Learning Complexity of Gradient Descent and Conjugate Gradient Algorithms

Xianqi Jiao, Jia Liu*, Zhiping Chen*

School of Mathematics and Statistics, Xi'an Jiaotong University, Shaanxi, China
jxq00913@stu.xjtu.edu.cn, {jialiu, zchen}@mail.xjtu.edu.cn

Abstract

Gradient Descent (GD) and Conjugate Gradient (CG) methods are among the most effective iterative algorithms for solving unconstrained optimization problems, particularly in machine learning and statistical modeling, where they are employed to minimize cost functions. In these algorithms, tunable parameters, such as step sizes or conjugate parameters, play a crucial role in determining key performance metrics, like runtime and solution quality. In this work, we introduce a framework that models algorithm selection as a statistical learning problem, and thus learning complexity can be estimated by the pseudo-dimension of the algorithm group. We first propose a new cost measure for unconstrained optimization algorithms, inspired by the concept of primal-dual integral in mixed-integer linear programming. Based on the new cost measure, we derive an improved upper bound for the pseudo-dimension of gradient descent algorithm group by discretizing the set of step size configurations. Moreover, we generalize our findings from gradient descent algorithm to the conjugate gradient algorithm group for the first time, and prove the existence a learning algorithm capable of probabilistically identifying the optimal algorithm with a sufficiently large sample size.

Extended version — https://arxiv.org/abs/2412.13473

Introduction

Gradient descent (GD) and conjugate gradient (CG) methods are widely used, iterative algorithms for effectively solving unconstrained optimization problems, especially in machine learning and statistical modeling to minimize cost functions. Typically in GD (Snyman 2005), each iteration involves calculating the gradient at the current iteration point, multiplying it by a scaling factor (also known as step size), and subtracting it from the current iteration point. With certain assumptions on the problem's convexity, GD converges to a global minimum. Similarly, in CG (Hestenes and Stiefel 1952), at each iteration step, the CG algorithm calculates the gradient at the current iteration point, determines the new search direction by combining the gradient of the current step with the previous direction multiplied by the conjugate parameter, and then adds the product of the search direction and the step size to the current iteration point. There are several well-known formulas for the conjugate parameter, including those proposed by Hestenes and Stiefel (1952), Fletcher and Reeves (1964), Polyak (1969), and Dai and Yuan (1999) . While different conjugate parameters yield varying convergence properties, it is well established that for an n-dimensional problem, the CG method theoretically converges in at most n iterations for quadratic programs. In recent years, theoretical research on the CG method continues (Du, Zhang, and Ma 2016; Liu, Liu, and Dai 2020; Stanimirovi et al. 2020; Babaie-Kafaki 2023; Andrei 2020), but most studies focus exclusively on specific iterative methods and their convergence analyses.

Both GD and CG methods are widely utilized in various machine learning and optimization tasks, particularly in the training of models such as neural networks. Understanding the learning complexity of these algorithms is essential for evaluating their scalability in the face of increasing data sizes or model parameters. Furthermore, complexity analysis is closely related to the generalization properties of a learning algorithm, providing insights into the maximum number of samples required to effectively learn the optimal parameters (e.g., step size) within the algorithm. By estimating the error of a particular learning model alongside the learning complexity of the GD/CG methods employed in it, we can establish a robust theoretical guarantee regarding the model's performance. Optimization algorithms problems often have tunable parameters that significantly impact performance metrics such as runtime, solution quality, and memory usage, particularly in models incorporating machine learning techniques. *However, in most machine learning models, parameter tuning rarely comes with provable guarantees*. Therefore, *Selecting an appropriate step size or conjugate parameter is crucial for both GD and CG methods as they significantly affect convergence speed and generalization performance.* However, determining the appropriate step size and conjugate parameter for a specific type of problem remains an open question. Researchers have approached this challenge by treating the tuning of these parameters as a meta-optimization problem, where the goal is to identify the optimal parameters for a meta-optimizer. Maclaurin, Duvenaud, and Adams (2015) initially proposed a learning-to-learn (L2L) approach, fol-

*Corresponding authors.

lowed by Andrychowicz et al. (2016), who trained neural networks to predict the step size in the GD method. This learning-to-learn approach, also known as meta-learning (Li and Malik 2016) or reversible learning (Maclaurin, Duvenaud, and Adams 2015), has gained considerable attention in recent years. Building on this framework, Denevi et al. (2019) introduced a novel meta-algorithm designed to estimate the bias term online from a family of tasks, where the single task is to apply stochastic gradient descent (SGD) on the empirical risk, regularized by the squared Euclidean distance from a bias vector. Harris et al. (2023) further extended meta-learning by providing guarantees for various well-known game classes, including two-player zero-sum, general-sum, and Stackelberg games, and also proposed results for single-game scenarios within a unified framework. For a comprehensive overview of meta-learning models, we refer readers to a recent survey (Hospedales et al. 2022).

Recently, the machine learning community has shown a growing interest in theoretical guarantees. Wang et al. (2021) highlighted that meta-optimization is particularly challenging due to issues like exploding or vanishing meta-gradients and poor generalization if the meta-objective is not carefully selected. They provided meta-optimization guarantees for tuning the step size in quadratic loss functions, demonstrating that a naive objective can suffer from meta-gradient issues and that a carefully designed meta-objective can maintain polynomially bounded meta-gradients. Yang (2022) introduced a stable adaptive stochastic conjugate gradient (SCG) algorithm tailored for the mini-batch setting. This algorithm integrates the stochastic recursive gradient approach with second-order information within a CG-type framework, allowing for the estimation of the step size sequence without requiring Hessian information. This approach achieves the low computational cost typical of first-order algorithms. Additionally, they established a linear convergence rate for SCG algorithms on strongly convex loss functions.

We would like to highlight a learning-theoretic framework for data-driven algorithm design introduced by Gupta and Roughgarden (2016), which established sample complexity bounds for tuning the step size in gradient descent. It comprises four key components: a fixed optimization problem, an unknown instance distribution, a family of algorithms, and a cost function that evaluates the performance of a given algorithm on a specific instance. Building on Guptas work, data-driven algorithms have been applied across a wide range of optimization problems. Balcan et al. (2017) utilized this framework for configuration problems in combinatorial partitioning and clustering. They further extended this approach to other optimization challenges, including tree search algorithms such as the branch-and-bound method (Balcan et al. 2018) and the branch-and-cut method (Balcan et al. 2022), as well as for non-convex piecewise-Lipschitz functions (Maria-Florina et al. 2021). Furthermore, Alabi et al. (2019) developed an algorithm for pruning problems that learns to optimally prune the search space in repeated computations.

