Gradient Descent with Provably Tuned Learning-rate Schedules

Dravyansh Sharma
Toyota Technological Institute at Chicago, Northwestern University

Abstract

Gradient-based iterative optimization methods are the workhorse of modern machine learning. They crucially rely on careful tuning of parameters like learning rate and momentum. However, one typically sets them using heuristic approaches without formal near-optimality guarantees. Recent work studies how to learn a good step-size in gradient descent. However, like most of the literature with theoretical guarantees for gradient-based optimization, their results rely on strong assumptions on the function class including convexity and smoothness which do not hold in typical applications. In this work, we develop novel analytical tools for provably tuning hyperparameters in gradient-based algorithms that apply to non-convex and non-smooth functions. We obtain matching sample complexity bounds for learning the step-size in gradient descent shown for smooth, convex functions in prior work (up to logarithmic factors) but for a much broader class of functions. Our analysis applies to gradient descent on neural networks with commonly used activation functions (including ReLU, sigmoid and tanh). We extend our framework to tuning multiple hyperparameters, including tuning the learning rate schedule, simultaneously tuning momentum and step-size, and pre-training the initialization vector. Our approach can be used to bound the sample complexity for minimizing both the validation loss as well as the number of gradient descent iterations.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 INTRODUCTION

Gradient descent is a foundational optimization algorithm widely employed in machine learning and deep learning to minimize loss functions and improve model performance. A critical hyperparameter in gradient descent is the step size or learning rate, which dictates how far the algorithm moves along the negative gradient direction in each iteration. Selecting an appropriate step size is essential: a value too large can cause divergence, while one too small can lead to slow convergence or the inability to escape undesirable local minima leading to poor quality of the final iterate.

Considerable research has focused on tuning the step size for individual tasks. However, many real-world applications involve multi-task learning or repeated optimization across a collection of tasks. In such settings, the optimal step size may vary significantly between different task domains, and naive strategies such as using a fixed or globally tuned step size often yield suboptimal performance. This raises an important question: how can we effectively tune the step size of gradient descent across multiple, related tasks with provable guarantees?

This paper explores principled approaches to tuning step size in gradient descent in multi-task environments. We investigate the theoretical underpinnings of step size sensitivity across tasks, by examining how the convergence varies as a function of the step size. We provide theoretical guarantees for the amount of data needed (sample complexity) for tuning the step size with provable guarantees, even in the presence of non-smooth and non-convex functions. Concretely,

- We study tuning of the step-size in gradient descent (Algorithm 1) across tasks in the framework of Gupta and Roughgarden [2016], but for a much broader class of functions. While prior work for sample complexity of step-size tuning assumes the class of optimized functions to be convex, smooth and satisfying a guaranteed progress assumption (roughly corresponds to strong convexity), our analysis works without any of these assumptions. We

Algorithm 1 Gradient descent(step size η)

Input: Initial point x, function to minimize f, maximum number of iterations H, gradient threshold for convergence θ

Initialize $x_1 \leftarrow x$
for $i = 1, \ldots, H$ **do**
  if $||\nabla f(x_i)|| < \theta$ **then**
    Return x_i
  $x_{i+1} = x_i - \eta \nabla f(x_i)$

Output: Return x_{H+1}

only assume that the class of functions is piecewise-polynomial, a property satisfied by neural networks with piecewise-polynomial activation functions (e.g., ReLU activation), or is piecewise-Pfaffian (which includes other commonly used activation functions like sigmoid and tanh).

- We show sample complexity bounds of $\tilde{O}(H^3/\epsilon^2)$ for uniform convergence for tuning the learning rate in the piecewise-polynomial case (asymptotically matching the previous bounds for convex and smooth functions, up to logarithmic terms), which implies generalization of the learned step-size to unseen instances (that is, small gap between training and test errors). H denotes the maximum number of iterations in Algorithm 1.
- We further extend our techniques to obtain a $\tilde{O}(H^4/\epsilon^2)$ bound on the sample complexity of tuning the entire learning rate schedule.
- We show how to use our framework to study tuning of the initialization scale parameter as well as the entire initialization vector. The latter corresponds to the sample complexity of pre-training using gradient descent.
- We show that our techniques extend beyond vanilla gradient descent. In particular, we show how to analyze simultaneous tuning of the stepsize and momentum parameters in momemtum-based gradient descent (applicable to widely used optimizers like Adam).
- Finally, we show that our analytical techniques can be used to optimize not only the speed of convergence (which corresponds to the efficiency of the training procedure using gradient-based optimization), but also to optimize the quality of the last iterate (e.g. in terms of the validation loss of the learned network weights).

Related work. Recently tuning the learning rate in gradient descent by meta learning has received significant research interest Maclaurin et al. [2015], Andrychowicz et al. [2016], Li and Malik [2017], Denevi et al. [2019], Khodak et al. [2019]. Despite empirical success, theoretical understanding of setting a good learning rate is largely limited to convex and strongly-convex functions. In this work, we develop a principled understanding beyond the convex case under the recently introduced *data-driven algorithm design* paradigm [Gupta and Roughgarden, 2016, Balcan, 2020] (see Appendix A for additional related work).

Data-driven tuning of deep networks. Recent work develops techniques for data-driven tuning of model hyperparameters in deep nets [Balcan et al., 2025b], but their techniques do not apply to parameters of training algorithms including tuning the learning rate which is the focus of this work. At the technical level, our analysis apply to multiple hyperparameters (both Gupta and Roughgarden [2016], Balcan et al. [2025b] apply only to single hyperparameters) and we do not need additional regularity assumptions (Balcan et al. [2025b] need the polynomials to be in a certain "general position") for our results. Finally, we study tuning of hyperparameters like learning-rate schedules and pre-trained initializations, which must be tuned very carefully in practice, with provably bounded sample complexity.

Gradient-descent analysis in non-convex regimes. The non-convexity of neural network optimization problems is well-known [Goodfellow et al., 2015], although the theoretical understanding is quite limited. A large body of research had been devoted to understanding conditions under which gradient descent avoids saddle points and sub-optimal local minima [Lee et al., 2017, Kawaguchi, 2016, Jin et al., 2017, Ge et al., 2017, Daneshmand et al., 2018]. Closer to the current work is the study of conditions under which gradient descent converges fast despite non-convexity [Karimi et al., 2016, Yue et al., 2023]. Although interesting for showing linear convergence of gradient descent in a non-convex setting, these conditions are far from capturing actual optimization in deep learning. Our analysis handles realistic deep network optimization settings, and gives guarantees for learning the learning rates and schedules with provably optimal convergence rates.

On tuning step-sizes and learning rate schedules. Several heuristics for setting the step-sizes and learning rate schedules are known and regularly employed in practice [Smith, 2015, Loshchilov and Hutter, 2017, Goyal et al., 2017]. This has motivated theoretical research into the impact of step-size and schedules on the convergence of gradient descent [Ghadimi and Lan, 2013, Reddi et al., 2018, Oymak, 2021, Kalra and Barkeshli, 2024]. While the impact is only understood in limited scenarios, it is observed more widely in practice. In the multi-task setting, several approaches are known including learning-to-optimize [Andrychowicz et al., 2016], meta-learning for SGD Li and Malik [2017], hypergradients [Baydin

et al., 2018], implicit differentiation [Maclaurin et al., 2015, Pedregosa, 2016, Lorraine et al., 2020], and population-based training [Jaderberg et al., 2017], to list a few. Another line of work shows the significance of adaptive stepsizes in gradient descent for logistic regression [Wu et al., 2024, Zhang et al., 2025].

Adaptive "parameter-free" approaches. A growing line of work seeks to eliminate or automatically tune learning rates in stochastic optimization. Early progress includes the algorithm of Carmon and Hinder [2022], which achieves near-optimal convergence for stochastic convex optimization without prior knowledge of smoothness, gradient norms, or noise levels. More recent approaches Ivgi et al. [2023] refine this idea by designing dynamic step-size rules based on observed gradients, achieving strong parameter-free guarantees and competitive empirical performance with tuned SGD. Similarly, Defazio and Mishchenko [2023] propose D-Adaptation, which automatically rescales updates to match optimal convergence rates for convex problems without manual tuning, while Defazio et al. [2024] achieve state-of-the-art performance without explicit schedules. Despite these advances, meta-learning or data-driven tuning of step sizes remains important for several reasons. Existing "parameter-free" methods still rely on specific structural assumptions (e.g., convexity, bounded gradients), may be conservative in practice, and typically optimize worst-case guarantees. In contrast, learned step-size schedules can exploit problem structure, distributional information, and task similarity to achieve faster convergence and better generalization in realistic settings, suggesting that adaptive theory-driven methods and learned schedules are complementary rather than competing approaches.

See Appendix A for additional related work.

2 NOTATION AND PRELIMINARIES

We recall the notation and setup for data-driven tuning of gradient step introduced by Gupta and Roughgarden [2016]. The instance space of problems Π here consists of pairs (x, f) of initial points $x \in \mathbb{R}^d$ and functions $f : \mathbb{R}^d \to \mathbb{R}$. The family of gradient descent algorithms $\mathcal{A}$ is given by $\mathcal{P} \subset \mathbb{R}_+$ consisting of valid values of the step size η. We recall the vanilla gradient descent algorithm (Algorithm 1). We define the cost function $\ell(\eta, x, f)$ for $\eta \in \mathcal{P}$ and $(x, f) \in \Pi$ as the number of iterations for which Algorithm 1 runs on the instance (x, f) when run with step size η. Let $\ell_\eta(x, f) := \ell(\eta, x, f)$ for all η, x, f. For guarantees on sample complexity of tuning η, we will be interested in the learning-theoretic complexity (pseudo-dimension) of the function class $\mathcal{L} = \{\ell_\eta \mid \eta \in \mathcal{P}\}$.

Notice that while prior work Gupta and Roughgarden [2016] assumes strong sufficient conditions—convexity, L-smoothness, and a guaranteed progress condition (that is $\|x_{i+1} - x^*\| \le (1-c)\|x_i - x^*\|$ for some $c > 0$ where x^* is the unique stationary point and x_i is the ith iterate in Algorithm 1, this roughly corresponds to strong-convexity)—for the convergence to always happen and includes appropriate restrictions on the step-size η. In particular, these results do not apply to deep neural networks. Our results hold when all of these conditions are violated. We handle non-convergence by assigning it the same cost as the maximum number of iterations, equal to H. Formally,

$$\ell(\eta, x, f) := \begin{cases} \operatorname*{argmin}_{i \in [H]} \|\nabla f(x_i)\| \le \theta, & \text{if such } i \text{ exists,} \\ H, & \text{otherwise.} \end{cases}$$

Note that x_i depends on η, x, f and i. We consider the step-size as the hyperparameter of interest in this work, and assume other parameters like maximum number of iterations H and gradient threshold for convergence θ are fixed and known. We will use the $\tilde{O}$ notation to suppress dependence on quantities apart from H and the generalization error ϵ for simplicity.

