Learning how to step in gradient-based optimization: beyond convexity and smoothness

Dravyansh Sharma DRAVY@TTIC.EDU
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 yet one typically relies on heuristic approaches without formal near-optimality guarantees. Recent work by Gupta and Roughgarden 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 theoretical 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 the step-size 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 for neural networks with piecewise-polynomial activation functions (including ReLU activation). Furthermore, we show the versatility of our framework by applying it to tuning momemtum and step-size simultaneously.

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.

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?

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,

© D. Sharma.

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

Output: Return x_{H+1}

- We study tuning of the step-size (learning rate) in gradient descent across tasks in the framework of [28], 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 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).
- We show sample complexity bounds of $\tilde{O}(H^3/\epsilon^2)$ for uniform convergence, which implies generalization of the learned step-size to unseen instances (that is, small gap between training and test errors). Our bounds have a logarithmic dependence on dimensionality d. We further extend our techniques to simultaneously tune the momentum and learning rate.

Related work. Recently tuning the learning rate in gradient descent by meta learning has received significant research interest [1, 26, 31, 33, 34]. 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 [3, 28] (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 [28]. 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 [28] assumes strong sufficient conditions—convexity, L-smoothness, and a guaranteed progress condition (that is $\|x_{i+1} - x^*\| \leq (1-c)\|x_i - x^*\|$ 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,

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

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

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

Theorem 2 ((ϵ, δ)-uniform convergence sample complexity via pseudo-dimension, [2]) *Suppose $\mathcal{F}$ is a class of real-valued functions with range in $[0, H]$ and finite $\operatorname{Pdim}(\mathcal{F})$. 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 $f \in \mathcal{F}$, we have $|\frac{1}{M}\sum_{x\in S} f(x) - \mathbb{E}_{x\sim\mathcal{D}}[f(x)]| \leq \epsilon$ for some $M = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\operatorname{Pdim}(\mathcal{F}) + \log\frac{1}{\delta}\right)\right)$.*

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 3 *(Lemma 2.3, [3]) 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 gradient descent 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 $\ell_{x,f}(\eta)$ [11]) is very different from prior

work. The key idea of prior work [28, 29] 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 [28], 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 for 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 proofs are in Appendix C).

Theorem 4 *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.*

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 5 *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.*

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

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}

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 [22]. 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 5 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 [22]). 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.*

Finally, we note that our techniques can be used to establish similar sample complexity bounds for tuning the learning rate schedule (see Theorem 9 in the Appendix).

4. 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 show how to tune the momemtum and learning rate parameters γ, η simultaneously in Algorithm 2. Momentum [35] 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. We extend our approach to show how to tune the momentum parameter and the learning rate in momentum-based gradient descent.

Theorem 6 *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). In conclusion, our work constitutes an important step forward towards the development of theoretically principled techniques for step-size selection in more realistic non-convex and non-smooth settings.

References

[1] 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.

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

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

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

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

[6] 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.

[7] 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.

[8] Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized Lloyd's families. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[9] Maria-Florina Balcan, Travis Dick, and Manuel Lang. Learning to link. In *International Conference on Learning Representation (ICLR)*, 2020.

[10] 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.

[11] 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, 2021.

[12] 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, 2021.

[13] Maria-Florina Balcan, Misha 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.

[14] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Structural analysis of branch-and-cut and the learnability of Gomory mixed integer cuts. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:33890–33903, 2022.

[15] Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. An analysis of robustness of non-Lipschitz networks. *Journal of Machine Learning Research*, 24(98):1–43, 2023.

[16] Maria-Florina Balcan, Anh Nguyen, and Dravyansh Sharma. New bounds for hyperparameter tuning of regression problems across instances. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 80066–80078. Curran Associates, Inc., 2023.

[17] 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)*, 2024.

[18] 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, 2024.

[19] Maria-Florina Balcan, Saumya Goyal, and Dravyansh Sharma. Distribution-dependent generalization bounds for tuning linear regression across tasks. *arXiv preprint arXiv:2507.05084*, 2025.

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

[21] 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. *arXiv preprint arXiv:2501.13734*, 2025.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

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

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

[36] 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.

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

[38] Dravyansh Sharma and Maxwell Jones. Efficiently learning the graph for semi-supervised learning. In *Uncertainty in Artificial Intelligence*, pages 1900–1910. PMLR, 2023.

[39] 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.

Appendix A. Additional related work

Data-driven algorithm design is a recently introduced paradigm for designing algorithms and provably tuning hyperparameters in machine learning [3, 28]. 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., [4–6, 8, 9, 13–16, 18–21, 23–25, 30, 32, 36]). The techniques allow selection of near-optimal continuous hyperparameters, using multiple related tasks which are either drawn from an unknown distribution [6, 11] or arrive online [7, 10, 12, 37–39]. In fact tuning discretized parameters can lead to provably much worse performance than the best continuous parameter [17]. Recent work develops techniques for data-driven tuning of model hyperparameters in deep nets [21], but their techniques do not apply to parameters of training algorithms including learning rate.

Appendix 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 [23]. We include below the formal details for completeness.

Definition 7 ([23, 27]) *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 $v \geq 0 \ldots$ else $\ldots$".*

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 [23] 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 8 ([23]) *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, $\mathrm{Pdim}(\mathcal{F}) = O(n \log(\Delta\Lambda))$.*

Appendix C. Complete proofs

Theorem 4 *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 determine 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 3, 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 5 *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 4 to conclude that the number of pieces in the dual cost function is $O((2pd\Delta)^H)$ here.

Thus, using Lemma 3, we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H \log pd\Delta)$, which implies the stated sample complexity. ■

Theorem 6 *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 $x_{i-1} = \boldsymbol{g}_{i-1}(\eta) = (g_{i-1}^{(j)}(\eta))_{j \in [d]}$ 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, by inductive hypothesis).

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 $(\Delta - 1)\Delta^{i-3} \leq \Delta^{i-2} - 1$. 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 $d(2pd)^{i-3} + (pd)(2pd)^{i-3} \leq (2pd)^{i-2}$,

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 $1 \leq i \leq H$, $x_i = \boldsymbol{g}_i(\eta, \gamma)$ 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 [23] 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 [23], we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H \log pd\Delta)$, which implies the stated sample complexity. ■

Appendix D. 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 9 *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 [23], we get a bound on the pseudo-dimension of $\mathcal{L}$ of $O(H^2 \log pd\Delta)$, which implies the stated sample complexity. ■