In the vein of research on sample complexity for algorithms, *the choice of cost function is equally critical in assessing an algorithms performance*. Common options include running time and the objective function value of the solution produced by the algorithm (Gupta and Roughgarden 2016; Balcan et al. 2021). In Gupta and Roughgarden (2016), for GD problems, the cost function is defined as the number of iterations (i.e., steps) the algorithm takes to solve a given optimization task. For clustering methods (Balcan et al. 2017), the cost function is set as the k-means or k-median objective value, the distance to a ground-truth clustering, or the expected value of the solution. In the branch-and-bound method (Balcan et al. 2022), the cost function is set as the size of the search tree under a given node selection policy. In a word, selecting an appropriate cost function can significantly enhance performance and may also yield stronger theoretical guarantees.

To the best of our knowledge, when learning the parameter in GD, the cost function is typically defined as *the number of iterations*. For instance, Gupta and Roughgarden (2016) show the existence of a learning algorithm that $(1+\epsilon, \delta)$-learns the optimal algorithm in an algorithm set $\mathcal{A}$ using $m = \tilde{O}(H^3/\epsilon^2)$ samples, here H is a given constant. *However, extending this result to the CG method meets significant challenges.* Moreover, when the optimization problem is too large or computational resources are limited, calculating the iteration number based cost function additionally becomes impractical. Furthermore, since the number of iterations must be an integer, the value $1+\epsilon$ cannot be further reduced. *To address these issues, we introduce a new cost function that sums the distances between the current and optimal values at each iteration.* We demonstrate that, using this cost function, there exists a learning algorithm that $(C+\epsilon, \delta)$-learns the optimal algorithm in the GD algorithm set $\mathcal{A}$ using $m = \tilde{O}(H^3/\epsilon^2)$ samples for any $C > 0$. We further extend the framework to the CG algorithm, and establish that there exists a learning algorithm that $(C+\epsilon, \delta)$-learns the optimal algorithm in the CG algorithm set $\mathcal{A}$ using $m = \tilde{O}(H^4/\epsilon^2)$ samples for any $C > 0$.

To conclude, the major contributions of this paper includes:

1. We propose a new cost function to more effectively measure the performance of GD and CG algorithms. In detail, the function calculates the sum of distances between the current and optimal values at each iteration.
2. We improve the learning complexity of GD under the new cost function. In detail, using the new cost function, we demonstrate that there exists a learning algorithm that $(C+\epsilon, \delta)$-learns the optimal algorithm within algorithm set $\mathcal{A}$ using $m = \tilde{O}(H^3/\epsilon^2)$ samples for any $C > 0$.
3. To the best of our knowledge, we firstly establish the learning complexity result for the CG algorithm. In detail, we prove that there exists a learning algorithm that $(C+\epsilon, \delta)$-learns the optimal algorithm within the CG algorithm set $\mathcal{A}$ using $m = \tilde{O}(H^4/\epsilon^2)$ samples for any $C > 0$.

The rest of this paper is organized as follows: Section 2 reviews the learning framework and establishes generalization guarantees by analyzing the pseudo-dimension of

the algorithm classes. Section 3 summarizes some results from Gupta and Roughgarden (2016), introduces our new cost function, and provides the corresponding generalization guarantees for the gradient descent method. Section 4 extends these generalization guarantees to the conjugate gradient method under specific assumptions. Finally, Section 5 concludes the paper.

All proofs and some introductory background can be found in the extended version (Jiao, Liu, and Chen 2024).

Background of Learning Complexity

In the algorithm selection model, the primary approach involves choosing the best algorithm for a poorly understood application domain by learning the optimal algorithm based on an unknown instance distribution. We adopt the learning complexity framework for data-driven algorithm design introduced by Gupta and Roughgarden (2016) to analyze the sample complexity bounds for tuning the step sizes of the gradient descent and conjugate gradient algorithms. We briefly review the learning-theoretic framework, which consists of the following four components:

- A fixed computational or optimization problem Π,
- An unknown distribution $\mathcal{D}$ over instances $x \in \Pi$,
- A set $\mathcal{A}$ of algorithms for solving Π,
- A cost function c: $\mathcal{A} \times \Pi \to [0, H]$, measuring the performance of a given algorithm on a given instance.

The choice of cost function is pivotal in determining an algorithms performance. Various cost functions apply different criteria to identify the optimal algorithm.

The primary goal of algorithm selection/learning is to identify an algorithm $A_{\mathcal{D}}$ that minimizes $\mathbf{E}_{x\sim\mathcal{D}}[c(A, x)]$ over $A \in \mathcal{A}$. Learning complexity theory examines the efficiency of the learned algorithm. A commonly used measurement is the concept of (ϵ, δ)-learning with probability.

Definition 1 ((ϵ, δ)-learning with probability (Kearns and Vazirani 1994))**.** *A learning algorithm L (ϵ, δ)-learns the optimal algorithm in $\mathcal{A}$ from m samples if, for every distribution $\mathcal{D}$ over Π, with probability at least $1 - \delta$ over m samples $x_1, x_2, \ldots, x_m \sim \mathcal{D}$, L outputs an algorithm $\hat{A} \in \mathcal{A}$ with error at most ϵ.*

Learning complexity essentially measures the performance of an algorithm in out-of-sample scenarios, often referred to as generalization guarantees. It can be estimated using the pseudo-dimension approach applied to the algorithm classes.

Definition 2 (Pseudo-dimension of the algorithm class (Pollard 1984))**.** *Let $\mathcal{H}$ denote a set of real-valued functions defined on the set X. A finite subset $S = \{x_1, x_2, \ldots, x_m\}$ of X is shattered by $\mathcal{H}$ if there exist real-valued witnesses $r_1, r_2, \ldots, r_m$ such that, for each of the 2^m subsets T of S, there exists a function $h \in \mathcal{H}$, such that $h(x_i) > r_i$ if and only if $i \in T$ (for $i = 1, 2, \ldots, m$). The pseudo-dimension of $\mathcal{H}$ is the cardinality of the largest subset shattered by $\mathcal{H}$.*

It is worth noting that the pseudo-dimension of a finite set A is always at most $\log_2 |A|$. To accurately estimate the expectation of all functions in $\mathcal{H}$ with respect to any probability distribution $\mathcal{D}$ on $\mathcal{X}$, it is sufficient to bound the pseudo-dimension of $\mathcal{H}$.

