Towards Robust Learning to Optimize with Theoretical Guarantees

Qingyu Song
CUHK

Wei Lin
CUHK

Juncheng Wang
HKBU

Hong Xu
CUHK

Abstract

Learning to optimize (L2O) is an emerging technique to solve mathematical optimization problems with learning-based methods. Although with great success in many real-world scenarios such as wireless communications, computer networks, and electronic design, existing L2O works lack theoretical demonstration of their performance and robustness in out-of-distribution (OOD) scenarios. We address this gap by providing comprehensive proofs. First, we prove a sufficient condition for a robust L2O model with homogeneous convergence rates over all In-Distribution (InD) instances. We assume an L2O model achieves robustness for an InD scenario. Based on our proposed methodology of aligning OOD problems to InD problems, we also demonstrate that the L2O model's convergence rate in OOD scenarios will deteriorate by an equation of the L2O model's input features. Moreover, we propose an L2O model with a concise gradient-only feature construction and a novel gradient-based history modeling method. Numerical simulation demonstrates that our proposed model outperforms the state-of-the-art baseline in both InD and OOD scenarios and achieves up to 10 $\times$ convergence speedup. The code of our method can be found from https://github.com/NetX-lab/GoMathL2O-Official.

1. Introduction

Learning to Optimize (L2O) is a promising new approach in applying learning-based methods to tackle optimization problems. In particular, L2O concentrates on problems with well-defined objective functions and constraints [7]. Thus, black-box optimization strategies, such as Bayesian Optimization [24], typically fall outside its scope. L2O has shown benefits in problems from various domains, including LASSO regression in sparse coding using multilayer perceptrons [8], and utility maximization in resource allocation wherein neural networks (NN) serve to approximate the expensive matrix inversion [11].

L2O can be categorized into three main types: black-box [6, 22, 26, 31], algorithm-unrolling [11, 21, 33], and math-inspired [9, 14]. Black-box L2O approaches the optimization problem as a traditional pattern recognition task, approximating a mapping function from manually constructed features to the solutions [26]. Algorithm-unrolling L2O leverages well-defined algorithms, such as gradient descent [19], to approximate the solutions of complex calculations. Besides, much research has gone into explainable and trustworthy L2O. For example, Heaton et al. [9] employ an existing algorithm to prevent the L2O model from entering irrecoverable areas. Liu et al. [14] introduce a mathematics-driven L2O (Math-L2O) framework for convex optimization, offering a general workflow for formulating an L2O model. Despite empirical results, a theoretical analysis on the robustness of L2O models under out-of-distribution (OOD) conditions is still missing in [14].

OOD generalization for L2O has emerged as a vital issue, often considered more critical in L2O than in other deep learning applications [23]. For L2O, OOD's challenge involves resolving previously unseen problems, potentially involving novel optimization problems with unique objectives [30]. Guaranteeing convergence in OOD scenarios remains elusive. For instance, a model's output in an OOD scenario could potentially veer into unpredictable areas when the domain changes significantly to an InD scenario.

Numerous efforts have been made to enhance the robustness of L2O models in training. Lv et al. [16] employ data augmentation to prevent L2O models from overfitting to specific tasks. Almeida et al. [2] transform the L2O model into a hyperparameter tuner for existing optimization algorithms. Wichrowska et al. [29] focus on minimizing parameters in NNs and assembling heterogeneous optimization tasks. Liu et al. [14] try to regularize L2O models with inspirations from existing algorithms. However, these studies predominantly aim to mitigate the limitations inherent in existing L2O methods, with no comprehensive analysis conducted on the impact of OOD on the deterioration of convergence. This gap in the literature motivates us to quantify this deterioration with rigorous analysis.

The central thesis of this paper is to propose a general and robust L2O model for both InD and OOD scenarios. Chiefly, we first investigate L2O's convergence behavior in InD contexts and derive the criteria for a uniformly robust

model applicable to all InD instances. Then, we characterize L2O's degradation of convergence under OOD conditions, presenting our findings as a series of corollaries. The main contributions of this paper are as follows.