Learning theory background. The pseudo-dimension is frequently used to analyze the learning theoretic complexity of real-valued function classes, and will be a main tool in our sample complexity analysis. For completeness, we include below the formal definition.

Definition 1 (Shattering and Pseudo-dimension, Anthony and Bartlett [1999])**.** *Let $\mathcal{L}$ be a set of functions mapping from $\mathcal{X}$ to $\mathbb{R}$, and suppose that $\mathcal{S} = \{x_1, \ldots, x_m\} \subseteq \mathcal{X}$. Then $\mathcal{S}$ is pseudo-shattered by $\mathcal{L}$ if there are real numbers $r_1, \ldots, r_m$ such that for each $b \in \{0,1\}^m$ there is a function l_b in $\mathcal{L}$ with $\operatorname{sign}(l_b(x_i) - r_i) = b_i$ for $i \in [m]$. We say that $r = (r_1, \ldots, r_m)$ witnesses the shattering. We say that $\mathcal{L}$ has pseudo-dimension d if d is the maximum cardinality of a subset $\mathcal{S}$ of $\mathcal{X}$ that is pseudo-shattered by $\mathcal{L}$, denoted $\operatorname{Pdim}(\mathcal{L}) = d$. If no such maximum exists, we say that $\mathcal{L}$ has infinite pseudo-dimension.*

Pseudo-dimension is a generalization of VC-dimension to real-valued functions, and is a classic complexity notion in learning theory due to the following theorem which implies the uniform convergence for any function in class $\mathcal{L}$ when $\operatorname{Pdim}(\mathcal{L})$ is finite.

Theorem 2.1 ((ϵ, δ)-uniform convergence sample complexity via pseudo-dimension, Anthony and Bartlett [1999])**.** *Suppose $\mathcal{L}$ is a class of real-valued functions with range in $[0, H]$ and finite $\operatorname{Pdim}(\mathcal{L})$.*

For every $\epsilon > 0$ and $\delta \in (0,1)$, given any distribution $\mathcal{D}$ over $\mathcal{X}$, with probability $1-\delta$ over the draw of a sample $S \sim \mathcal{D}^M$, for all functions $l \in \mathcal{L}$, we have $|\frac{1}{M}\sum_{x\in S} l(x) - \mathbb{E}_{x\sim\mathcal{D}}[l(x)]| \leq \epsilon$ for some $M = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\mathrm{Pdim}(\mathcal{L}) + \log\frac{1}{\delta}\right)\right)$.

We also need the following lemma from data-driven algorithm design, which bounds the pseudo-dimension of the class of loss functions, when the dual losses (i.e. losses as a function of some algorithmic hyperparameter computed on any fixed problem instance) have a piecewise constant structure with a bounded number of pieces.

Lemma 2.2. *(Lemma 2.3, Balcan [2020]) Suppose that for every problem instance $x \in \mathcal{X}$, the function $u_x^*(\alpha) : \mathbb{R} \to \mathbb{R}$ is piecewise constant with at most N pieces. Then the family $\{u_\alpha(\cdot)\}$ over instances in $\mathcal{X}$ has pseudo-dimension $O(\log N)$.*

3 IMPROVED SAMPLE COMPLEXITY GUARANTEES FOR TUNING THE STEP SIZE

Our overall approach towards bounding the pseudo-dimension of the cost function class $\mathcal{L} = \{\ell_\eta : \Pi \to [0,H] \mid \eta \in \mathcal{P}\}$ (which implies a bound on the sample complexity of tuning η by classical learning theory), is by examining the structure of $\ell(\eta, x, f)$ on any fixed instance (x, f) as the step-size η is varied (also called the *dual* cost function $\ell_{x,f}(\eta)$ Balcan et al. [2021a]) is very different from prior work. The key idea of prior work [Gupta and Roughgarden, 2016, Jiao et al., 2025] is to establish a near-Lipschitzness property for the dual cost function, by bounding how far the number of steps to converge may diverge as the step-size is changed slightly, and then use a discretization argument over the space of step-sizes. It is easy to see that this approach cannot extend beyond very nicely-behaved functions (roughly, strongly-convex and L-smooth) as generally small changes to the step-size can cause dramatic changes to the number of the steps needed for convergence. In contrast, intuitively our analysis examines the *piecewise monotonicity* of the number of steps needed to converge as the step-size η is varied.

3.1 Gradient descent for piecewise polynomial functions

In this section, we will consider several interesting function classes which are non-convex and non-smooth but intuitively have a bounded amount of oscillations. Our new analytical approach allows us to significantly extend the class of convex and smooth functions (with near strong-convexity properties) studied by Gupta and Roughgarden [2016], for which they obtain a $\tilde{O}(H^3/\epsilon^2)$ bound on the sample complexity of tuning the step-size η in Algorithm 1. Remarkably, we will achieve the same asymptotic dependence on the sample complexity for the much broader class of functions using our new techniques.

We first consider the class of functions f to be the class of polynomial functions in d variables with a bounded degree Δ. Our sample complexity bound only has a logarithmic dependence on Δ, implying that they are meaningful even when the number of oscillations is exponentially large. Our overall approach is to show that location of the i-th iterate x_i can be expressed as a bounded degree polynomial in η using an inductive argument. This allows us to bound the number of intervals of η for which Algorithm 1 converges in i steps for any $1 \leq i \leq H$. We can finally also bound the number of pieces of the dual cost function $\ell_{x,f}$ where gradient descent fails to converge. Formally, we have the following theorem (all omitted proofs are in Appendix D).

Theorem 3.1. *Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of polynomial functions $\mathbb{R}^d \to \mathbb{R}$ of degree at most $\Delta \geq 1$. Then (ϵ,δ)-uniform convergence is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ using $m = O\left(\frac{H^2}{\epsilon^2}(H\log\Delta + \log\frac{1}{\delta})\right)$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta), g_i^{(2)}(\eta), \ldots, g_i^{(d)}(\eta))$ where $g_i^{(j)}$ is a polynomial function with degree at most Δ^{i-2}. We will show this by induction.

Base case: $i = 2$. $x_2 = x_1 - \eta\nabla f(x_1) = x - \eta\nabla f(x)$ is a polynomial of degree $1 = \Delta^{2-2}$ in η in each coordinate.

Inductive case: $i > 2$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(\eta) = (g_{i-1}^{(j)}(\eta))_{j\in[d]}$ where $g_{i-1}^{(j)}$ is a polynomial of degree at most Δ^{i-3} (inductive hypothesis). Now $x_i = x_{i-1} - \eta\nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(\eta) - \eta\nabla f(\boldsymbol{g}_{i-1}(\eta)) =: \boldsymbol{g}_i(\eta)$. Clearly, $\boldsymbol{g}_i^{(j)}$ is a polynomial in η, with degree at most $\Delta^{i-3}(\Delta-1)+1 = \Delta^{i-2} - \Delta^{i-3} + 1 \leq \Delta^{i-2}$.

Thus, for any fixed $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(\eta)$ where each coordinate $g_i^{(j)}$ is a polynomial with degree at most Δ^{i-2}. For a fixed initial point x, this implies that $\nabla f(x_i)$ is a polynomial in η of degree at most Δ^{i-1} in each coordinate. Thus, $\|\nabla f(x_i)\|^2$ is a polynomial of degree at most $2\Delta^{i-1}$ in η. Now, consider the set of points η for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant θ. This consists of at most $O(\Delta^{i-1})$ intervals.

Furthermore, we note that for any η, the cost is determined by the smallest i such that $\|\nabla f(x_i)\| < \theta$ (if one exists). Since the cost takes only discrete values,

Parameter	Setting	Sample complexity bound	Result
Stepsize	Convex+smooth+guaranteed progress	$\tilde{O}\left(\frac{H^3}{\epsilon^2}\right)$	GR2016
Stepsize	Polynomial with degree Δ	$\tilde{O}\left(\frac{H^3 \log \Delta}{\epsilon^2}\right)$	Theorem 3.1
Stepsize	Piecewise-polynomial with degree Δ and p polynomial boundaries	$\tilde{O}\left(\frac{H^3 \log(pd\Delta)}{\epsilon^2}\right)$	Theorem 3.2
Stepsize	Pfaffian with chain length q, degree Δ and Pfaffian degree M	$\tilde{O}\left(\frac{H^2(q^2d^2H^2+qdH\log(\Delta+M))}{\epsilon^2}\right)$	Theorem 3.3
Learning-schedule	Piecewise-polynomial with degree Δ and p polynomial boundaries	$\tilde{O}\left(\frac{H^4 \log(pd\Delta)}{\epsilon^2}\right)$	Theorem 4.1
Initialization scale σ	Piecewise-polynomial with degree Δ and p polynomial boundaries	$\tilde{O}\left(\frac{H^3 \log(pd\Delta)}{\epsilon^2}\right)$	Theorem 4.2
Initialization vector	Piecewise-polynomial with degree Δ and p polynomial boundaries	$\tilde{O}\left(\frac{dH^3 \log(pd\Delta)}{\epsilon^2}\right)$	Theorem 4.3
Stepsize+ momentum	Piecewise-polynomial with degree Δ and p polynomial boundaries	$\tilde{O}\left(\frac{H^3 \log(pd\Delta)}{\epsilon^2}\right)$	Theorem 4.4
Stepsize	Objective: Quality of final iterate; Piecewise-polynomial with degree Δ and p polynomial boundaries	$\tilde{O}\left(\frac{H^2(H\log(pd\Delta)+\log \Delta_v p_v)}{\epsilon^2}\right)$	Theorem 5.1

Table 1: Summary of sample complexity bounds for tuning hyperparameters in gradient-based optimization. The objective is number of steps to convergence, unless stated otherwise. d denotes the dimension of points x.

it is a piecewise-constant function of η, and we seek to bound the number of these pieces. A naive counting argument gives a $O(\Pi_{i=1}^{H}\Delta^{i-1}) = O(\Delta^{H^2})$ bound on the number of pieces, since if there are K_{i-1} intervals corresponds to values of η for which the algorithm converges within $i-1$ steps, then each of the $O(\Delta^{i-1})$ intervals in round i computed above may result in at most $K_{i-1}+1$ new pieces. We can, however, use an amortized counting argument to give a tighter bound. Indeed, suppose there are K_{i-1} intervals with different values of cost $\leq i-1$. Of the new $O(\Delta^{i-1})$ intervals say T_i intersect at least one of the existing pieces, resulting in at most $T_i + K_{i-1}$ new pieces overall. Thus, the total number of pieces of the cost function with cost $\leq i$ is $K_i \leq (T_i + K_{i-1}) + O(\Delta^{i-1}) - T_i + K_{i-1} \leq O(\Delta^{i-1}) + 2K_{i-1}$. Thus, across i, we have at most $O(\sum_{i=1}^{H} 2^{H-i}\Delta^{i-1}) = O(\Delta^H)$ intervals over each of which Algorithm 1 converges with a constant number of steps. The cost over the remainder of the domain $\mathcal{P}$ where the algorithm does not converge, which consists of $O(\Delta^H)$ intervals, is H.