Theorem 1 (Uniform convergence (Anthony and Bartlett 1999))**.** *Let $\mathcal{H}$ be a class of functions with domain X and range in $[0, H]$, and suppose $\mathcal{H}$ has pseudo-dimension $d_{\mathcal{H}}$. For every distribution $\mathcal{D}$ over X, every $\epsilon > 0$, and every $\delta \in (0, 1]$, if*

$$m \geq k(\frac{H}{\epsilon})^2(d_{\mathcal{H}} + \ln(\frac{1}{\delta}))$$

for a suitable constant k, then with probability at least $1 - \delta$ over m samples $x_1, x_2, \ldots, x_m \sim \mathcal{D}$,

$$\left|(\frac{1}{m}\sum_{i=1}^{m} h(x_i)) - \mathbf{E}_{x\sim\mathcal{D}}[h(x)]\right| < \epsilon,$$

for every $h \in \mathcal{H}$.

For the class $\mathcal{A}$, considered as a set of real-valued functions defined on Π, we analyze its pseudo-dimension as defined above. Finally, we provide one additional definition.

Definition 3 (Empirical risk minimization (ERM) (Vapnik 1991))**.** *Fix an optimization problem Π, a performance measure c, and a set of algorithms $\mathcal{A}$. An algorithm L is an empirical risk minimization (ERM) algorithm if, given any finite subset S of Π, L returns an algorithm from $\mathcal{A}$ with the best average performance on S.*

The concept of empirical risk minimization can be employed to ensure the existence of an algorithm that can be (ϵ, δ)-learned with probability. Specifically, Gupta and Roughgarden (2016) demonstrated that in any algorithm set with pseudo-dimension d, one can $(2\epsilon, \delta)$-learn the optimal algorithm with high probability, given a sufficiently large number of samples.

Corollary 1. (Gupta and Roughgarden 2016) Fixing parameters $\epsilon > 0$, $\delta \in (0, 1]$, a set of problem instances Π, and a cost function c. Let $\mathcal{A}$ be a set of algorithms that has pseudo-dimension d with respect to Π and c. Then any ERM algorithm $(2\epsilon, \delta)$-learns the optimal algorithm in $\mathcal{A}$ from m samples, where m is defined as in Theorem 1.

Learning Complexity of Gradient Descent Algorithm

Learning Complexity on Number of Iterations for Gradient Descent Algorithm

We apply the aforementioned learning complexity theory to the problem of learning step sizes from samples in an algorithm selection problem for unconstrained optimization. Initially, we restrict the algorithm set to gradient descent algorithms with varying step sizes.

Recall the basic gradient descent algorithm for minimizing a function f given an initial point z_0 over $\mathbb{R}^N$:

1. Initialize $z := z_0$,
2. While $||\nabla f(z)||_2 > \nu$, $z := z - \rho\nabla f(z)$.

where the step size ρ is the parameter of interest. For example, larger values of ρ can make bigger progress per step but also carry the risk of overshooting a minimum of f.

We make the following assumptions on the gradient descent algorithm for unconstrained optimization problems (Gupta and Roughgarden 2016). The explanation and motivation regarding Assumption 1.4 here and Assumption 2.4 in the next section, are detailed in (Jiao, Liu, and Chen 2024).

Assumption 1. *1) f is convex and L-smooth, i.e., for any $z_1, z_2 \in \mathbb{R}^N$, $||\nabla f(z_1) - \nabla f(z_2)|| \leq L||z_1 - z_2||$,*
2) The initial points are bounded with $\nu < ||z_0|| \leq Z$,
3) The step size ρ is drawn from some interval: $\rho \in [\rho_l, \rho_u] \subset (0, \infty)$,
4) There exists a constant $\beta \in (0, 1)$ such that $||z - \rho\nabla f(z)|| \leq (1 - \beta)||z||$ for all $\rho \in [\rho_l, \rho_u]$.

We define that $g(z, \rho) := z - \rho\nabla f(z)$ as the one-step iteration starting at z with step size ρ. $g^j(z, \rho)$ represents the result after j iterations. Based on the above assumptions, we have

$$||g^j(z, \rho)|| \leq (1 - \beta)^j||z||$$

and

$$||\nabla f(g^j(z, \rho))|| \leq L(1 - \beta)^j||z||.$$

We first review some theoretical results derived by Gupta and Roughgarden (2016), where a cost measure $c(A_\rho, x)$ is defined as the number of iterations required by the algorithm for the instance x. Since $||z_0|| \leq Z$ and the algorithm stops once the gradient is less than ν, it follows that $c(A_\rho, x) \leq \log(\nu/LZ)/\log(1 - \beta)$ for all ρ and x. Let $H = \log(\nu/LZ)/\log(1 - \beta)$. We then estimate the error bound of the cost function with respect to different step sizes of a gradient decent algorithm:

Theorem 2 (Error estimation of cost function (Gupta and Roughgarden 2016))**.** *For any instance x, step sizes $\rho \in [\rho_l, \rho_u]$ and $\eta \in [\rho_l, \rho_u]$ with $0 \leq \eta - \rho \leq \frac{\nu\beta^2}{LZ} D(\rho)^{-H}$, where $D(\rho) := \max\{1, L\rho - 1\}$, we have $|c(A_\rho, x) - c(A_\eta, x)| \leq 1$.*

The following lemma provides an estimate of the error bound for iteration processes using different step sizes, which will be used in the proofs in the next section.

Lemma 1 (Error estimation of iteration processes (Gupta and Roughgarden 2016))**.** *For any $z \in \mathbb{R}^N$, $j > 0$, and step sizes $\rho \leq \eta$,*

$$||g^j(z, \rho) - g^j(z, \eta)|| \leq (\eta - \rho)\frac{D(\rho)^j LZ}{\beta}.$$

Moreover, let $K = \frac{\nu\beta^2}{LZ} D(\rho_u)^{-H}$. We choose N as a minimal K-net, consisting of all integer multiples of K that lie in $[\rho_l, \rho_u]$. Then $|N| \leq \rho_u/K + 1$. It is known that the pseudo-dimension of $\mathcal{A}_N = \{A_\rho, \rho \in N\}$ is at most $\log|N|$. Since $\mathcal{A}_N$ is finite and $\log|N| \sim \tilde{O}(H)$, it also admits an ERM algorithm L_N, which (ϵ, δ)-learns the optimal algorithm in $\mathcal{A}_N$ using $m = \tilde{O}(H^2 \log|N|/\epsilon^2)$ samples, where $\tilde{O}(\cdot)$ suppresses logarithmic factors in Z/ν, β, L and ρ_u. The following theorem presents the result.