1. We propose a methodology to link the L2O model's performances in InD and OOD situations based on the Math-L2O approach from Liu et al. [14]. First, we construct a virtual feature by subtracting the L2O model's input feature in InD from that in OOD. We then compute the corresponding difference in the model's outputs by applying this virtual feature. To depict a comprehensive deviation of OOD from InD, we align the variable sequence from an OOD situation with that from InD and construct a trajectory of virtual features. We use this trajectory to illustrate OOD's divergence from InD and then conduct theoretical analyses.
2. We establish the criteria for a robust L2O model in an InD setting and examine its response to OOD. First, we present a sufficient condition to guarantee a homogeneous convergence improvement in each iteration, confirming robustness in InD scenarios. Then, we derive the equations describing convergence gain in a single iteration and the overall convergence rate of the entire sequence relative to our proposed virtual feature. A collection of theorems and observations underscore that the magnitude of virtual features inherently exacerbates the deterioration of convergence in OOD situations.
3. Based on our theoretical insights, we propose a robust L2O model, GO-Math-L2O, that exclusively employs gradients as input features. This gradient-only approach enables a more concise virtual feature in OOD settings. We introduce a new gradient-only history modeling technique to model the optimization process's historical sequence. This method employs gradient (and subgradient) values as status indicators to modulate updates provided by the L2O model. We propose to recover the historical subgradient from an inversible model definition, thus eliminating the ambiguity of subgradient selection.
4. Through numerical experiments, we show that GO-Math-L2O outperforms state-of-the-art (SOTA) L2O models on convergence and optimality across both InD and OOD scenarios. Following training with a synthetic dataset, we deploy various OOD test cases with identical optimal values. Our proposed model's convergence speed is up to $10\times$ faster than SOTA L2O models in OOD scenarios.

The rest of this paper is organized as follows. In Sec. 2, we define OOD problems for L2O. In Sec. 3, we propose a method to quantify the solutions given by an L2O model in OOD scenarios. Then, in Sec. 4, we derive the convergence rate of an L2O model in OOD scenarios. Based on this, we propose our robust GO-Math-L2O model in Sec. 5. We empirically verify the proposed model with simulations in Sec. 6, and conclude the work in Sec. 7.

Notations: A smooth convex function and a non-smooth convex function are denoted by f and r, respectively. NNs' input vectors are denoted by z and z'. Variables of an optimization problem are denoted by x and x'. The optimal solution is denoted by x^*. An iteration and stopping iteration are denoted by k and K, respectively. A smooth gradient at x_k and a set of subgradients at x_k are denoted by $\nabla f(x_k)$ and $\partial r(x_k)$, respectively. A subgradient value of $\partial r(x_k)$ is denoted by g_k. Frobenius norm for a matrix and $L2$-norm for a vector is denoted by $\|\cdot\|_{\mathrm{F}}$ and $\|\cdot\|$ respectively. Transpose is defined by $^\top$. The maximum length of history modeling is denoted by T. The Jacobian matrix of a vector-to-vector function is denoted by $\mathbf{J}$. An L2O model is denoted by d. A NN is denoted by operator $\mathbf{N}$.

2. Definitions

In this section, we first introduce the objective of the L2O problem. We then introduce the Math-L2O model in [14], whose iterative updates are defined by NNs. Last, we define the domains for both InD and OOD scenarios, which leads to the definitions of InD L2O and OOD L2O problems.