Thus, using Lemma 2.2, since each function $\ell_{x,f}$ is $O(\Delta^H)$-monotonic, we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H\log\Delta)$, which implies the stated sample complexity. □

The uniform convergence guarantee in the above result is quite strong. It implies that we only need to find an approximately optimal η on the training set, and our guarantees will imply a small generalization error. We further extend the result to piecewise polynomial functions with polynomial boundaries. For simplicity, we assume that the gradient descent procedure never lands exactly on any boundary (necessary for Algorithm 1 to be well-defined).

Theorem 3.2. *Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ using $m = O\left(\frac{H^2}{\epsilon^2}(H\log(pd\Delta) + \log\frac{1}{\delta})\right)$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta), g_i^{(2)}(\eta), \ldots, g_i^{(d)}(\eta))$ where each $g_i^{(j)}$ is a piecewise-polynomial function with degree at most Δ^{i-2} and at most $(2pd\Delta)^{i-2}$ pieces. We will show this by induction.

Base case: $i = 2$. $x_2 = x_1 - \eta\nabla f(x_1) = x - \eta\nabla f(x)$ is a polynomial of degree $1 = \Delta^{2-2}$ in η in each coordinate.

Inductive case: $i > 2$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(\eta) =$

$(g_{i-1}^{(j)}(\eta))_{j\in[d]}$ where $g_{i-1}^{(j)}$ is piecewise-polynomial with at most $(2pd\Delta)^{i-3}$ pieces and degree at most Δ^{i-3} (inductive hypothesis). Now $x_i = x_{i-1} - \eta\nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(\eta) - \eta\nabla f(\boldsymbol{g}_{i-1}(\eta)) =: \boldsymbol{g}_i(\eta)$. Now, $\nabla f(\boldsymbol{g}_{i-1}(\eta))$ is piecewise-polynomial. The degree is at most $(\Delta - 1)\Delta^{i-3} \leq \Delta^{i-2} - 1$. Any critical point is either a critical point of some $g_{i-1}^{(j)}$, or a solution of the equation $f^{(k)}(g_{i-1}^{(1)}(\eta), \ldots, g_{i-1}^{(d)}(\eta)) = 0$ for some boundary function $f^{(k)}$ ($k \in [p]$) of f. The total number of critical points of $\boldsymbol{g}_i^{(j)}$ (for any $j \in [d]$) is therefore at most $(2pd\Delta)^{i-3} + (pd\Delta)(2pd\Delta)^{i-3} \leq (2pd\Delta)^{i-2}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in η, with degree at most Δ^{i-2} and at most $(2pd\Delta)^{i-2}$ pieces.

Thus, for any fixed $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(\eta)$ where each coordinate $g_i^{(j)}$ is piecewise-polynomial with degree at most Δ^{i-2} and at most $(2pd\Delta)^{i-2}$ pieces. Using the argument above, for any fixed initial point x, $\|\nabla f(x_i)\|^2$ is piecewise-polynomial with degree at most $2\Delta^{i-1}$ in η and at most $(2pd\Delta)^{i-1}$ pieces. Now, consider the set of points η for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant θ. This consists of at most $O((2pd\Delta)^{i-1})$ intervals.

We can now apply the amortized counting argument from the proof of Theorem 3.1 to conclude that the number of pieces in the dual cost function is $O((2pd\Delta)^H)$ here.

Thus, using Lemma 2.2, we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H \log pd\Delta)$, which implies the stated sample complexity. □

The case of piecewise-polynomial functions is particularly interesting as it captures gradient descent for an important class of feedforward neural networks.

Example 1. *For deep neural networks with piecewise polynomial activation functions, the network computes a piecewise polynomial function of its weights on any fixed input x Bartlett et al. [1998]. Therefore the MSE loss of the network on a given dataset is also a piecewise polynomial function of its weights. Therefore our results for tuning the gradient descent step-size above apply in this case. Concretely, Theorem 3.2 implies a sample complexity bound of $\tilde{O}\left(\frac{H^3 L \log k}{\epsilon^2}\right)$, where L is the number of layers in the network and k is the number of nodes (using Bartlett et al. [1998]). The piecewise-polynomial structure also holds if we add regularization terms related to flatness of the minima e.g. $\|\nabla^2 f\|$ to the loss function.*

We note that our techniques can be used to establish sample complexity bounds for tuning the entire learning rate schedule (see Theorem 4.1).

3.2 Beyond polynomial functions

We extend our techniques beyond polynomial functions to a much broader class of Pfaffian functions. This will allow us to give guarantees for gradient descent for learning neural networks with non-polynomial activation functions including sigmoid and tanh. We will employ tools used for the analysis of Pfaffian functions [Karpinski and Macintyre, 1997, Balcan et al., 2025a].

Intuitively, Pfaffian functions are functions for which the derivatives can be expressed as a polynomial of the variables and the function itself (or other functions from a sequence of functions, as explained below). Formally, we first need the notion of a *Pfaffian chain*. Roughly speaking, it consists of an ordered sequence of functions, in which the derivative of each function can be represented as a polynomial of the variables and previous functions in the sequence (including the function itself).

Definition 2 (Pfaffian Chain, Khovanski, 1991)**.** *A finite sequence of continuously differentiable functions $\eta_1, \ldots, \eta_q : \mathbb{R}^d \to \mathbb{R}$ and variables $\boldsymbol{a} = (a_1, \ldots, a_d) \in \mathbb{R}^d$ form a Pfaffian chain $\mathcal{C}(\boldsymbol{a}, \eta_1, \ldots, \eta_q)$ if there are real polynomials $P_{i,j}(\boldsymbol{a}, \eta_1, \ldots, \eta_j)$ in $a_1, \ldots, a_d, \eta_1, \ldots, \eta_j$, for all $i \in [d]$ and $j \in [q]$, such that*

$$\frac{\partial \eta_j}{\partial a_i} = P_{i,j}(\boldsymbol{a}, \eta_1, \ldots, \eta_j).$$

Note that $P_{i,j}(\boldsymbol{a}, \eta_1, \ldots, \eta_j)$ is a polynomial in $\boldsymbol{a}$ and the functions $\eta_1(\boldsymbol{a}), \ldots, \eta_j(\boldsymbol{a})$. Pfaffian chains are associated with a *length* and a *Pfaffian degree*, that dictate their complexity. The length of a Pfaffian chain is the number of functions q that appear on that chain, while the Pfaffian degree of a chain is the maximum degree of polynomials $P_{i,j}$ that can be used to express the partial derivative of functions on that chain. Given a Pfaffian chain, one can define the Pfaffian function as a polynomial of variables and functions on that chain.

Definition 3 (Pfaffian functions, Khovanski, 1991)**.** *Given a Pfaffian chain $\mathcal{C}(\boldsymbol{a}, \eta_1, \ldots, \eta_q)$, as defined in Definition 2, a Pfaffian function over the chain $\mathcal{C}$ is a function of the form $g(\boldsymbol{a}) = Q(\boldsymbol{a}, \eta_1, \ldots, \eta_q)$, where Q is a polynomial in variables $\boldsymbol{a}$ and functions $\eta_1, \ldots, \eta_q$ in the chain $\mathcal{C}$. The degree Δ of the Pfaffian function $g(\boldsymbol{a})$ is the degree of the polynomial $Q(\boldsymbol{a}, \eta_1, \ldots, \eta_q)$.*

For example, $\frac{de^x}{dx} = e^x$, and therefore the function $g(x) = e^x$ is a Pfaffian function of degree $\Delta = 1$, associated with a Pfaffian chain of length 1 and Pfaffian degree 1. Note that polynomial functions are Pfaffian with degree equal to their usual degree and associated with a chain of length zero. We show the following generalization of Theorem 3.1.

Theorem 3.3. *Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of Pfaffian functions $\mathbb{R}^d \to \mathbb{R}$ of degree at most $\Delta \geq 1$, associated with Pfaffian chains of length at most q and Pfaffian degree at most M. Then (ϵ, δ)-uniform convergence is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ using $m = O\left(\frac{H^2}{\epsilon^2}(q^2d^2H^2 + qdH\log(\Delta + M) + \log\frac{1}{\delta})\right)$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. Fix a function $f \in \mathcal{F}$ and initial point x. Suppose $f(x) = Q(x, f_1(x), \ldots, f_q(x))$ is associated with the Pfaffian chain $\mathcal{C}(x, f_1, \ldots, f_q)$ of length at most q, with degree of Q at most Δ. Note that, by the definition of Pfaffian functions, each coordinate of $\nabla f(x)$ is a polynomial in $x, f_1, \ldots, f_q$ of degree at most $\Delta + M - 1$.

We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta), g_i^{(2)}(\eta), \ldots, g_i^{(d)}(\eta))$ where $g_i^{(j)}$ is a Pfaffian function with degree at most $(\Delta + M)^{i-2}$, and associated with the Pfaffian chain

$$\mathcal{C}_i := \mathcal{C}(\eta, f_1, \ldots, f_q, (g_2^{(j)})_{j\in[d]}, (f_k(g_2^{(j)}))_{k\in[q], j\in[d]}, \ldots, (g_{i-1}^{(j)})_{j\in[d]}, (f_k(g_{i-1}^{(j)}))_{k\in[q], j\in[d]})$$

of length at most $q + (q+1)d(i-2)$ as the function f. We will show this by induction.

Base case: $i = 2$. $x_2 = x_1 - \eta\nabla f(x_1) = x - \eta\nabla f(x)$ is a polynomial of degree $1 = (\Delta + M)^{2-2}$ in η in each coordinate.