Theorem 3 (Generalization Guarantees for GD on number of iterations (Gupta and Roughgarden 2016))**.** *There is a learning algorithm that $(1 + \epsilon, \delta)$-learns the optimal algorithm in $\mathcal{A}$ using $m = \tilde{O}(H^3/\epsilon^2)$ samples from $\mathcal{D}$.*

A New Cost Function

Theorem 2 and Theorem 3 use the number of iterations in gradient descent as the cost function. However, it is challenging to extend this approach to more complex methods like CG. Moreover, when the optimization problem is too large or computation time is constrained, determining the number of iterations becomes impractical. Additionally, since the number of iterations must be an integer, the learning error $1 + \epsilon$ cannot be reduced.

Considering above facts, we introduce an alternative cost measure. In mixed integer linear programming (MILP), the primal-dual integral was proposed by (Berthold 2013), aiming to determine the quality of best feasible solutions and the time when they are found during the solving process. Specifically, for an MILP instance minimizing $\mathbf{c}^\top\mathbf{x}$ subject to $\mathbf{A}^\top\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$, the primal and dual integrals are respectively defined as follows (Berthold 2013):

$$P(T) := \int_{t=0}^{T} \mathbf{c}^\top\mathbf{x}_t^* dt - T\mathbf{c}^\top\mathbf{x}^*,$$

$$D(T) := T\mathbf{c}^\top\mathbf{x}^* - \int_{t=0}^{T} \mathbf{z}_t^* dt,$$

where $\mathbf{x}_t^*$ is the best feasible solution found at time t, $\mathbf{z}_t^*$ is the best dual bound at time t, and $T\mathbf{c}^\top\mathbf{x}^*$ is an instance-specific constant that depends on the optimal solution x^*. Theoretically, both the primal and dual integrals are to be minimized, and takes an optimal value of 0, and a small dual integral will lead to both good and fast decisions (Gasse et al. 2022). Using primal and dual integrals to measure the performance of MILPs has been widely used in recent years (Qu et al. 2022; Zhang et al. 2023). Inspired by the concept of primal-dual integrals in MILP, we define

$$c(A_\rho, x) = \sum_{j=1}^{M} ||z^* - g^j(z_0, \rho)||,$$

where M is the number of iterations. Since gradient descent is translation invariant and f is convex, we introduce the following conventions for iteration augmentation for convenience:

1. $z^* = 0$ and $f(z^*) = 0$,
2. For $j > M$, $g^j(z_0, \rho) = z^*$ and correspondingly, $||z^* - g^j(z_0, \rho)|| = 0$.

Then $c(A_\rho, x) \leq M||z_0|| \leq Z\log(\nu/LZ)/\log(1 - \beta)$ for all ρ and x, meaning that the cost measure is also bounded. Suppose at each iteration, the step size ρ chosen in N adheres to Assumption 1, then $\mathcal{A}_N = \{A_\rho : \rho \in N\}$ is also finite, and therefore its pseudo-dimension is at most $[\log(\nu/LZ)/\log(1 - \beta)]\log|N|$.

Learning Complexity of Gradient Descent Algorithm Under the New Cost Function

It is evident that, similar to the primal-dual integral, the new cost function offers several advantages. It can be calculated even if the iteration is halted before reaching the optimal value (for instance, when computation time exceeds a specified tolerance); a lower cost function generally signifies an

algorithm that is both efficient and fast; among algorithms with the same number of iterations, the new cost function can discern the most effective one, theoretically corresponding to the algorithm with the highest convergence speed.

Drawing on the new cost function, we are able to establish new learning complexity results for the gradient descent algorithm. Consider the error $|c(A_\rho, x) - c(A_\eta, x)|$, we have the following conclusion:

Theorem 4 (Error estimation of new cost function)**.** *For any constant $C > 0$, instance x, and step sizes $\rho \in [\rho_l, \rho_u]$, $\eta \in [\rho_l, \rho_u]$ with $0 \leq \eta - \rho \leq \frac{\beta}{LZ}\frac{1-D(\rho)}{1-D(\rho)^H}D(\rho)^{-1}C$, we have*

$$|c(A_\rho, x) - c(A_\eta, x)| \leq C.$$

Let $K = \frac{\beta}{LZ}\frac{1-D(\rho_u)}{1-D(\rho_u)^H}D(\rho_u)^{-1}C$. We choose N as a minimal $K-$net, consisting of all integer multiples of K within $[\rho_l, \rho_u]$. Then $|N| \leq \rho_u/K + 1$. Consequently, by uniform convergence theorem and Corollary 1, an ERM algorithm can $(C + \epsilon, \delta)$-learn the optimal step size with sufficient number of samples. The following theorem demonstrates this result.

Theorem 5 (Generalization guarantees for GD on new cost function)**.** *For any constant $C > 0$, there exists a learning algorithm that $(C + \epsilon, \delta)$-learns the optimal algorithm in $\mathcal{A}$ using $m = \tilde{O}(H^3/\epsilon^2)$ samples from $\mathcal{D}$.*

Compared with the learning complexity results based on number of iterations (Gupta and Roughgarden 2016), we improved the learning complexity from $(1+\epsilon, \delta)$-learns the optimal algorithm to $(C+\epsilon, \delta)$-learns the optimal algorithm in $\mathcal{A}$ using $m = \tilde{O}(H^3/\epsilon^2)$ samples.

Learning Complexity of Conjugate Gradient Algorithm

Based on the new cost function, we are able to investigate the learning complexity of the conjugate gradient algorithm focusing on both the step size and the conjugate parameter. Recall the basic conjugate gradient algorithm for minimizing a function f given an initial point z_0 over $\mathbb{R}^N$:

1. Initialize $z := z_0$,
2. For $n \geq 1$, while $||\nabla f(z_n)||_2 > \nu$, $z_1 := z_0 - \rho\nabla f(z_0)$, $z_{n+1} := z_n - \rho\nabla f(z_n) - \eta(z_n - z_{n-1})$.

where ρ and η denote the step size and the conjugate parameter (we simply call both of them the step sizes). We denote the $j-$step iteration from z_0 and z_1 with step sizes ρ and η as $g^j(z_1, z_0, \rho, \eta) = z_j - \rho\nabla f(z_j) - \eta(z_j - z_{j-1})$. The j-th step starting from z_0 and z_1 can be viewed as the first step starting from z_j and z_{j-1}, thus $g^j(z_1, z_0, \rho, \eta) = g^1(z_j, z_{j-1}, \rho, \eta)$. For convenience, we define that $g(z_j, z_{j-1}, \rho, \eta) := g^1(z_j, z_{j-1}, \rho, \eta)$ and $g^0(z_1, z_0, \rho, \eta) = z_1$.

