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

**author names withheld**

**Under Review for the Workshop on High-dimensional Learning Dynamics, 2025**

## 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. 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 getting stuck in local minima.

While considerable research has focused on tuning step size for individual tasks, many real-world applications involve multi-task learning or repeated optimization across a variety of tasks. In such settings, the optimal step size may vary significantly between tasks, 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, potentially diverse 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,

**Algorithm 1** Gradient descent(step size $\eta$)

```
Input: Initial point $x$, function to minimize $f$, maximum number of iterations $H$, gradient threshold for convergence $\theta$
1: Initialize $x_1 \leftarrow x$
2: for $i = 1, \ldots, H$ do
3:     if $||\nabla f(x_i)|| < \theta$ then
4:         Return $x_i$
5:     $x_{i+1} = x_i - \eta \nabla f(x_i)$
Output: Return $x_H$
```

- We study tuning of the step-size (learning rate) in gradient descent across tasks in the framework of [12], 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. Learning-to-learn or meta learning has received widespread research attention in the recent years [1, 11, 16–18]. [14] provide a detailed overview of various meta-learning based approaches for neural networks. [13] show usefulness beyond machine learning and provide a unified framework for studying meta-learning for well-known game classes, including two-player zero-sum, general-sum, and Stackelberg games.

*Data-driven algorithm design* is a recently introduced paradigm for designing algorithms and provably tuning hyperparameters in machine learning [3, 12]. 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 simulated annealing, low-rank approximation and tree search (see e.g., [4, 9, 10]). The techniques allow selection of near-optimal continuous hyperparameters, and in fact tuning discretized parameters can lead to provably much worse performance than the best continuous parameter [6]. Recent work develops techniques for data-driven tuning of model hyperparameters in deep nets [7], but their techniques do not apply to parameters of training algorithms including learning rate.

## 2. Notation and preliminaries

We recall the notation and setup for data-driven tuning of gradient step introduced by [12]. The instance space of problems $\Pi$ here consists of pairs $(x, f)$ consisting of points $x \in \mathbb{R}^d$ and functions $f : \mathbb{R}^d \to \mathbb{R}$. The family of algorithms $\mathcal{A}$ is given by $\mathcal{P} \subset \mathbb{R}_+$ consisting of valid values of the step size $\eta$. We recall the vanilla gradient descent algorithm (Algorithm 1). We define the cost function as $\ell(\eta, x, f)$ for $\eta \in \mathcal{P}$ and $(x, f) \in \Pi$ is the number of iterations for which Algorithm 1 runs on the instance $(x, f)$ when run with step size $\eta$. Let $\ell_\eta(x, f) := \ell(\eta, x, f)$ for all $\eta, x, f$. For guarantees on sample complexity of tuning $\eta$, 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 [12] assumes strong sufficient conditions—convexity, $L$-smoothness, guaranteed progress (that is $\|x_{i+1} - x^*\| \leq (1-c)\|x_i - x^*\|$ for some $c > 0$ where $x^*$ is the unique stationary point, this roughly corresponds to strong-convexity)—for the convergence to always happen and includes appropriate restrictions on the step-size $\eta$, our results hold when all of these conditions are violated. We handle non-convergence by assigning it the same cost as maximum number of iterations, equal to $H$. Formally,

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

We consider the step-size as the hyperparameter of interest for the algorithm, and assume other parameters like maximum number of iterations $H$ and gradient threshold for convergence $\theta$ are fixed and known. We will use the $\tilde{O}$ notation to suppress dependence on quantities apart from $H$ and the generalization error $\epsilon$ 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 ($(\epsilon, \delta)$-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 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 $\eta$ by classical learning theory), is by examining the structure of $\ell(\eta, x, f)$ on any fixed instance $(x, f)$ as the step-size $\eta$ is varied (also called the *dual* cost function $\ell_{x,f}(\eta)$ [5]) is very different from prior work. The key idea of prior work [12, 15] 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 rate of convergence of the gradient descent algorithm as the step-size $\eta$ 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 [12], for which they obtain a $\tilde{O}(H^3/\epsilon^2)$ bound on the sample complexity of tuning the step-size $\eta$ 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 $\Delta$. Our sample complexity bound only has a logarithmic dependence on $\Delta$, 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 $\eta$ using an inductive argument. This allows us to bound the number for intervals of $\eta$ for which Algorithm 1 converges in $i$ steps for any $1 \le i \le H$. 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 A).