Inductive case: $i > 2$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(\eta) = (g_{i-1}^{(j)}(\eta))_{j\in[d]}$ where $g_{i-1}^{(j)}$ is a Pfaffian function of degree at most $(\Delta+M)^{i-3}$ and associated with the chain $\mathcal{C}_{i-1}$ (inductive hypothesis).

Now $x_i = x_{i-1} - \eta\nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(\eta) - \eta\nabla f(\boldsymbol{g}_{i-1}(\eta)) =: \boldsymbol{g}_i(\eta)$. By the definition of Pfaffian functions, $(\nabla f(\boldsymbol{g}_{i-1}(\eta)))_j$ is a polynomial in $\boldsymbol{g}_{i-1}(\eta), f_1(\boldsymbol{g}_{i-1}(\eta)), \ldots, f_q(\boldsymbol{g}_{i-1}(\eta))$ with degree at most $\Delta + M - 1$, and therefore x_i has degree at most $(\Delta + M)^{i-3}(\Delta + M - 1) + 1 \leq (\Delta + M)^{i-2}$ in the chain $\mathcal{C}_i$.

Thus, for any fixed $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(\eta)$ where each coordinate $g_i^{(j)}$ is a Pfaffian function with degree at most $(\Delta + M)^{i-2}$, associated with the chain $\mathcal{C}_i$. For a fixed initial point x, this implies that $\|\nabla f(x_i)\|^2$ is a Pfaffian function of degree at most $2(\Delta + M)^{i-1}$ in the chain $\mathcal{C}_i$ of length $O(qdi)$. Now, consider the set of points η for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant θ. By a standard connected components bound for Pfaffian functions (e.g. see Corollary B.3, [Balcan et al., 2025a]), this consists of at most $2^{O(q^2d^2i^2)} \cdot (\Delta + M)^{O(qdi)}$ intervals.

By the counting argument in the proof of Theorem 3.1, across $i \in [H]$, we have $2^{O(q^2d^2H^2)}(\Delta+M)^{O(qdH)}$ intervals over each of which Algorithm 1 converges with a constant number of steps (or does not converge and the cost is H). Thus, using Lemma 2.2, we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(q^2d^2H^2 + qdH\log(\Delta + M))$, which implies the stated sample complexity. □

A key technical insight in the above proof is the careful construction of the Pfaffian chain $\mathcal{C}_i$. A naive construction would include all function composition sequences $f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_n}$ in the chain, where f_{i_j} are functions from the chain $f_1, \ldots, f_q$ and $n \in [H]$, resulting in an exponential (in q) upper-bound on the sample complexity. We note that the above result applies to tuning the learning rate in gradient descent for neural networks with other commonly used activations, including the sigmoid activation $\sigma(x) = 1/(1 + e^{-x})$, and the tanh activation $\tau(x) = (e^x - e^{-x})/(e^x + e^{-x})$.

Remark 1. *For deep networks with sigmoid (or tanh) activation functions, the network computes a piecewise Pfaffian function of its weights on any fixed input x Karpinski and Macintyre [1997]. The MSE loss of the network on a given dataset is also a piecewise Pfaffian function of its weights. Our results for tuning the gradient descent step-size above apply in this case. Concretely, Theorem 3.2 implies a sample complexity bound of $\tilde{O}\left(\frac{H^4k^4}{\epsilon^2}\right)$, where k is the number of nodes (using results from Karpinski and Macintyre [1997]).*

We conclude this section with some more examples and some non-examples of Pfaffian functions. All polynomials and ratios of polynomials are Pfaffian, as well as polynomials in the exponential and logarithm function. Functions involving fractional exponents (e.g. $f(x) = x^{1/2} + x^{1/3}$) are also Pfaffian. However, periodic trigonometric functions like sine and cosine are not Pfaffian (although non-periodic inverse trigonometric functions, like arctan are).

4 TUNING MULTIPLE HYPERPARAMETERS IN GRADIENT-BASED OPTIMIZATION

We will now extend our framework from tuning a single stepsize parameter to multiple parameters. First, we bound the sample complexity of tuning the entire learning rate schedule from data. For piecewise-polynomial functions, we show that a slightly larger $\tilde{O}(H^4/\epsilon^2)$ bound on the number of samples is sufficient to learn the entire step-size schedule for gradient descent. Next, we show the generality of our framework by showing that it can be used to learn how to initialize gradient descent. In the context of using gradient descent for tuning the weights of neural networks, this

implies a bound on the sample complexity of effective pre-training (we focus on efficient convergence here but our framework extends to optimizing the quality of the final iterate, see Section 5). Finally, we show that our framework can be used to analyze the tuning of hyperparameters beyond vanilla gradient descent, by analyzing simultaneous tuning of the learning rate and momentum hyperparameters.

4.1 Learning the step-size schedule

Designing a good learning rate schedule is considered a crucial problem in gradient-based iterative optimization. Our framework allows learning an iterate-dependent learning rate. That is, we set the learning rate to η_i for $i \in [H]$, and we learn the sequence η_i from data.

Theorem 4.1. *Consider a variant of Algorithm 1, where a different step-size η_i is used in the i-th update, i.e., $x_{i+1} = x_i - \eta_i \nabla f(x_i)$ with parameters η_i chosen from some continuous set $\mathcal{P} \subset \mathbb{R}^H_{\geq 0}$. Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all $(\eta_1, \ldots, \eta_H) \in \mathcal{P}$ using $m = O(\frac{H^2}{\epsilon^2}(H^2 \log(pd\Delta) + \log \frac{1}{\delta}))$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta_1, \ldots, \eta_{i-1}), \ldots, g_i^{(d)}(\eta_1, \ldots, \eta_{i-1}))$ where each $g_i^{(j)}$ is a piecewise-polynomial function with degree at most Δ^{i-2} and at most $(2pd)^{i-2}$ algebraic boundaries of degree at most Δ^{i-2}. We show this by induction.

Base case: $i = 2$. $x_2 = x_1 - \eta_1 \nabla f(x_1) = x - \eta_1 \nabla f(x)$ is a polynomial of degree $1 = \Delta^{2-2}$ in η_1 in each coordinate.

Inductive case: $i > 2$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(\boldsymbol{\eta}_{i-2}) = (g_{i-1}^{(j)}(\eta_1, \ldots, \eta_{i-2}))_{j \in [d]}$ where $g_{i-1}^{(j)}$ is piecewise-polynomial with at most $(2pd)^{i-3}$ boundaries and degree at most Δ^{i-3} (for both pieces and boundaries, by the inductive hypothesis).

Now $x_i = x_{i-1} - \eta_{i-1} \nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(\boldsymbol{\eta}_{i-2}) - \eta_{i-1} \nabla f(\boldsymbol{g}_{i-1}(\boldsymbol{\eta}_{i-2})) =: \boldsymbol{g}_i(\boldsymbol{\eta}_{i-1})$. Now, $\nabla f(\boldsymbol{g}_{i-1}(\boldsymbol{\eta}_{i-2}))$ is piecewise-polynomial. The degree of is at most $(\Delta - 1)\Delta^{i-3} \leq \Delta^{i-2} - 1$. Any discontinuity point is either a discontinuity point of some $g_{i-1}^{(j)}$, or a solution of the equation $f^{(k)}(g_{i-1}^{(1)}(\boldsymbol{\eta}_{i-2}), \ldots, g_{i-1}^{(d)}(\boldsymbol{\eta}_{i-2})) = 0$ for some boundary function $f^{(k)}$ ($k \in [p]$) of f. The total number of boundary functions of $\boldsymbol{g}_i^{(j)}$ (for any $j \in [d]$) is therefore at most $d(2pd)^{i-3} + (pd)(2pd)^{i-3} \leq (2pd)^{i-2}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in $\boldsymbol{\eta}_{i-1}$, with degree at most Δ^{i-2} and at most $(2pd\Delta)^{i-2}$ pieces.

Thus, for any fixed $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(\boldsymbol{\eta}_{i-1})$ where each coordinate $g_i^{(j)}$ is piecewise-polynomial with degree at most Δ^{i-2} and at most $(2pd)^{i-2}$ algebraic boundaries. Using the argument above, for any fixed initial point x, $\|\nabla f(x_i)\|^2$ is piecewise-polynomial with degree at most $2\Delta^{i-1}$ in $\boldsymbol{\eta}_{i-1}$ and at most $(2pd\Delta)^{i-1}$ boundaries. This implies that the dual cost function can be computed using a GJ algorithm with degree at most $O(\Delta^H)$ and predicate complexity $O(\sum_i (2pd)^i) = (2pd)^{O(H)}$. Using Bartlett et al. [2022], we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H^2 \log pd\Delta)$, which implies the stated sample complexity. □

4.2 Learning to initialize

Another crucial parameter that impacts the convergence of gradient descent is the initial point x. For example, one may set the initial weights of a neural network by drawing each coordinate according to the Gaussian distribution $\mathcal{N}(0, \sigma^2)$, where σ is an important hyperparameter that must be carefully set [LeCun et al., 1998, Glorot and Bengio, 2010, He et al., 2015]. Here, we will treat the norm of the initial point x as the *initialization scale* parameter, i.e. $x = \sigma \hat{x}$, where $\|\hat{x}\| = 1$ and $\sigma > 0$.

Theorem 4.2. *Consider a variant of Algorithm 1, where the learning rate η is fixed, but the initial point is given by $x = \sigma \hat{x}$ for some fixed $\hat{x}$ and hyperparameter $\sigma \in \mathbb{R}_{>0}$. Suppose the instance space Π consists of $(\hat{x} \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all σ using $m = O(\frac{H^2}{\epsilon^2}(H \log(pd\Delta) + \log \frac{1}{\delta}))$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

We can also learn the entire initialization vector x, with an extra factor of d in the sample complexity. This is remarkable, as in the context of neural networks this corresponds to learning a pre-trained weight for a collection of tasks (given by the task distribution $\mathcal{D}$).

Theorem 4.3. *Consider a variant of Algorithm 1, where the learning rate η is fixed, but the initial point is given by $x \in \mathbb{R}^d$ (a hyperparameter). Suppose the instance space Π consists of $(f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all $x \in \mathbb{R}^d$ using $m =$*

Algorithm 2 Momentum-based gradient descent(step size η, momentum parameter γ)

Input: Initial point x, function to minimize f, maximum number of iterations H, gradient threshold for convergence θ

```
Initialize x_1 ← x, y_1 ← 0
for i = 1, ..., H do
    if ||∇f(x_i)|| < θ then
        Return x_i
    y_{i+1} = γ y_i − η ∇f(x_i)
    x_{i+1} = x_i + y_i
```

Output: Return x_{H+1}

$O(\frac{H^2}{\epsilon^2}\left(dH\log(pd\Delta)+\log\frac{1}{\delta}\right))$ *samples from* $\mathcal{D}$ *for any distribution* $\mathcal{D}$ *over* Π.