We make the following assumptions about the conjugate gradient algorithm for unconstrained optimization problems.

Assumption 2. *1) f is convex and L-smooth, i.e. for any z_1 and z_2, $||\nabla f(z_1) - \nabla f(z_2)|| \leq L||z_1 - z_2||$,*
2) The initial points are bounded with $\nu < ||z_0|| \leq Z$,
3) The step sizes ρ and η are drawn from some interval $\rho \in [\rho_l, \rho_u] \subset (0, \infty)$ and $\eta \in [\eta_l, \eta_u] \subset (0, \infty)$ respectively,
4) There exists a constant $\beta \in (0, 1)$ such that for any z_n and z_{n-1}, $||z_n - \rho\nabla f(z_n) - \eta(z_n - z_{n-1})|| \leq (1 - \beta)||z_n||$ for all $\rho \in [\rho_l, \rho_u]$ and $\eta \in [\eta_l, \eta_u] \subset (0, \infty)$ respectively.

If the condition 4) in Assumption 2 holds, then $||z_{n+1}|| \leq (1 - \beta)||z_n||$. Given the arbitrariness of n, it follows that $||z_n|| \leq (1 - \beta)||z_{n-1}||$. By induction, this implies $||z_n|| \leq (1 - \beta)^n||z_0||$, and consequently, $||g^j(z_i, z_{i-1}, \rho, \eta)|| \leq (1 - \beta)^j||z_i||$. Then by condition 1) in Assumption 2, we have $||\nabla g^j(z_1, z_0, \rho, \eta)|| \leq L(1 - \beta)^j||z_0||$. According to the termination condition $||\nabla f(z_n)||_2 \leq \nu$, the number of iterations is less than $H := \log(LZ/\nu)/\log(1 - \beta)$.

To establish the desired results, we first need to utilize some results related to second-order homogeneous linear recurrence relations.

Definition 4 (Second-order homogeneous linear recurrence relation)**.** *Any recurrence relation of the form*

$$x_n = ax_{n-1} + bx_{n-2}$$

is a second-order homogeneous linear recurrence relation.

The general formula for any second-order homogeneous linear recurrence relation is given in the following lemma.

Lemma 2 (General formula (Epp 2010))**.** *The general solution for $x_n = ax_{n-1} + bx_{n-2}$ is given by $x_n = c_1 r_1^n + c_2 r_2^n$, where r_1 and r_2 are the roots of the characteristic equation $r^2 - ar - b = 0$, c_1 and c_2 are constants by plugging in given x_0 and x_1.*

We first demonstrate the Lipschitz continuity of g.

Lemma 3 (Lipschitz continuity of g)**.** *For any w_n, $w_{n-1} \in \mathbb{R}^N$ in a CG iteration process (resp. y_n, $y_{n-1} \in \mathbb{R}^N$), and any step sizes $\rho \in [\rho_l, \rho_u]$, $\eta \in [\eta_l, \eta_u]$,*

$$\begin{aligned}&||g(w_n, w_{n-1}, \rho, \eta) - g(y_n, y_{n-1}, \rho, \eta)||\\ \leq&[D(\rho) + \eta]||w_n - y_n|| + \eta||w_{n-1} - y_{n-1}||,\end{aligned}$$

where $D(\rho) := max\{1, L\rho - 1\}$.

We extend Lemma 3 to the case of g^j and derive the following result.

Corollary 2 (Lipschitz continuity of g^j)**.** *For any w_n, $w_{n-1} \in \mathbb{R}^N$ in a CG iteration process started at w_0 (resp. y_n, $y_{n-1} \in \mathbb{R}^N$ started at y_0), and any step sizes $\rho \in [\rho_l, \rho_u]$, $\eta \in [\eta_l, \eta_u]$,*

$$\begin{aligned}&||g(w_n, w_{n-1}, \rho, \eta) - g(y_n, y_{n-1}, \rho, \eta)||\\ =&||g^n(w_1, w_0, \rho, \eta) - g^n(y_1, y_0, \rho, \eta)||\\ \leq&\left[\frac{D(\rho) - r_2}{r_1 - r_2}r_1^n + \frac{D(\rho) - r_1}{r_2 - r_1}r_2^n\right]||w_0 - y_0||\\ :=&F(\rho, \eta, n)||w_0 - y_0||,\end{aligned}$$

where r_1 and r_2 are the roots of characteristic equation of second-order homogeneous linear recurrence relation $x_n = [D(\rho) + \eta]x_{n-1} + \eta x_{n-2}$.

We aim to derive a bound on how far two conjugate gradient iteration processes, with different algorithm parameters ρ and η, can derive from each other when they start from the same point. In detail, we assume that the two algorithms use the same starting point z_0, but different step sizes (ρ_1, η_1) and (ρ_2, η_2) respectively at any iteration step j. Our goal is to estimate the upper bound of $||g^j(z_1, z_0, \rho_1, \eta_1) - g^j(z_1', z_0, \rho_2, \eta_2)||$. Here z_1' corresponds to another iteration processes different from z_1 due to a different choice of step sizes ρ_2, η_2.

To obtain the estimation, we first consider two special cases. The first case involves fixing the step size η and examining the error only when ρ changes.

Lemma 4 (Iteration error estimation fixing the step size η)**.** *For any initial point $z_0 \in \mathbb{R}^N$, iteration step $j \geq 2$, step sizes $\eta \in \mathbb{R}$, $\rho_1 \in \mathbb{R}$, $\rho_2 \in \mathbb{R}$ such that $\rho_1 \leq \rho_2$, we have*

$$\begin{aligned}
&||g^j(z_1, z_0, \rho_1, \eta) - g^j(z_1', z_0, \rho_2, \eta)|| \\
\leq&(\rho_2 - \rho_1)\Bigg[(\frac{R_0 r_2 - R_1}{r_2 - r_1} r_1^j + \frac{R_0 r_1 - R_1}{r_1 - r_2} r_2^j) \\
&- \frac{LZ(1-\beta)^{j+2}}{[D(\rho_1)+\eta](1-\beta)+\eta-(1-\beta)^2}\Bigg] \\
:=&(\rho_2 - \rho_1) G(\rho_1, \eta, j),
\end{aligned}$$

where $D(\rho) := \max\{1, L\rho - 1\}$, r_1 and r_2 are the roots of the characteristic equation of second-order homogeneous linear recurrence relation $A_n = [D(\rho_1)+\eta]A_{n-1} + \eta A_{n-2}$,

$$\begin{aligned}
R_0 &= \frac{LZ(1-\beta)^2}{[D(\rho_1)+\eta](1-\beta)+\eta-(1-\beta)^2}, \\
R_1 &= \frac{D(\rho_1)LZ}{\beta} + \frac{LZ(1-\beta)^3}{[D(\rho_1)+\eta](1-\beta)+\eta-(1-\beta)^2}.
\end{aligned}$$

Correspondingly, we can analyze the error estimation only when η changes while the step size ρ is fixed. To save the space, we present the results in (Jiao, Liu, and Chen 2024, Lemma 5).