**Theorem 4** *Suppose the instance space $\Pi$ 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 $\Delta \ge 1$. Then $(\epsilon, \delta)$-uniform convergence is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\ge 0}$ using $m = O(\frac{H^2}{\epsilon^2}(H \log \Delta + \log \frac{1}{\delta}))$ samples from $\mathcal{D}$ for any distribution $\mathcal{D}$ over $\Pi$.*

The uniform convergence guarantee in the above result is quite strong. It implies that we only need to find an approximately optimal $\eta$ 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 $\Pi$ 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 is at most $\Delta \ge 1$. Then $(\epsilon, \delta)$-uniform convergence is achieved for all step-sizes $\eta \in \mathcal{P} \subset \mathbb{R}_{\ge 0}$ 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 $\Pi$.*

**Algorithm 2** Momentum-based gradient descent(step size $\eta$, momentum parameter $\gamma$)

```
Input: Initial point $x$, function to minimize $f$, maximum number of iterations $H$, gradient threshold for convergence $\theta$
1: Initialize $x_1 \leftarrow x, y_1 \leftarrow 0$
2: for $i = 1, \ldots, H$ do
3:     if $||\nabla f(x_i)|| < \theta$ then
4:         Return $x_i$
5:     $y_{i+1} = \gamma y_i - \eta \nabla f(x_i)$
6:     $x_{i+1} = x_i + y_i$
Output: Return $x_H$
```

The case of piecewise-polynomial functions is particularly interesting. As discussed below 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 $x$ [8]. 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 [8]). 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.*

## 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 $\gamma, \eta$ simultaneously in Algorithm 2. Momentum [19] 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 $\eta, \gamma$ in Algorithm 2 over some continuous set $\mathcal{P} \subset \mathbb{R}^2_{\geq 0}$. Suppose the instance space $\Pi$ 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 is at most $\Delta \geq 1$. Then $(\epsilon, \delta)$-uniform convergence is achieved for all $\gamma, \eta \in \mathcal{P}$ 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 $\Pi$.*

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 $L$ ($\Delta$ 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. Learning accurate and interpretable decision trees. In *Uncertainty in Artificial Intelligence (UAI)*, pages 288–307. PMLR, 2024.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Keegan Harris, Ioannis Anagnostides, Gabriele Farina, Mikhail Khodak, Zhiwei Steven Wu, and Tuomas Sandholm. Meta-learning in games. *International Conference on Learning Representations (ICLR)*, 2023.

[14] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44:5149–5169, 2020.

[15] 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.

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

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

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

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

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

## Appendix A. Complete proofs

**Theorem 4** *Suppose the instance space* $\Pi$ *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* $(\epsilon, \delta)$*-uniform convergence is achieved for all step-sizes* $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ *using* $m = O(\frac{H^2}{\epsilon^2}(H \log \Delta + \log \frac{1}{\delta}))$ *samples from* $\mathcal{D}$ *for any distribution* $\mathcal{D}$ *over* $\Pi$.

**Proof** We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta), g_i^{(2)}(\eta), \dots, g_i^{(d)}(\eta))$ where $g_i^{(j)}$ is a polynomial function with degree at most $\Delta^{i-2}$. 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 $\eta$ 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 $\Delta^{i-3}$ (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 $\eta$, with degree at most $\Delta^{i-3}(\Delta - 1) + 1 = \Delta^{i-2} - \Delta + 1 \leq \Delta^{i-2}$.