4.3 Beyond vanilla gradient descent

We will now show that our technique extends beyond tuning the learning rate in gradient descent to tuning relevant hyperparameters in other popular iterative gradient based optimization methods, showing the versatile applicability of our analytical framework. In particular, we will study tuning of the momemtum and learning rate parameters γ, η simultaneously in Algorithm 2. Momentum Rumelhart et al. [1986] takes an exponentially weighted average of the gradients to update the points at each iteration, and is particularly important for optimizing non-convex functions. It is widely used in practice and is a part of optimizers like Adam[1].

Theorem 4.4. *Consider the problem of tuning the* η, γ *in Algorithm 2 over some continuous set* $\mathcal{P} \subset \mathbb{R}^2_{\geq 0}$. *Suppose the instance space* Π *consists of* $(x \in \mathbb{R}^d, f \in \mathcal{F})$, *where* $\mathcal{F}$ *consists of functions* $\mathbb{R}^d \to \mathbb{R}$ *that are piecewise-polynomial with at most* p *polynomial boundaries, with the maximum degree of any piece function or boundary function at most* $\Delta \geq 1$. *Then* (ϵ, δ)*-uniform convergence is achieved for all* $\gamma, \eta \in \mathcal{P}$ *using* $m = O\left(\frac{H^2}{\epsilon^2}(H\log(pd\Delta)+\log\frac{1}{\delta})\right)$ *samples from* $\mathcal{D}$ *for any distribution* $\mathcal{D}$ *over* Π.

Note the logarithmic dependence on the degree and dimensionality in our bounds. This makes our bounds meaningful for tuning networks with a large number weights d and a large number of layers L (Δ is typically exponential in L, so our bounds imply a linear dependence on L).

[1]Full implementation of Adam is more complicated and involves normalization, we only analyze the simpler version given by Algorithm 2.

5 BEYOND CONVERGENCE RATES

Let $\tilde{x}$ denote the final point output by Algorithm 1. The quality of optimum learned is given by some function $g(\tilde{x})$, which may be different from the function f on which gradient descent was performed.

Formally, a problem instance is given by a tuple (x, f, f_v) consisting of an initial point $x \in \mathbb{R}^d$, and $f, f_v : \mathbb{R}^d \to \mathbb{R}$ denoting the *training* and *validation* functions respectively. Gradient descent (Algorithm 1) is performed using x as the initial point, with f as the function to minimize and with access to ∇f. The output of gradient descent on the training function, $\tilde{x}$, is evaluated using the validation function f_v. For example, $f_v(x)$ could denote the loss of the neural network with weights x on the validation set.

Theorem 5.1. *Suppose the instance space* Π *consists of* $(x \in \mathbb{R}^d, f \in \mathcal{F}, f_v \in \mathcal{F}_v)$, *where* $\mathcal{F}$ *(resp.* $\mathcal{F}_v$*) consists of functions* $\mathbb{R}^d \to \mathbb{R}$ *that are piecewise-polynomial with at most* p *(resp.* p_v*) polynomial boundaries, with the maximum degree of any piece function or boundary function at most* $\Delta \geq 1$ *(resp.* Δ_v*). Then* (ϵ, δ)*-uniform convergence w.r.t. the validation loss is achieved for all step-sizes* $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ *using* $m = O\left(\frac{H^2}{\epsilon^2}(H\log(pd\Delta)+\log\frac{\Delta_v p_v}{\delta})\right)$ *samples from* $\mathcal{D}$ *for any distribution* $\mathcal{D}$ *over* Π.

6 CONCLUSION

We develop a new framework for analyzing the sample complexity of tuning the hyperparameters in gradient-based optimization. While prior theoretical techniques are largely limited in their scope to convex and smooth functions, we significantly expand the set of functions for which one can provably tune the stepsize of gradient descent including piecewise-polynomial and piecewise-Pfaffian functions. Our results imply finite (polynomial) bounds on the sample complexity of tuning the stepsize and stepsize schedules, and also apply to learning a pre-trained network using gradient descent.

For future work, it would be interesting to obtain a better understanding of the computational complexity and lower bounds. For example, can we give efficient exact or approximate implementations of the Empirical Risk Minimization (ERM) algorithm for tuning the learning rate from multiple instances in the training set? The tightness with respect to various terms that appear in our bounds is another very interesting question. Tight lower bounds are known only in a few cases in prior work on data-driven parameter tuning.

Acknowledgements

This work involved extensive discussions with Aravindan Vijayaraghavan throughout the course of its development. This work was supported in part by the National Science Foundation under grants ECCS-2216899 and ECCS-2216970.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Martin Anthony and Peter Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 1999.

Maria-Florina Balcan. Data-Driven Algorithm Design (book chapter). In *Beyond Worst-Case Analysis of Algorithms, Tim Roughgarden (Ed)*. Cambridge University Press, 2020.

Maria-Florina Balcan and Dravyansh Sharma. Data driven semi-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 14782–14794, 2021.

Maria-Florina Balcan and Dravyansh Sharma. Learning accurate and interpretable decision trees. In *Uncertainty in Artificial Intelligence (UAI)*, pages 288–307. PMLR, 2024.

Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Conference on Learning Theory*, pages 213–274. PMLR, 2017.

Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614. IEEE, 2018.

Maria-Florina Balcan, Travis Dick, and Dravyansh Sharma. Learning piecewise Lipschitz functions in changing environments. In *International Conference on Artificial Intelligence and Statistics*, pages 3567–3577. PMLR, 2020.

Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? Generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 919–932, 2021a.

Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. Learning-to-learn non-convex piecewise-Lipschitz functions. *Advances in Neural Information Processing Systems*, 34: 15056–15069, 2021b.

Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. Provably tuning the ElasticNet across instances. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27769–27782. Curran Associates, Inc., 2022.

Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch: Generalization guarantees and limits of data-independent discretization. *Journal of the ACM (JACM)*, 2024a.

Maria-Florina Balcan, Christopher Seiler, and Dravyansh Sharma. Accelerating ERM for data-driven algorithm design using output-sensitive techniques. *Advances in Neural Information Processing Systems*, 37:72648–72687, 2024b.

Maria-Florina Balcan, Anh Tuan Nguyen, and Dravyansh Sharma. Algorithm configuration for structured Pfaffian settings. *Transactions of Machine Learning Research (TMLR)*, 2025a.

Maria-Florina Balcan, Anh Tuan Nguyen, and Dravyansh Sharma. Sample complexity of data-driven tuning of model hyperparameters in neural networks with structured parameter-dependent dual function. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Adnan Barakat, Alexander Chatzimichailidis, Leonard Berrada, Anton Obukhov, and Aurelien Lucchi. Understanding and improving adamw for faster optimization. *arXiv preprint arXiv:2102.10482*, 2021. URL `https://arxiv.org/abs/2102.10482`.

Peter Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear VC dimension bounds for piecewise polynomial networks. *Neural Information Processing Systems*, 11, 1998.

Peter Bartlett, Piotr Indyk, and Tal Wagner. Generalization bounds for data-driven numerical linear algebra. In *Conference on Learning Theory (COLT)*, pages 2013–2040. PMLR, 2022.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations*, 2018.

Avrim Blum, Chen Dan, and Saeed Seddighin. Learning complexity of simulated annealing. In *International Conference on Artificial Intelligence and*

Statistics (AISTATS), pages 1540–1548. PMLR, 2021.

Sebastian Bock and Michael Weiß. Non-convergence and limit cycles in the adam optimizer. *arXiv preprint arXiv:2210.02070*, 2022. URL https://arxiv.org/abs/2210.02070.

Sebastian Bock, Johannes Goppold, and Michael Weiß. An improvement of the convergence proof of the adam-optimizer. *arXiv preprint arXiv:1804.10587*, 2018. URL https://arxiv.org/abs/1804.10587.

Yair Carmon and Oliver Hinder. Making SGD parameter-free. In *Conference on Learning Theory (COLT)*, pages 2360–2389. PMLR, 2022.

Jing Chen, Shiqiang Wang, Jianmin Zhang, and Bo Li. An iterative k-fac algorithm for deep learning. *arXiv preprint arXiv:2101.00218*, 2021. URL https://arxiv.org/abs/2101.00218.

Hongyu Cheng and Amitabh Basu. Learning cut generating functions for integer programming. *Advances in Neural Information Processing Systems*, 37:61455–61480, 2024.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018. URL https://arxiv.org/abs/1812.07956.

Hadi Daneshmand, Jonas M Kohler, Aurelien Lucchi, and Thomas Hofmann. Escaping saddle points with stochastic gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by D-adaptation. In *International Conference on Machine Learning (ICML)*, pages 7449–7479. PMLR, 2023.

Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:9974–10007, 2024.

Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. *International Conference on Machine Learning (ICML)*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. URL https://arxiv.org/abs/1810.04805.

Laurent Dinh, Razvan Pascanu, and Yoshua Bengio. Momentum is critical for stochastic gradient descent in deep learning: Understanding generalization and implicit bias. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=SkxJYoBFPH.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A Simple Convergence Proof of Adam and Adagrad. *arXiv preprint arXiv:2003.02395*, 2020. URL https://arxiv.org/abs/2003.02395.

Rong Ge, Jason D Lee, and Tengyu Ma. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *International Conference on Machine Learning (ICML)*, pages 1233–1242, 2017.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013. technical report / arXiv:1309.5549.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

Paul Goldberg and Mark Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 361–369, 1993.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations (ICLR)*, 2015.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 123–134, 2016.

Vaibhav Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. *arXiv preprint arXiv:1802.09568*, 2018. URL https://arxiv.org/abs/1802.09568.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*, 2020. URL https://arxiv.org/abs/2004.10964.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing

human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

João F Henriques and Andrea Vedaldi. Small steps and giant leaps: Minimal newton solvers for deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. URL `https://arxiv.org/abs/1805.08095`.

Maor Ivgi, Oliver Hinder, and Yair Carmon. DoG is SGD's best friend: A parameter-free dynamic step size schedule. In *International Conference on Machine Learning (ICML)*, pages 14465–14499. PMLR, 2023.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. URL `https://arxiv.org/abs/1806.07572`.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. URL `https://arxiv.org/abs/1711.09846`.

Xianqi Jiao, Jia Liu, and Zhiping Chen. Learning complexity of gradient descent and conjugate gradient algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17671–17679, 2025.