2.1. Optimizee (Optimization Objective)

Consider function $F(x) = f(x) + r(x)$. Here, $f(x)$ is a L-smooth function, and $r(x)$ is a non-smooth function. They are defined within the following function spaces:

$$\begin{aligned}\mathcal{F}_L\left(\mathbb{R}^n\right) =& \{f : \mathbb{R}^n \to \mathbb{R} | f \text{ is convex, differentiable, and} \\ & \|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|, \forall x, y \in \mathbb{R}^n\}, \\ \mathcal{F}\left(\mathbb{R}^n\right) =& \{r : \mathbb{R}^n \to \mathbb{R} | r \text{ is proper, closed, and convex}\}.\end{aligned}$$

We assume $r(x)$ is sub-differentiable, with its subgradient set at any point x defined below:

$$\partial r(x) = \{g \in \mathbb{R}^n \mid r(y) - r(x) \geq g^\top (y-x), \forall x, y \in \mathbb{R}^n\}.$$

We note here that the above optimization objective applies to both the InD and the OOD scenarios.

2.2. Optimizor (L2O Model)

Denote the L2O model as $d(z)$, where the input vector space is designated as $\mathcal{Z}$ such that $z \in \mathcal{Z} \subseteq \mathbb{R}^m$. We define $d(z)$ as a function mapping within the given function space [14]:

$$\begin{aligned}\mathcal{D}_C(\mathcal{Z}) = \{d : \mathcal{Z} \to \mathbb{R}^n \mid & d \text{ is differentiable}, \\ & \|\mathbf{J}_{d(z)}\|_{\mathrm{F}} \leq C, \forall z \in \mathcal{Z}, C \in \mathbb{R}^+\}.\end{aligned} \tag{1}$$

We choose features from x and $F(x)$ to define z, offering a wide range of feasible options. For instance, z could be defined with the optimization variable and its gradient as $\left[x^\top, \nabla f(x)^\top\right]^\top$ in [14]. Different from [14], we propose to define z solely as $\nabla f(x)$ to improve convergence

in OOD scenarios. From our experimental results, our approach achieves near-optimal solutions in some OOD cases and more robust performance than SOTA baselines in all OOD scenarios. Moreover, Corollaries 2 and 3 theoretically demonstrate the outperformance over the method in [14].

$d(z)$ iteratively updates the optimization variable. At each iteration k, given the previous variable $x_{k-1} \in \mathbb{R}^n$ and the input vector z_{k-1} for the L2O model, $d(z_{k-1})$ updates x_k as follows:

$$x_k = x_{k-1} - d(z_{k-1}). \tag{2}$$

2.3. InD and OOD Problems

The InD and OOD problems share the same space of optimization objective defined in Sec. 2.1 but with different optimization objectives or variable domains. Consider a convex and compact set, $\mathcal{S}_P \subseteq \mathbb{R}^n$. The complementary set of $\mathcal{S}_P$ is denoted as $\mathcal{S}_O$, such that $\mathcal{S}_O := \mathbb{R}^n \backslash \mathcal{S}_P$. We also suppose the existence of two function sets: $\mathcal{F}_{L,P} \subseteq \mathcal{F}_L(\mathbb{R}^n)$ and $\mathcal{F}_P \subseteq \mathcal{F}(\mathbb{R}^n)$. We define InD optimization problems as follows:

$$\min_x F(x), \tag{P}$$

where $x \in \mathcal{S}_P$, $F(x) = f(x) + r(x)$, $f \in \mathcal{F}_{L,P}$, and $r \in \mathcal{F}_P$. The dataset employed for training an L2O model is derived from a specific domain of x, f, and r. Consider an L2O model $d(z)$ that has undergone training with a domain of x, f, and r, sampled from Problem P. We then define the *InD L2O Problem* as: Given any initial point $x_0 \in \mathcal{S}_P$, using $d(z)$ to iteratively update x_0 in order to find a solution for any arbitrary InD problem as depicted in Problem P.

Note that instances outside this domain potentially yield more erroneous $d(z)$ outputs. Furthermore, non-learning algorithms, such as gradient descent, have demonstrated robustness across all domains [19]. One of the main goals of this paper is to propose an L2O model that is robust to OOD.