With the above two results, we can address the general case where both ρ and η change. We have

Lemma 6 (Iteration error estimation)**.** *For any initial point $z_0 \in \mathbb{R}^N$, iteration step $j \geq 2$, step sizes $\rho_1 \in \mathbb{R}$, $\rho_2 \in \mathbb{R}$ $\eta_1 \in \mathbb{R}$, $\eta_2 \in \mathbb{R}$, we have*

$$\begin{aligned}
&||g^j(z_1, z_0, \rho_1, \eta_1) - g^j(z_1', z_0, \rho_2, \eta_2)|| \\
\leq&|\rho_2 - \rho_1| G(\rho_1, \eta_1, j) + |\eta_2 - \eta_1| H(\eta^*, \rho_1, j),
\end{aligned}$$

where further details of $H(\eta^*, \rho_1, j)$ and the proof are presented in (Jiao, Liu, and Chen 2024).

We then get the following corollaries.

Corollary 3. *1) For any $C > 0$, if ρ is invariant in the whole iteration process and $|\eta_2 - \eta_1| \leq \frac{C}{H(\eta^*, \rho, j)}$, then the iteration error satisfies $||g^j(z_1, z_0, \rho, \eta_1) - g^j(z_1', z_0, \rho, \eta_2)|| \leq C$.*

2) For any $C > 0$, if η is invariant during the whole iteration process, and $|\rho_2 - \rho_1| \leq \frac{C}{G(\eta, \rho_1, j)}$, then the iteration error satisfies $||g^j(z_1, z_0, \rho_1, \eta) - g^j(z_1', z_0, \rho_2, \eta)|| \leq C$.

3) More generally, for any $C > 0$, if the parameter combination satisfies $|\rho_2 - \rho_1| G(\rho_1, \eta_1, j) + |\eta_2 - \eta_1| H(\eta^*, \rho_1, j) \leq C$, then we have $||g^j(z_1, z_0, \rho_1, \eta_1) - g^j(z_1', z_0, \rho_2, \eta_2)|| \leq C$.

Finally, we present a more general theorem concerning the learning complexity of the conjugate gradient algorithm with respect to the algorithm parameters. This result follows directly from Lemma 6.

Theorem 6 (Iteration error estimation)**.** *For any z_0, $j \geq 2$, ρ_1, ρ_2, η_1, and η_2 with $|\rho_2 - \rho_1| < \varepsilon_1$ and $|\eta_2 - \eta_1| < \varepsilon_2$,*

$$\begin{aligned}
&||g^j(z_1, z_0, \rho_1, \eta_1) - g^j(z_1', z_0, \rho_2, \eta_2)|| \\
\leq&\varepsilon_1 G(\rho_1, \eta_1, j) + \varepsilon_2 H(\eta^*, \rho_1, j),
\end{aligned}$$

where z_1' corresponds to another iteration processes different from z_1 due to a different choice of step sizes ρ_2, η_2. Therefore, $||g^j(z_1, z_0, \rho_1, \eta_1) - g^j(z_1', z_0, \rho_2, \eta_2)||$ converges to zero if ε_1 and ε_2 tend to zero.

We next refine our cost function. We can also define $c(A_{\rho,\eta}, x) = \sum_{i=0}^{M} ||z^* - g^i(z_1, z_0, \rho, \eta)||$, where $A_{\rho,\eta}$ stands for the conjugate gradient algorithm with step sizes ρ and η. Note that $M < H$, also for convenience, we introduce the following conventions:

1. $z^* = 0$ and $f(z^*) = 0$,
2. For $j > M$, $g^j(z_1, z_0, \rho, \eta) = z^*$ and correspondingly, $||z^* - g^j(z_1, z_0, \rho, \eta)|| = 0$.

Then $c(A_{\rho,\eta}, x) \leq M||z_0|| \leq Z \log(\nu/LZ)/\log(1-\beta)$ for all ρ, η and x. This means the cost measure is bounded.

Consider the error in the cost function between different algorithms, measured by $|c(A_{\rho_1,\eta_1}, x) - c(A_{\rho_2,\eta_2}, x)|$, we have the following theorem.

Theorem 7 (Error estimation of cost function)**.** *For any instance x, step sizes ρ_1, ρ_2, η_1, η_2, and constant $C > 0$, if*

$$\begin{aligned}
&|\rho_2 - \rho_1| \frac{LZD(\rho_1)}{\beta} + \sum_{j=1}^{M} \Bigg[G(\rho_1, \eta_1, j)] \\
&+ |\eta_2 - \eta_1| \sum_{j=1}^{M} [H(\eta^*, \rho_1, j)\Bigg] \leq C,
\end{aligned}$$

Then

$$|c(A_{\rho_1,\eta_1}, x) - c(A_{\rho_2,\eta_2}, x)| \leq C.$$

We readily derive the following corollary, which provides a sufficient condition for Theorem 7.