Billy Jin, Thomas Kesselheim, Will Ma, and Sahil Singla. Sample complexity of posted pricing for a single item. In *Advances in Neural Information Processing Systems*, 2024.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1724–1732, 2017.

Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. *Advances in Neural Information Processing Systems*, 37:111760–111801, 2024.

Hadi Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 795–803, 2016.

Marek Karpinski and Angus Macintyre. Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, 54(1):169–176, 1997.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Mikhail Khodak, Edmond Chow, Maria-Florina Balcan, and Ameet Talwalkar. Learning to relax: Setting solver parameters across a sequence of linear system instances. *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Askold G Khovanski. *Fewnomials*, volume 88. American Mathematical Soc., 1991.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL `https://arxiv.org/abs/1412.6980`.

Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert Múller. Efficient backprop in neural networks: Tricks of the trade, 1998.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. *Journal of Machine Learning Research*, 18(1):7750–7808, 2017.

Ke Li and Jitendra Malik. Learning to optimize. *International Conference on Learning Representations (ICLR)*, 2017.

Xiang Li, Feng Wei, Shiji Xu, and Heng Zhang. Rethinking the hyperparameters for adam. *arXiv preprint arXiv:2006.08217*, 2020. URL `https://arxiv.org/abs/2006.08217`.

Wu Lin, Scott C Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger B Grosse. Understanding and improving the shampoo optimizer via kullback-leibler minimization. *arXiv preprint arXiv:2509.03378*, 2025.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of NAACL*, 2019. URL `https://arxiv.org/abs/1903.08855`.

Xiang Liu, Hao Yuan, and Tengyu Ma. Adam-type algorithms converge for non-convex optimization: Theory and empirical evidence. *arXiv preprint arXiv:2110.11680*, 2021. URL `https://arxiv.org/abs/2110.11680`.

Jonathan Lorraine, Petar Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. *arXiv preprint arXiv:2002.11702*, 2020. URL `https://arxiv.org/abs/2002.11702`.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Dougal Maclaurin, David Kristjanson Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. *International Conference on Machine Learning (ICML)*, 2015.

Fee Nern, Jimmy Ba, Gintare Karolina Dziugaite, Daniel M Roy, and Volodymyr Mnih. On transfer of adversarial robustness from pretraining to downstream tasks. *arXiv preprint arXiv:2208.03835*, 2022. URL `https://arxiv.org/abs/2208.03835`.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 2nd edition, 2006. ISBN 9780387400655.

Samet Oymak. Provable super-convergence with a large cyclical learning rate. *IEEE Signal Processing Letters*, 28:1645–1649, 2021.

Jonathan Pauloski, Adil Al-Hawaj, Manya Singh, et al. Kaisa: An adaptive second-order optimizer framework for deep neural networks. *arXiv preprint arXiv:2107.01739*, 2021. URL `https://arxiv.org/abs/2107.01739`.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. *arXiv preprint arXiv:1602.02355*, 2016. URL `https://arxiv.org/abs/1602.02355`.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *NeurIPS*, 2017.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *NeurIPS*, 2016. URL `https://arxiv.org/abs/1606.05340`.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Shinsaku Sakaue and Taihei Oki. Generalization bound and learning methods for data-driven projections in linear programming. *Advances in Neural Information Processing Systems*, 37:12825–12846, 2024.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013. URL `https://arxiv.org/abs/1312.6120`.

Dravyansh Sharma. *Data-driven algorithm design and principled hyperparameter tuning in machine learning*. PhD thesis, Carnegie Mellon University, 2024a.

Dravyansh Sharma. No internal regret with non-convex loss functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14919–14927, 2024b.

Dravyansh Sharma and Arun Suggala. Offline-to-online hyperparameter transfer for stochastic bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20362–20370, 2025.

Leslie N. Smith. Cyclical learning rates for training neural networks. *arXiv preprint arXiv:1506.01186*, 2015.

Nilesh Tripuraneni, Michael I Jordan, and Pradeep Ravikumar. On theory of transfer learning: Learning representations for new tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Hugh E Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968.

Jingfeng Wu, Peter L Bartlett, Matus Telgarsky, and Bin Yu. Large stepsize gradient descent for logistic loss: Non-monotonicity of the loss improves optimization efficiency. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 5019–5073. PMLR, 2024.

Yixin Wu, James Zou, Mark Braverman, Qihang Gu, and Sham M Kakade. The power and limitation of pretraining-finetuning for linear regression under covariate shift. *arXiv preprint arXiv:2205.12020*, 2022a. URL `https://par.nsf.gov/servlets/purl/10379046`.

Zhaofeng Wu, Nelson F Liu, and Christopher Potts. Identifying the limits of cross-domain knowledge transfer. In *Proceedings of the 7th Workshop on Representation Learning for NLP (RepL4NLP)*, 2022b. URL `https://aclanthology.org/2022.repl4nlp-1.11/`.

Li Yang, Yu Wu, Tongliang Wei, and Dacheng Lin. On the generalization ability of unsupervised pretraining. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. URL `https://pubmed.ncbi.nlm.nih.gov/39420981/`.

Mengdi Yue, Shixiang Wang, Tianyu Li, and Lin Yang. On the lower bound of minimizing polyak-lojasiewicz functions. *arXiv preprint arXiv:2303.00749*, 2023.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *ICLR*, 2019a. URL https://arxiv.org/abs/1901.09321.

Michael R Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

Ruiqi Zhang, Jingfeng Wu, Licong Lin, and Peter L Bartlett. Minimax optimal convergence of gradient descent in logistic regression via large and adaptive stepsizes. *arXiv preprint arXiv:2504.04105*, 2025.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Convergence of adaptive gradient methods for nonconvex optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *arXiv preprint arXiv:2010.07468*, 2020. URL https://arxiv.org/abs/2010.07468.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes, in Section 2 and at the beginning of other sections as applicable.]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, the main focus of this work is to establish sample complexity.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes, assumptions are stated in the proof statements.]
   (b) Complete proofs of all theoretical results. [Yes, either next to the theoretical result or in the appendix.]
   (c) Clear explanations of any assumptions. [Yes, in the main text.]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Gradient Descent with Provably Tuned Learning-rate Schedules: Supplementary Material

A Additional related work

Data-driven algorithm design is a recently introduced paradigm for designing algorithms and provably tuning hyperparameters in machine learning [Gupta and Roughgarden, 2016, Balcan, 2020, Sharma, 2024a]. The framework can be viewed as a generalization of *average case analysis* from uniform distribution over the instances to arbitrary unknown distributions, that is, the tuned hyperparameters adapt to data distribution at hand. Data-driven design has been successfully used for designing several fundamental learning algorithms including regression, low-rank approximation, tree search and many more (see e.g., [Balcan et al., 2017, 2022, Balcan and Sharma, 2021, 2024, Balcan et al., 2024b, 2025b, Blum et al., 2021, Bartlett et al., 2022, Khodak et al., 2024, Cheng and Basu, 2024, Sakaue and Oki, 2024, Jin et al., 2024]). The techniques allow selection of near-optimal continuous hyperparameters, using multiple related tasks which are either drawn from an unknown distribution [Balcan et al., 2017, 2021a] or arrive online [Balcan et al., 2018, 2020, 2021b, Sharma, 2024b, Sharma and Suggala, 2025]. In fact tuning discretized parameters can lead to provably much worse performance than the best continuous parameter Balcan et al. [2024a].

Tuning initialization scale. Glorot and Bengio [2010] is one of the first works to highlight the importance of carefully setting the initialization scale hyperparameter. While their approach works well for neural networks with sigmoid or tanh activations, the initialization proposed by He et al. [2015] is known to work better for ReLU and LeakyReLU activations. This highlights the need to set proper initialization scale, with strong dependence on the network architecture. Several efforts have been made to theoretically understand how the dynamics of the deep neural network are impacted by the initialization scale [Saxe et al., 2013, Poole et al., 2016, Pennington et al., 2017, Jacot et al., 2018, Chizat et al., 2018, Zhang et al., 2019a].

Connections to pre-training. Pre-training to learn foundation models forms the basis for successful language models [Devlin et al., 2018, Radford et al., 2018, Liu et al., 2019, Gururangan et al., 2020, Wu et al., 2022b]. Generalization guarantees and sample complexity are of fundamental interest towards a theoretical understanding of the effectiveness and data requirements of pre-trained models Yang et al. [2020], Tripuraneni et al. [2020]. Of particular interest for these models are properties like adversarial robustness and effectiveness under distribution shift Nern et al. [2022], Wu et al. [2022a], which are relevant future directions for this work.

Beyond gradient descent. Our framework applies to gradient-based optimization beyond gradient descent. Interesting variants include second-order methods [Nocedal and Wright, 2006, Henriques and Vedaldi, 2018], Adam [Kingma and Ba, 2014, Zhuang et al., 2020], Shampoo [Gupta et al., 2018, Lin et al., 2025], and many more [Zhang et al., 2019b, Chen et al., 2021, Pauloski et al., 2021]. In general, these methods are sensitive to hyperparameters which need careful tuning for good convergence and generalization [Li et al., 2020, Barakat et al., 2021]. Understanding their convergence is an active area of research [Reddi et al., 2018, Bock et al., 2018, Zhou et al., 2020, Défossez et al., 2020, Dinh et al., 2020, Liu et al., 2021, Bock and Weiß, 2022].

B GJ algorithm and pseudo-dimension

A useful technique for bounding the pseudo-dimension in data-driven algorithm design is the GJ framework based approach proposed by Bartlett et al. [2022]. We include below the formal details for completeness.

Definition 4 (Goldberg and Jerrum [1993], Bartlett et al. [2022])**.** *A GJ algorithm Γ operates on real-valued inputs, and can perform two types of operations:*

- *Arithmetic operations of the form $v = v_0 \odot v_1$, where $\odot \in \{+, -, \times, \div\}$.*
- *Conditional statements of the form "if $v \geq 0$ ... else ... ".*

In both cases, v_0, v_1 are either inputs or values previously computed by the algorithm (which are rational functions of the inputs). The degree of a GJ algorithm is the maximum degree of any rational function it computes of the inputs. The predicate complexity of a GJ algorithm is the number of distinct rational functions that appear in its conditional statements.

The following theorem due to Bartlett et al. [2022] is useful in obtaining some of our pseudodimension bounds by showing a GJ algorithm that computes the loss for all values of the hyperparameters, on any fixed input instance.