We characterize OOD in the context of L2O in the optimization objective's domain. We define the *OOD L2O Problem* as: Consider an L2O model $d(z)$ that has undergone training with a domain of x, f, and r, sampled from Problem P, using $d(z)$ to iteratively update $x'_0 \in \mathcal{S}_O$ in order to a solution for any following problem:

$$\min_{x'} F'(x'). \tag{O}$$

where $F'(x') = f'(x') + r'(x')$, $f' \notin \mathcal{F}_{L,P}$, and $r' \notin \mathcal{F}_P$.

We delineate the InD and OOD input vector spaces of $d(z)$. We denote the input vector spaces for an L2O model in the context of *InD L2O Problem* and *OOD L2O Problem* as $\mathcal{Z}_P$ and $\mathcal{Z}_O$, respectively. Then, we choose features of the variables and the objective functions to construct the input feature of $d(z)$. Specifically, we define $\mathcal{Z}_P$ and $\mathcal{Z}_O$ as the ensuing sets:

$$\begin{aligned}\mathcal{Z}_P =& \big\{ \big[x\text{-feature}^\top, f(x)\text{-feature}^\top, r(x)\text{-feature}^\top, \dots \big]^\top \\ & \mid \forall x \in \mathcal{S}_P, \forall f' \in \mathcal{F}_{L,P}, \forall r' \in \mathcal{F}_P \big\}, \\ \mathcal{Z}_O =& \big\{ \big[x'\text{-feature}^\top, f'(x')\text{-feature}^\top, r'(x')\text{-feature}^\top, \dots \big]^\top \\ & \mid \exists x' \in \mathcal{S}_O \text{ or } \exists f' \notin \mathcal{F}_{L,P} \text{ or } \exists r' \notin \mathcal{F}_P \big\},\end{aligned}$$

where "..." represents other feasible features such as the history of x. Some feasible feature constructions for x, $f(x)$, and $r(x)$ include x itself, $\nabla f(x)$, and $\partial r(x)$. Later in Sec. 5, we show how to contruct the input features of the L2O model $d(z)$ based only on $\nabla f(x)$ and $\partial r(x)$.

3. Virtual Feature and Trajectory

In this section, we introduce a virtual feature methodology to correlate any arbitrary variable yielded by the L2O model in the OOD scenario (x'_k) to a corresponding variable x_k in the InD scenario. The virtual features are generated as a linear combination of the OOD and InD features and serve as a bridge to connect each L2O model's OOD outcome to its InD outcome. We then leverage the virtual-feature method to connect OOD and InD variable trajectories generated by the L2O model. Since the convergence of InD trajectories is deterministic, such a method facilitates the convergence and robustness analysis for OOD scenarios in Sec. 4.

3.1. Virtual Feature

Consider an arbitrary OOD variable $x' \in \mathcal{S}_O$ and a InD variable $x \in \mathcal{S}_P$ yielded by the L2O model. Let $s \in \mathbb{R}^n$ such that $s = x' - x$. In that case, we define the difference s' between the L2O model's features in the OOD scenario z' and the features in the InD scenario z. From the Mean Value Theorem [20], there exists a virtual Jacobian matrix $\mathbf{J}_d$, $\|\mathbf{J}_d\| \leq C\sqrt{n}$ such that the following inequality holds:

$$d(z') = d(z) + \mathbf{J}_d(z' - z) = d(z) + \mathbf{J}_d s'. \tag{3}$$

The demonstrations are in Sec. 8.1. From equation 3, we can relate any variable of the L2O model in the OOD scenario to the InD scenario. Although the virtual Jacobian matrix $\mathbf{J}_d$ is non-deterministic, it is upper bounded from the definition of $d(z)$ in equation 1. This suffices for a quantitative analysis of the impact of the "shift" s' on convergence. For instance, our proposed Theorem 1 in Sec. 4 provides an upper bound on the convergence gain for a single iteration.