Corollary 4. *For any constant $C > 0$, instance x, step sizes ρ_1, ρ_2, η_1, η_2 with*

$$|\rho_2 - \rho_1| \left[\frac{LZD(\rho_1)}{\beta} + \sum_{j=1}^{M} G(\rho_1, \eta_1, j)\right] \leq C/2,$$

and

$$|\eta_2 - \eta_1| \sum_{j=1}^{M} H(\eta^*, \rho_1, j) \leq C/2,$$

we have

$$|c(A_{\rho_1,\eta_1}, x) - c(A_{\rho_2,\eta_2}, x)| \leq C.$$

In the remainder of this section, for convenience, we assume $\rho_1 \leq \rho_2$ and $\eta_1 \leq \eta_2$. For ρ_1 and ρ_2 (and similarly for η_1 and η_2), due to $(\rho_2 - \rho_1)[\frac{LZD(\rho_1)}{\beta} + \sum_{j=1}^{M} G(\rho_1, \eta_1, j)] \leq C/2$, which implies that if we find an upper bound of $\frac{LZD(\rho_1)}{\beta} + \sum_{j=1}^{M} G(\rho_1, \eta_1, j)$, denoted as G^*, then Corollary 4 holds if $\rho_2 - \rho_1 \leq \frac{C}{2G^*}$. Thus, the problem reduces to finding an appropriate G^*. One straightforward method to obtain G^* is to compute $G(\rho_1, \eta_1, j)$ for all $j = 1$ to M. If we let the maximum of these M values be $\tilde{G}$, then $G^* = \frac{LZD(\rho_1)}{\beta} + H\tilde{G}$. The complexity of calculating this G^* is $O(Hn^H)$, where n is a constant and n^H measures the complexity of computing each $G(\rho_1, \eta_1, j)$. The following lemma provides a refined estimation of the upper bound G^* to reduce the complexity.

Lemma 7 (A refined upper bound for G)**.**

$$G^* = \frac{LZD(\rho_1)}{\beta} + R^* \left[r_1 \frac{1 - r_1^H}{1 - r_1} \right]$$

is an upper bound for $\frac{LZD(\rho_1)}{\beta} + \sum_{j=1}^{M} G(\rho_1, \eta_1, j)$*, where* $r_1 \geq r_2$ *are the roots of the characteristic equation of second-order homogeneous linear recurrence relation* $A_n = [D(\rho_1) + \eta_1]A_{n-1} + \eta_1 A_{n-2}$*, and*

$$\begin{aligned}
R_0 =& \frac{LZ(1-\beta)^2}{[D(\rho_1) + \eta_1](1-\beta) + \eta_1 - (1-\beta)^2}, \\
R_1 =& \frac{LZD(\rho_1)}{\beta} + \frac{LZ(1-\beta)^3}{[D(\rho_1) + \eta_1](1-\beta) + \eta_1 - (1-\beta)^2}, \\
R^* =& \frac{R_0 r_2 - R_1}{r_2 - r_1} + \left| \frac{R_0 r_1 - R_1}{r_1 - r_2} \right|.
\end{aligned}$$

Similarly, for $\sum_{i=1}^{M} [H(\eta^*, \rho_1, j)]$ and its upper bound H^*, we have

Lemma 8 (A refined upper bound for H)**.**

$$H^* = R^{*'} \left[r_1 \frac{1 - r_1^H}{1 - r_1} \right]$$

is an upper bound for $\sum_{i=1}^{M} [H(\eta^*, \rho_1, i)]$*, where* $r_1 \geq r_2$ *are the roots to the characteristic equation of second-order homogeneous linear recurrence relation* $A_n = [D(\rho_1) + \eta_1]A_{n-1} + \eta_1 A_{n-2}$*, and*

$$\begin{aligned}
R_0 =& \frac{F(\eta^*)}{D(\rho_1) + 2\eta_1 - 1}, \\
R_1 =& F(\eta^*) \frac{1}{\eta_1} \left[1 + \frac{1}{D(\rho_1) + 2\eta_1 - 1} \right], \\
F(\eta^*) =& \max_{\substack{n \in (0,1,\cdots,j-1) \\ \eta \in [\eta_l, \eta_u]}} \eta^n \left[\rho_1 LZ + \frac{\rho_1 Z}{\eta - L} L \right] - \frac{\rho_1 Z}{\eta - L} L^{n+1}, \\
R^{*'} =& \frac{R_0 r_2 - R_1}{r_2 - r_1} + \left| \frac{R_0 r_1 - R_1}{r_1 - r_2} \right|.
\end{aligned}$$

Lemma 7 and Lemma 8 demonstrate that for any given instance and for two distinct CG algorithms with different step sizes, the difference in their corresponding cost functions is bounded, if the difference between the step sizes is below a certain tolerance. Thus, we derive the following result:

Theorem 8 (Error estimation of cost function)**.** *For any constant* $C > 0$*, instance* x*, step sizes* ρ_1*,* ρ_2*,* η_1*,* η_2 *satisfying*

$$|\rho_2 - \rho_1| \leq \frac{C}{2G^*}, \quad |\eta_2 - \eta_1| \leq \frac{C}{2H^*},$$

we have the difference between the corresponding cost functions bounded by

$$|c(A_{\rho_1, \eta_1}, x) - c(A_{\rho_2, \eta_2}, x)| \leq C,$$

where G^* *and* H^* *are defined in Lemma 7 and Lemma 8.*

We can also set a K_ρ-net (resp. a K_η-net) for ρ (resp. for η). Suppose N_ρ is a minimal K_ρ-net, consisting of all integer multiples of K_ρ that lie in $[\rho_l, \rho_u]$. Then $|N_\rho| \leq \rho_u / K_\rho + 1$ (and the same applies to N_η). Let

$$K_\rho = \max_{\substack{\rho \in [\rho_l, \rho_u] \\ \eta \in [\eta_l, \eta_u]}} \frac{C}{2G^*(\rho, \eta)}, \quad K_\eta = \max_{\substack{\rho \in [\rho_l, \rho_u] \\ \eta \in [\eta_l, \eta_u]}} \frac{C}{2H^*(\rho, \eta)},$$

then $\log |N_\rho| \sim \tilde{O}(H)$ and $\log |N_\eta| \sim \tilde{O}(H)$.

Finally, by uniform convergence theorem and Corollary 1, an ERM algorithm can $(C + \epsilon, \delta)$-learn the optimal step size with a sufficient number of samples. The following theorem demonstrates this result.

Theorem 9 (Generalization Guarantees for CG)**.** *For any constant* $C > 0$*, there exists a learning algorithm that* $(C + \epsilon, \delta)$*-learns the optimal algorithm in* $\mathcal{A}_{N_\rho, N_\eta}$ *using* $\tilde{O}(H^4/\epsilon^2)$ *samples from* $\mathcal{D}$*.*

By now, we have shown the existence a learning algorithm capable of probabilistically identifying the optimal algorithm with a sufficiently large sample size by using the proposed new cost function under the learning complexity framework.

Conclusion

In this study, we employed a learning-complexity framework for data-driven algorithm design. One of our major contributions is to introduce a new cost function based on the sum of distances between current and optimal values for each iteration. We proved that for both gradient descent and conjugate gradient methods, optimal learning algorithms exist that can $(C + \epsilon, \delta)$-learn the optimal algorithm using $m = \tilde{O}(H^3/\epsilon^2)$ and $m = \tilde{O}(H^4/\epsilon^2)$ samples, respectively.