Theorem B.1 (Bartlett et al. [2022])**.** *Suppose that each function $f \in \mathcal{F}$ is specified by n real parameters. Suppose that for every $x \in \mathcal{X}$ and $r \in \mathbb{R}$, there is a GJ algorithm $\Gamma_{x,r}$ that given $f \in \mathcal{F}$, returns "true" if $f(x) \geq r$ and "false" otherwise. Assume that $\Gamma_{x,r}$ has degree Δ and predicate complexity Λ. Then,* $\mathrm{Pdim}(\mathcal{F}) = O(n \log(\Delta\Lambda))$.

C Background on Real Zero Bounds

Understanding the number of real solutions to systems of equations is a classical problem in real algebraic geometry. Two important results in this area are Warren's theorem for polynomials [Warren, 1968] and its analogue for Pfaffian functions due to Khovanskii [Khovanski, 1991].

C.1 Warren's Theorem for Polynomials

Let $f_1, \ldots, f_s : \mathbb{R}^n \to \mathbb{R}$ be real polynomials of degree at most d. Warren's theorem Warren [1968] gives an upper bound on the number of connected components of the set

$$\{x \in \mathbb{R}^n : f_i(x) > 0,\ i = 1, \ldots, s\}.$$

Theorem C.1 (Warren, 1968)**.** *Let $f_1, \ldots, f_s$ be real polynomials in n variables of degree at most d. Then the number of connected components of the set*

$$\{x \in \mathbb{R}^n : f_i(x) > 0 \text{ for all } i = 1, \ldots, s\}$$

is at most

$$\left(\frac{4eds}{n}\right)^n.$$

This result provides a polynomial analogue of the classical Descartes' rule of signs, giving a combinatorial bound on the complexity of semi-algebraic sets.

C.2 Khovanskii's Theorem for Pfaffian Functions

Pfaffian functions generalize polynomials and include many special functions such as exponentials and logarithms. Let $f_1, \ldots, f_s : \mathbb{R}^n \to \mathbb{R}$ be Pfaffian functions of order r and degree at most α. Khovanskii Khovanski [1991] extended Warren's type bounds to these functions.

Lemma C.2 (Khovanski, 1991, page 91)**.** *Let $\mathcal{C}$ be a Pfaffian chain of length q and Pfaffian degree M, consists of functions $f_1, \ldots, f_q$ in $\boldsymbol{a} \in \mathbb{R}^d$. Consider a non-singular system of equations $\Theta_1(\boldsymbol{a}) = \cdots = \Theta_r(\boldsymbol{a}) = 0$ where $r \leq d$, in which $\Theta_i(\boldsymbol{a})$ ($i \in [r]$) is a polynomial of degree at most Δ in the variable $\boldsymbol{a}$ and in the Pfaffian functions $f_1, \ldots, f_q$. Then the manifold of dimension $k = d - r$ determined by this system has at most $2^{q(q-1)}\Delta^d S^{d-r}[(r-d+1)S - (r-d)]^q$ connected components, where $S = r(\Delta - 1) + dM + 1$.*

The following corollary is the direct consequence of Lemma C.2.

Corollary C.3. *Consider the binary-valued function $\Phi(\boldsymbol{x}, \boldsymbol{a})$, for $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{a} \in \mathbb{R}^d$ constructed using the boolean operators AND and OR, and boolean predicates in one of the two forms "$\tau(\boldsymbol{x}, \boldsymbol{a}) > 0$" or "$\tau(\boldsymbol{x}, \boldsymbol{a}) = 0$". Assume that the function $\tau(\boldsymbol{x}, \boldsymbol{a})$ can be one of at most K forms $(\tau_1, \ldots, \tau_K)$, where $\tau_i(\boldsymbol{x}, \cdot)$ ($i \in [K]$) is a C^∞ function of $\boldsymbol{a}$ for any fixed x. Assume that for any fixed $\boldsymbol{x}$, $\tau_i(\boldsymbol{x}, \cdot)$ ($i \in [K]$) is a Pfaffian function of degree at most Δ from a Pfaffian chain $\mathcal{C}$ with length q and Pfaffian degree M. Then*

$$B \leq 2^{dq(dq-1)/2}\Delta^d[(d^2(\Delta + M)]^{dq}.$$

Khovanskii's bound generalizes Warren's result from polynomials to a broad class of functions while keeping an explicit combinatorial control over the number of solutions.

As noted in the main body, Pfaffians include several commonly occurring functions including exponentials and logarithms. On the other hand, non-examples include some trigonometric functions like sine and cosine.

D Complete proofs

Theorem 3.1. *Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of polynomial functions $\mathbb{R}^d \to \mathbb{R}$ of degree at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ using $m = O\left(\frac{H^2}{\epsilon^2}(H \log \Delta + \log \frac{1}{\delta})\right)$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta), g_i^{(2)}(\eta), \ldots, g_i^{(d)}(\eta))$ where $g_i^{(j)}$ is a polynomial function with degree at most Δ^{i-2}. We will show this by induction.

Base case: $i = 2$. $x_2 = x_1 - \eta \nabla f(x_1) = x - \eta \nabla f(x)$ is a polynomial of degree $1 = \Delta^{2-2}$ in η in each coordinate.

Inductive case: $i > 2$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(\eta) = (g_{i-1}^{(j)}(\eta))_{j \in [d]}$ where $g_{i-1}^{(j)}$ is a polynomial of degree at most Δ^{i-3} (inductive hypothesis). Now $x_i = x_{i-1} - \eta \nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(\eta) - \eta \nabla f(\boldsymbol{g}_{i-1}(\eta)) =: \boldsymbol{g}_i(\eta)$. Clearly, $\boldsymbol{g}_i^{(j)}$ is a polynomial in η, with degree at most $\Delta^{i-3}(\Delta - 1) + 1 = \Delta^{i-2} - \Delta^{i-3} + 1 \leq \Delta^{i-2}$.

Thus, for any fixed $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(\eta)$ where each coordinate $g_i^{(j)}$ is a polynomial with degree at most Δ^{i-2}. For a fixed initial point x, this implies that $\nabla f(x_i)$ is a polynomial in η of degree at most Δ^{i-1} in each coordinate. Thus, $\|\nabla f(x_i)\|^2$ is a polynomial of degree at most $2\Delta^{i-1}$ in η. Now, consider the set of points η for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant θ. This consists of at most $O(\Delta^{i-1})$ intervals.

Furthermore, we note that for any η, the cost is determined by the smallest i such that $\|\nabla f(x_i)\| < \theta$ (if one exists). Since the cost takes only discrete values, it is a piecewise-constant function of η, and we seek to bound the number of these pieces. A naive counting argument gives a $O(\Pi_{i=1}^{H} \Delta^{i-1}) = O(\Delta^{H^2})$ bound on the number of pieces, since if there are K_{i-1} intervals corresponds to values of η for which the algorithm converges within $i-1$ steps, then each of the $O(\Delta^{i-1})$ intervals in round i computed above may result in at most $K_{i-1} + 1$ new pieces. We can, however, use an amortized counting argument to give a tighter bound. Indeed, suppose there are K_{i-1} intervals with different values of cost $\leq i-1$. Of the new $O(\Delta^{i-1})$ intervals say T_i intersect at least one of the existing pieces, resulting in at most $T_i + K_{i-1}$ new pieces overall. Thus, the total number of pieces of the cost function with cost $\leq i$ is $K_i \leq (T_i + K_{i-1}) + O(\Delta^{i-1}) - T_i + K_{i-1} \leq O(\Delta^{i-1}) + 2K_{i-1}$. Thus, across i, we have at most $O(\sum_{i=1}^{H} 2^{H-i}\Delta^{i-1}) = O(\Delta^H)$ intervals over each of which Algorithm 1 converges with a constant number of steps. The cost over the remainder of the domain $\mathcal{P}$ where the algorithm does not converge, which consists of $O(\Delta^H)$ intervals, is H.

Thus, using Lemma 2.2, since each function $\ell_{x,f}$ is $O(\Delta^H)$-monotonic, we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H \log \Delta)$, which implies the stated sample complexity. □

Theorem 4.4. *Consider the problem of tuning the η, γ in Algorithm 2 over some continuous set $\mathcal{P} \subset \mathbb{R}^2_{\geq 0}$. Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all $\gamma, \eta \in \mathcal{P}$ using $m = O\left(\frac{H^2}{\epsilon^2}(H \log(pd\Delta) + \log \frac{1}{\delta})\right)$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. We claim that for $i \geq 2$,

$$x_i = (g_i^{(1)}(\eta, \gamma), g_i^{(2)}(\eta, \gamma), \ldots, g_i^{(d)}(\eta, \gamma))$$

and

$$y_i = (h_i^{(1)}(\eta, \gamma), h_i^{(2)}(\eta, \gamma), \ldots, h_i^{(d)}(\eta, \gamma)),$$

where each $g_i^{(j)}$ and $h_i^{(j)}$ is a piecewise-polynomial function with degree at most Δ^{i-2} and at most $(2pd)^{i-2}$ boundaries, each an algebraic curve with degree at most Δ^{i-2}. We will show this by a simultaneous induction argument for x_i and y_i.

Base case: $i = 2$. $y_2 = -\eta \nabla f(x_1)$ and $x_2 = x_1 + y_1 = x_1 - \eta \nabla f(x_1)$ are both polynomials of degree $1 = \Delta^{2-2}$ in η, γ in each coordinate.

Inductive case: $i > 2$. Suppose, by inductive hypothesis, that $x_{i-1} = \boldsymbol{g}_{i-1}(\eta) = (g_{i-1}^{(j)}(\eta))_{j\in[d]}$ and $y_{i-1} = \boldsymbol{h}_{i-1}(\eta) = (h_{i-1}^{(1)}(\eta,\gamma), h_{i-1}^{(2)}(\eta,\gamma), \ldots, h_{i-1}^{(d)}(\eta,\gamma))$, where $g_{i-1}^{(j)}$ and $h_{i-1}^{(j)}$ are piecewise-polynomial with at most $(2pd)^{i-3}$ pieces and degree at most Δ^{i-3} (for both pieces and boundaries).

