD (Bottou et al. 2018)).** As before, under the previous assumptions, the first order oracle complexity of SGD is bounded by

\[
\Gamma_{SGD} = O \left( \max \left\{ \| \theta^{(0)} - \theta_* \|^2_2, \sup_{x \in [0, 1]^d, \theta \in \mathbb{R}^p} \| \nabla \theta f(x; \theta) \|^2_2 \right\} \right) = O \left( \frac{p}{\epsilon} \right),
\]

where $\theta^{(0)} \in \mathbb{R}^p$ is the initial parameter vector of SGD, $\sup_{x \in [0, 1]^d, \theta \in \mathbb{R}^p} \| \nabla \theta f(x; \theta) \|^2_2 = O(p)$, and $\| \theta^{(0)} - \theta_* \|^2_2 = O(p)$.
Note that since the iterates of SGD are random variables, the notion of an $\epsilon$-approximate solution in (8) is defined in expectation for SGD. So, Proposition 5 presents the number of first order oracle queries required to obtain an $\epsilon$-approximate solution in expectation. Furthermore, we remark that the scaling of $\sup_{x \in [0,1]^d, \theta \in \mathbb{R}^p} \| \nabla f(x; \theta) \|^2_2$ is obtained as follows:

$$
\sup_{x \in [0,1]^d, \theta \in \mathbb{R}^p} \| \nabla f(x; \theta) \|^2_2 = \sup_{x \in [0,1]^d, \theta \in \mathbb{R}^p} \| \nabla f(x; \theta) - \nabla f(x; \theta_*) \|^2_2 \leq L_1^2 \sup_{x \in [0,1]^d, \theta \in \mathbb{R}^p} \| \theta - \theta_* \|^2_2 = O(p),
$$

where $\theta_* \in \mathbb{R}^p$ denotes the unique global minimizer of the strongly convex function $f(x; \cdot) : \mathbb{R}^p \to \mathbb{R}$ for any $x \in [0,1]^d$, $\nabla f(x; \theta_*) = 0$ (i.e., the zero vector) at the minimum point, the inequality follows from the smoothness in parameter assumption in Section 1.3, and $\sup_{x \in [0,1]^d, \theta \in \mathbb{R}^p} \| \theta - \theta_* \|^2_2 = O(p)$ because the parameter belongs to an $\ell^\infty$-ball with constant radius.

Finally, it is worth mentioning that standard comparisons between the oracle complexities of GD and SGD in the literature (as we described at the outset of Section 1) often assume that the factors $\| \theta(0) - \theta_* \|^2_2$ and $\sup_{x \in [0,1]^d, \theta \in \mathbb{R}^p} \| \nabla f(x; \theta) \|^2_2$ are constants with respect to the parameter dimension $p$. However, we perform our comparisons between oracle complexities by allowing these factors to scale as $O(p)$, which is reasonable when the parameter belongs to an $\ell^\infty$-ball with constant radius.

Acknowledgments

This research was supported in part by an ONR grant N000142012394, an ARO MURI grant W911NF-19-1-0217, and the NSF TRIPODS Foundations of Data Science.

References