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 $\Delta^{i-2}$. For a fixed initial point $x$, this implies that $\nabla f(x_i)$ is a polynomial in $\eta$ of degree at most $\Delta^{i-1}$ in each coordinate. Thus, $\|\nabla f(x_i)\|^2$ is a polynomial of degree at most $2\Delta^{i-1}$ in $\eta$. Now, consider the set of points $\eta$ for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant $\theta$. This consists of at most $O(\Delta^{i-1})$ intervals.

Furthermore, we note that for any $\eta$, 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 $\eta$, 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 $\eta$ 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* $\Pi$ *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 is at most* $\Delta \geq 1$*. Then* $(\epsilon, \delta)$*-uniform convergence is achieved for all step-sizes* $\eta \in \mathcal{P} \subset \mathbb{R}_{\geq 0}$ *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* $\Pi$.

**Proof** We first claim that for $i \geq 2$, $x_i = (g_i^{(1)}(\eta), g_i^{(2)}(\eta), \dots, g_i^{(d)}(\eta))$ where each $g_i^{(j)}$ is a piecewise-polynomial function with degree at most $\Delta^{i-2}$ 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 $\eta$ 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 $\Delta^{i-3}$ (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 $(\Delta-1)\Delta^{i-3} \le \Delta^{i-2} - 1$. 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 $(2pd\Delta)^{i-3} + (pd\Delta)(2pd\Delta)^{i-3} \le (2pd\Delta)^{i-2}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in $\eta$, with degree at most $\Delta^{i-2}$ and at most $(2pd\Delta)^{i-2}$ pieces.

Thus, for any fixed $1 \le i \le H$, $x_i = \boldsymbol{g}_i(\eta)$ where each coordinate $g_i^{(j)}$ is piecewise-polynomial with degree at most $\Delta^{i-2}$ 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 $\eta$ and at most $(2pd\Delta)^{i-1}$ pieces. Now, consider the set of points $\eta$ for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant $\theta$. 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 $\eta, \gamma$ in Algorithm 2 over some continuous set $\mathcal{P} \subset \mathbb{R}^2_{\ge 0}$. Suppose the instance space $\Pi$ 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 is at most $\Delta \ge 1$. Then $(\epsilon, \delta)$-uniform convergence is achieved for all $\gamma, \eta \in \mathcal{P}$ 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 $\Pi$.*

**Proof** We claim that for $i \ge 2$,

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

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 $\Delta^{i-2}$ and at most $(2pd)^{i-2}$ boundaries, each an algebraic curve with degree at most $\Delta^{i-2}$. 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 $\eta, \gamma$ 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 $\Delta^{i-3}$ (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 $(2pd)^{i-3} + (pd)(2pd)^{i-3} \leq (2pd)^{i-2}$, with degree at most $\Delta \cdot \Delta^{i-3} = \Delta^{i-2}$. Therefore, $\boldsymbol{g}_i^{(j)}$ is piecewise-polynomial in $\eta, \gamma$, with degree at most $\Delta^{i-2}$ (for both pieces and boundaries) and at most $(2pd\Delta)^{i-2}$ 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 $\Delta^{i-2}$ 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 $\eta, \gamma$ and at most $(2pd)^{i-1}$ boundaries. By Warren's theorem [20], these boundaries induce at most $O\left((2\Delta^{i-1} \cdot (2pd)^{i-1})^2\right) = O\left((2pd\Delta)^{2(i-1)}\right)$ connected components. Now, consider the set of points $\eta, \gamma$ for which $\|\nabla f(x_i)\|^2 < \theta^2$ for some constant $\theta$. This consists of at most $O((2pd\Delta)^{2(i-1)} \cdot (2\Delta^{i-1})^2) = O\left((2pd\Delta)^{4(i-1)}\right)$ connected components.

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 $O((2pd\Delta)^{4H})$ 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. ■