Future researches include: extend this framework to other optimization algorithms, develop new cost functions for better performance, or investigate the scalability for large-scale problems. These efforts will enhance the applicability and effectiveness of data-driven algorithm design in solving complex optimization challenges. In recent years, researchers have predominantly focused on the worst-case complexity of algorithms, emphasizing the derivation of upper bounds. In contrast, lower bounds on complexity are often applicable only to specific problems. We believe that exploring this area further presents an intriguing avenue for future research.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 11991023 and 12371324) and National Key R&D Program of China (No. 2022YFA1004000).

References

Alabi, D.; Kalai, A. T.; Liggett, K.; Musco, C.; Tzamos, C.; and Vitercik, E. 2019. Learning to prune: speeding up repeated computations. In *Proceedings of the Thirty-Second Conference on Learning Theory*, 30–33.

Andrei, N. 2020. *Nonlinear Conjugate Gradient Methods for Unconstrained Optimization*. SpringerCham.

Andrychowicz, M.; Denil, M.; Gómez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; and de Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*.

Anthony, M.; and Bartlett, P. L. 1999. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

Babaie-Kafaki, S. 2023. A survey on the DaiLiao family of nonlinear conjugate gradient methods. *RAIRO-Operations Research*, 57(1): 43–58.

Balcan, M.-F.; DeBlasio, D.; Dick, T.; Kingsford, C.; Sandholm, T.; and Vitercik, E. 2021. 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*, 919932.

Balcan, M.-F.; Dick, T.; Sandholm, T.; and Vitercik, E. 2018. Learning to branch. In *Proceedings of the 35th International Conference on Machine Learning*, 344–353.

Balcan, M.-F.; Nagarajan, V.; Vitercik, E.; and White, C. 2017. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Proceedings of the 2017 Conference on Learning Theory*, 213–274.

Balcan, M.-F.; Prasad, S.; Sandholm, T.; and Vitercik, E. 2022. Improved sample complexity bounds for branch-and-cut. arXiv:2111.11207.

Berthold, T. 2013. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6): 611–614.

Dai, Y. H.; and Yuan, Y. X. 1999. A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property. *SIAM J. Optimization*, 10(1): 177–182.

Denevi, G.; Ciliberto, C.; Grazzi, R.; and Pontil, M. 2019. Learning-to-learn stochastic gradient descent with biased regularization. In *Proceedings of the 36th International Conference on Machine Learning*.

Du, X.; Zhang, P.; and Ma, W. 2016. Some modified conjugate gradient methods for unconstrained optimization. *Journal of Computational and Applied Mathematics*, 305: 92–114.

Epp, S. S. 2010. *Discrete Mathematics with Applications*. Brooks/Cole Publishing Company, 4th edition.

Fletcher, R.; and Reeves, C. M. 1964. Function minimization by conjugate gradients. *The Computer Journal*, 7: 149–154.

Gasse, M.; Bowly, S.; Cappart, Q.; Charfreitag, J.; Charlin, L.; Chételat, D.; Chmiela, A.; Dumouchelle, J.; Gleixner, A.; Kazachkov, A. M.; Khalil, E.; Lichocki, P.; Lodi, A.; Lubin, M.; Maddison, C. J.; Christopher, M.; Papageorgiou, D. J.; Parjadis, A.; Pokutta, S.; Prouvost, A.; Scavuzzo, L.; Zarpellon, G.; Yang, L.; Lai, S.; Wang, A.; Luo, X.; Zhou, X.; Huang, H.; Shao, S.; Zhu, Y.; Zhang, D.; Quan, T.; Cao, Z.; Xu, Y.; Huang, Z.; Zhou, S.; Binbin, C.; Minggui, H.; Hao, H.; Zhiyu, Z.; Zhiwu, A.; and Kun, M. 2022. The machine learning for combinatorial optimization competition (ML4CO): Results and insights. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, 220–231.

Gupta, R.; and Roughgarden, T. 2016. A PAC approach to application-specific algorithm selection. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, 123134.

Harris, K.; Anagnostides, I.; Farina, G.; Khodak, M.; Wu, Z. S.; and Sandholm, T. 2023. Meta-learning in games. arXiv:2209.14110.

Hestenes, M. R.; and Stiefel, E. 1952. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49: 409–436.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2022. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9): 5149–5169.

Jiao, X.; Liu, J.; and Chen, Z. 2024. Learning complexity of gradient descent and conjugate gradient algorithms-extended version. arXiv:2412.13473.

Kearns, M. J.; and Vazirani, U. V. 1994. *An Introduction to Computational Learning Theory*. Cambridge, MA, USA: MIT Press.

Li, K.; and Malik, J. 2016. Learning to optimize. arXiv:1606.01885.

Liu, Z.; Liu, H.; and Dai, Y. 2020. An improved Dai-Kou conjugate gradient algorithm for unconstrained optimization. *Computational Optimization and Applications*, 75(1): 145–167.

Maclaurin, D.; Duvenaud, D.; and Adams, R. 2015. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning*.

Maria-Florina, B.; Khodak, M.; Sharma, D.; and Talwalkar, A. 2021. Learning-to-learn non-convex piecewise-Lipschitz functions. In *Advances in Neural Information Processing Systems*, 15056–15069.

Pollard, D. 1984. *Convergence of Stochastic Processes*. Springer New York.

Polyak, B. 1969. The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, 9(4): 94–112.

Qu, Q.; Li, X.; Zhou, Y.; Zeng, J.; Yuan, M.; Wang, J.; Lv, J.; Liu, K.; and Mao, K. 2022. An improved reinforcement learning algorithm for learning to branch. arXiv:2201.06213.

Snyman, J. 2005. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer New York.

Stanimirovi, P. S.; Ivanov, B.; Ma, H.; and Mosi, D. 2020. A survey of gradient methods for solving nonlinear optimization. *Electronic Research Archive*, 28(4): 1573–1624.

Vapnik, V. 1991. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*.

Wang, X.; Yuan, S.; Wu, C.; and Ge, R. 2021. Guarantees for tuning the step size using a learning-to-learn approach. In *Proceedings of the 38th International Conference on Machine Learning*, 10981–10990.

Yang, Z. 2022. Adaptive stochastic conjugate gradient for machine learning. *Expert Systems with Applications*, 206: 117719.

Zhang, J.; Liu, C.; Li, X.; Zhen, H.; Yuan, M.; Li, Y.; and Yan, J. 2023. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519: 205–217.