3.2. Trajectory

For the OOD Problem O, denote the initial variable as $x'_0 \in \mathcal{S}_O$. In the optimization process, we have two trajectories for the variable x' and the features of the L2O model z':

$$\{x'_0, x'_1, x'_2, \dots, x'_K\}, \{z'_0, z'_1, z'_2, \dots, z'_K\}.$$

where $x'_k \in \mathcal{S}_O, z'_k \in \mathcal{Z}_O, k = 0, 1, 2, \ldots, K$. Similarly, for the InD Problem P, denote the initial variable as $x_0 \in \mathcal{S}_P$. We have also have two trajectories for the variables x and the features of the L2O model z:

$$\{x_0, x_1, x_2, \ldots, x_K\}, \{z_0, z_1, z_2, \ldots, z_K\},$$

where $x_k \in \mathcal{S}_P, z_k \in \mathcal{Z}_P, k = 0, 1, 2, \ldots, K$. Utilizing the definitions in Sec. 3.1, we compute the differences between the variables and the features of the OOD trajectories and the InD trajectories as follows:

$$\{s_0, s_1, s_2, \ldots, s_K\}, \{s'_0, s'_1, s'_2, \ldots, s'_K\},$$

where $s_k := x'_k - x_k$ and $s'_k := z'_k - z_k$. Thus, we can represent the OOD trajectory by $\{x_k + s_k\}$ and $\{z_k + s'_k\}$. Furthermore, utilizing the virtual-feature method in Sec. 3.1, we have:

$$d(z'_{k-1}) = d(z_{k-1}) + \mathbf{J}'_{d,k-1}, \tag{4}$$

where $\mathbf{J}_{d,k-1}$ is a virtual Jacobian matrix of $d(\tilde{z}_{k-1})$. Due to equation 2 in Sec. 2.2, x'_k is updated by $x'_{k-1} - d(z'_{k-1})$ and x_k is updated by $x_{k-1} - d(z_{k-1})$. Based on equation 4, we have:

$$s_k = s_{k-1} - \mathbf{J}_{d,k-1} s'_{k-1}. \tag{5}$$

4. White-Box OOD Generalization Analysis

In this section, we rigorously demonstrate that the robustness of the L2O model is limited by its input features of NNs. We prove that increased features adversely impact the L2O model's generalization ability in OOD scenarios.

4.1. The Smooth Case

Building upon the state-of-the-art Math-L2O [14], we systematically detail our conclusions through a series of theorems and lemmas.

We analyze the convergence rate of the OOD scenario when the objective function $F(x)$ is smooth, i.e., $r(x) = 0$ and $F(x) = f(x)$. Leveraging Theorem 1 from [14], the update of the variable at the k-th iteration can be expressed as $x_k = x_{k-1} - \mathbf{P}_{k-1}\nabla f(x_{k-1}) - b_{k-1}$, where $\mathbf{P}_{k-1} \in \mathbb{R}^{n\times n}$ and $b_{k-1} \in \mathbb{R}^n$ are parameters learned by NNs.

Let $\mathbf{P}_{k-1}$ and b_{k-1} be $\mathbf{N}_1(\mathcal{Z}) \in \mathcal{D}_{C_1}(\mathcal{Z})$ and $\mathbf{N}_2(\mathcal{Z}) \in \mathcal{D}_{C_2}(\mathcal{Z})$ respectively, for some positive constants $C_1, C_2 \in \mathbb{R}^+$. As suggested in [14], we assign $\mathbf{P}_k$ as a diagonal matrix. Without loss of generality, for any given variable x_{k-1}, where $x_{k-1} \in \mathbb{R}^n$, and any given function $f \in \mathcal{F}_L(\mathbb{R}^n)$, we define $z_{k-1} = [x_{k-1}^\top, \nabla f(x_{k-1})^\top]^\top$ [14]. The update of variable x_k at each iteration k can then be expressed as:

$$x_k = x_{k-1} - \mathrm{diag}(\mathbf{N}_1(z_{k-1}))\nabla f(x_{k-1}) - \mathbf{N}_2(z_{k-1}). \tag{6}$$