Now $g_i = \gamma g_{i-1} - \eta \nabla f(x_{i-1}) = \gamma \boldsymbol{h}_{i-1}(\eta,\gamma) - \eta \nabla f(\boldsymbol{g}_{i-1}(\eta,\gamma)) =: \boldsymbol{h}_i(\eta,\gamma)$. Since $\boldsymbol{g}_{i-1}(\eta,\gamma)$ is piecewise-polynomial, $\nabla f(\boldsymbol{g}_{i-1}(\eta,\gamma))$ is also piecewise-polynomial. The degree is at most $(\Delta-1)\Delta^{i-3} \le \Delta^{i-2} - 1$. Any boundary function (algebraic curve along which $\nabla f(\boldsymbol{g}_{i-1}(\eta,\gamma))$ is discontinuous) is either a boundary of $g_{i-1}^{(j)}$ for some j, or a solution of $f^{(k)}(g_{i-1}^{(1)}(\eta,\gamma), \ldots, g_{i-1}^{(d)}(\eta,\gamma)) = 0$ for some boundary function $f^{(k)}$ ($k \in [p]$) of f. The total number of algebraic curves corresponding boundaries of $\boldsymbol{g}_i^{(j)}$ (for any $j \in [d]$) is therefore at most $d(2pd)^{i-3} + (pd)(2pd)^{i-3} \le (2pd)^{i-2}$, with degree at most $\Delta \cdot \Delta^{i-3} = \Delta^{i-2}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in η, γ, with degree at most Δ^{i-2} (for both pieces and boundaries) and at most $(2pd)^{i-2}$ boundaries.

Thus, for any fixed $1 \le i \le H$, $x_i = \boldsymbol{g}_i(\eta,\gamma)$ where each coordinate $g_i^{(j)}$ is piecewise-polynomial with degree at most Δ^{i-2} and at most $(2pd)^{i-2}$ boundaries. Using the argument above, for any fixed initial point x, $\|\nabla f(x_i)\|^2$ is piecewise-polynomial with degree at most $2\Delta^{i-1}$ in η, γ and at most $(2pd)^{i-1}$ boundaries. Therefore, for a fixed θ, the condition $\|\nabla f(x_i)\|^2 < \theta^2$ can be evaluated using a GJ algorithm [Bartlett et al., 2022] of degree $O(\Delta^i)$ and predicate complexity $O((2pd)^i)$. This implies that the dual cost function can be computed using a GJ algorithm with degree at most $O(\Delta^H)$ and predicate complexity $O(\sum_i (2pd)^i) = (2pd)^{O(H)}$. Using Bartlett et al. [2022], we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H \log pd\Delta)$, which implies the stated sample complexity. □

Theorem 4.2. *Consider a variant of Algorithm 1, where the learning rate η is fixed, but the initial point is given by $x = \sigma\hat{x}$ for some fixed $\hat{x}$ and hyperparameter $\sigma \in \mathbb{R}_{>0}$. Suppose the instance space Π consists of $(\hat{x} \in \mathbb{R}^d, f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \ge 1$. Then (ϵ, δ)-uniform convergence is achieved for all σ using $m = O(\frac{H^2}{\epsilon^2}(H \log(pd\Delta) + \log\frac{1}{\delta}))$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. The proof is similar to the proof of Theorem 3.2. Intuitively, we can express the i-th iterate as a polynomial of bounded degree in σ, much like η, and the rest of the argument applies. We include below the details for completeness.

We first claim that for $i \ge 1$, $x_i = (g_i^{(1)}(\sigma), g_i^{(2)}(\sigma), \ldots, g_i^{(d)}(\sigma))$ where each $g_i^{(j)}$ is a piecewise-polynomial function with degree at most Δ^{i-1} and at most $(2pd\Delta)^{i-1}$ pieces. We will show this by induction.

Base case: $i = 1$. $x_1 = \sigma\hat{x}$ is a polynomial of degree $1 = \Delta^{1-1}$ in σ in each coordinate.

Inductive case: $i > 1$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(\sigma) = (g_{i-1}^{(j)}(\sigma))_{j\in[d]}$ where $g_{i-1}^{(j)}$ is piecewise-polynomial with at most $(2pd\Delta)^{i-2}$ pieces and degree at most Δ^{i-2} (inductive hypothesis). Now $x_i = x_{i-1} - \eta\nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(\sigma) - \eta\nabla f(\boldsymbol{g}_{i-1}(\sigma)) =: \boldsymbol{g}_i(\sigma)$. $\nabla f(\boldsymbol{g}_{i-1}(\sigma))$ is piecewise-polynomial, with degree is at most $(\Delta-1)\Delta^{i-2} \le \Delta^{i-1}$. Any critical point is either a critical point of some $g_{i-1}^{(j)}$, or a solution of the equation $f^{(k)}(g_{i-1}^{(1)}(\sigma), \ldots, g_{i-1}^{(d)}(\sigma)) = 0$ for some boundary function $f^{(k)}$ ($k \in [p]$) of f. The total number of critical points of $\boldsymbol{g}_i^{(j)}$ (for any $j \in [d]$) is therefore at most $(2pd\Delta)^{i-2} + (pd\Delta)(2pd\Delta)^{i-2} \le (2pd\Delta)^{i-1}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in σ, with degree at most Δ^{i-2} and at most $(2pd\Delta)^{i-1}$ pieces.

Thus, for any fixed $1 \le i \le H$, $x_i = \boldsymbol{g}_i(\sigma)$ where each coordinate $g_i^{(j)}$ is piecewise-polynomial with degree at most Δ^{i-1} and at most $(2pd\Delta)^{i-1}$ pieces. Using the argument above, for any fixed initial point x, $\|\nabla f(x_i)\|^2$ is piecewise-polynomial with degree at most $2\Delta^i$ in σ and at most $(2pd\Delta)^i$ pieces. Now, consider the set of points σ for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant θ. This consists of at most $O((2pd\Delta)^i)$ intervals.

We can now apply the amortized counting argument from the proof of Theorem 3.1 to conclude that the number of pieces in the dual cost function is $O((2pd\Delta)^H)$ here. Using Lemma 2.2, we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H \log pd\Delta)$, which implies the stated sample complexity. □

Theorem 4.3. *Consider a variant of Algorithm 1, where the learning rate η is fixed, but the initial point is given*

by $x \in \mathbb{R}^d$ (a hyperparameter). Suppose the instance space Π consists of $(f \in \mathcal{F})$, where $\mathcal{F}$ consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$. Then (ϵ, δ)-uniform convergence is achieved for all $x \in \mathbb{R}^d$ using $m = O(\frac{H^2}{\epsilon^2}\left(dH \log(pd\Delta) + \log \frac{1}{\delta}\right))$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.

Proof. Let $x = (x_0^{(1)}, \ldots, x_0^{(d)})$ denote the initial point hyperparameter. We first claim that for each $i \geq 1$, $x_i = (g_i^{(1)}(x_0^{(1)}, \ldots, x_0^{(d)}), \ldots, g_i^{(d)}(x_0^{(1)}, \ldots, x_0^{(d)}))$ where each $g_i^{(j)}$ is a piecewise-polynomial function with degree at most Δ^{i-1} and at most $(2pd)^{i-1}$ algebraic boundaries of degree at most Δ^{i-1}. We establish this by induction.

Base case: $i = 1$. $x_1 = (x_0^{(1)}, \ldots, x_0^{(d)})$ is a polynomial of degree $1 = \Delta^{1-1}$ in $x_0^{(1)}, \ldots, x_0^{(d)}$ in each coordinate.

Inductive case: $i > 1$. Suppose $x_{i-1} = \boldsymbol{g}_{i-1}(x) = (g_{i-1}^{(j)}(x_0^{(1)}, \ldots, x_0^{(d)}))_{j \in [d]}$ where $g_{i-1}^{(j)}$ is piecewise-polynomial with at most $(2pd)^{i-2}$ boundaries and degree at most Δ^{i-2} (for both pieces and boundaries, by the inductive hypothesis).

Now $x_i = x_{i-1} - \eta \nabla f(x_{i-1}) = \boldsymbol{g}_{i-1}(x) - \eta \nabla f(\boldsymbol{g}_{i-1}(x)) =: \boldsymbol{g}_i(x)$. Now, $\nabla f(\boldsymbol{g}_{i-1}(x))$ is piecewise-polynomial. The degree is at most $(\Delta - 1)\Delta^{i-2} \leq \Delta^{i-1}$. Any discontinuity point is either a discontinuity point of some $g_{i-1}^{(j)}$, or a solution of the equation $f^{(k)}(g_{i-1}^{(1)}(x), \ldots, g_{i-1}^{(d)}(x)) = 0$ for some boundary function $f^{(k)}$ $(k \in [p])$ of f. The total number of boundary functions of $\boldsymbol{g}_i^{(j)}$ (for any $j \in [d]$) is therefore at most $d(2pd)^{i-2} + (pd)(2pd)^{i-2} \leq (2pd)^{i-1}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in x, with degree at most Δ^{i-1} and at most $(2pd\Delta)^{i-1}$ pieces.

Thus, for any fixed $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(x)$ where each coordinate $g_i^{(j)}$ is piecewise-polynomial with degree at most Δ^{i-1} and at most $(2pd)^{i-1}$ algebraic boundaries. Using the argument above, for any fixed η, $\|\nabla f(x_i)\|^2$ is piecewise-polynomial with degree at most $2\Delta^i$ in the initial vector x and at most $(2pd\Delta)^i$ boundaries. This implies that the dual cost function can be computed using a GJ algorithm with degree at most $O(\Delta^H)$ and predicate complexity $O(\sum_i (2pd)^i) = (2pd)^{O(H)}$. Using Bartlett et al. [2022], we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(dH \log pd\Delta)$, which implies the stated sample complexity. □

Theorem 5.1. *Suppose the instance space Π consists of $(x \in \mathbb{R}^d, f \in \mathcal{F}, f_v \in \mathcal{F}_v)$, where $\mathcal{F}$ (resp. $\mathcal{F}_v$) consists of functions $\mathbb{R}^d \to \mathbb{R}$ that are piecewise-polynomial with at most p (resp. p_v) polynomial boundaries, with the maximum degree of any piece function or boundary function at most $\Delta \geq 1$ (resp. Δ_v). Then (ϵ, δ)-uniform convergence w.r.t. the validation loss is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ using $m = O\left(\frac{H^2}{\epsilon^2}(H \log(pd\Delta) + \log \frac{\Delta_v p_v}{\delta})\right)$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over Π.*

Proof. As shown in the proof of Theorem 3.2, $\tilde{x} = g(\eta)$ where g is a piecewise polynomial function with degree at most Δ^H and at most $O((2pd\Delta)^H)$ pieces. Therefore, $f_v(\tilde{x}) = f_v(g(\eta))$ is a piecewise polynomial function with degree at most $\Delta_v \Delta^H$ and at most $p_v + (2pd\Delta)^H$ pieces. Using Bartlett et al. [2022], we get a bound on the pseudo-dimension of loss function class of $O(H \log(pd\Delta) + \log(\Delta_v p_v))$, which implies the stated sample complexity. □