The OOD shift applied to the variable and its gradient yields the definition of virtual feature (Sec. 3):

$$s'_{k-1} := [s_{k-1}^\top, (\nabla f'(x'_{k-1}) - \nabla f(x_{k-1}))^\top]^\top. \tag{7}$$

We present the following lemma for $\mathbf{N}_1(z)$ and $\mathbf{N}_2(z)$ to yield a variable x_k that is no worse than the previous variable x_{k-1} at each iteration k.

Lemma 1. *Denote the angle between $\mathbf{N}_2(z_{k-1})$ and corresponding $\nabla f(x_{k-1})$ as θ_{k-1}. For $\forall z_{k-1} \in \mathcal{Z}_P, \forall x_{k-1} \in \mathcal{S}_P$, if $\mathbf{N}_1(z_{k-1})$ and $\mathbf{N}_2(z_{k-1})$ are respectively bounded by following compact sets:*

$$\mathbf{N}_1(z_{k-1}) := \lambda_{k-1}\mathbf{1}, \lambda_{k-1} \in \left[0, \frac{1}{L}\right],$$
$$\mathbf{N}_2(z_{k-1}) \in \left[\mathbf{0}, \frac{\|\nabla f(x_{k-1})\|\cos(\theta_{k-1})}{L}\mathbf{1}\right], \theta \in \left[0, \frac{\pi}{2}\right],$$

then, for x_k generated by L2O model in equation 6, we have:

$$F(x_k) - F(x_{k-1}) \leq 0.$$

Proof. See Sec. 8.2 in Appendix. □

As stated in Lemma 1, to maintain homogeneous improvement on the convergence, it is sufficient to set $\mathbf{N}_1(z)$ as an input-invariant constant, and limit $\mathbf{N}_2(z)$ according to the gradient $\nabla F(x_{k-1})$. Moreover, we can utilize some bounded activation functions in training an L2O model to fulfill the conditions to ensure convergence, such as Sigmoid [17] and Tanh [12].

The proof for Lemma 1 establishes that improvement is characterized by a quadratic relation to each element in $\mathbf{N}_1(z_{k-1})$ and $\|\mathbf{N}_2(z_{k-1})\|$. We can identify the optimal upper bound for convergence improvement in the InD L2O model by optimizing this quadratic relation, leading us to Corollary 1.

Corollary 1. *For any $z_{k-1} \in \mathcal{Z}_P$, we let:*

$$\mathbf{N}_1(z_{k-1}) := \frac{1}{2L}\mathbf{1}, \mathbf{N}_2(z_{k-1}) := \frac{\nabla f(x_{k-1})}{2L},$$

the Math-L2O model in equation 6 is exactly gradient descent update with convergence rate:

$$F(x_K) - F(x^*) \leq \frac{L}{2K}\|x_0 - x^*\|^2.$$

Proof. See Sec. 8.3 in Appendix. □

Corollary 1 implies that the L2O model can achieve gradient descent's convergence rate by particular settings. The $\mathbf{N}_1(z_{k-1})$ is set to be a homogeneous constant across all elements. The $\mathbf{N}_2(z_{k-1})$ is set to in correspondence with the gradient $\nabla f(x_{k-1})$. Moreover, Corollary 1 also provides the most robust L2O model with an identical per iteration convergence gain among all InD instances.

Per-Iteration Convergence Gain

To ascertain the convergence rate of OOD, following Corollary 1, we suppose that after training, the following assumption holds for the *InD L2O Problem* (not for the *OOD L2O Problem*) to ensure best robustness for the InD scenario:

Assumption 1. *After training,* $\forall x_{k-1} \in \mathcal{S}_P, \forall z_{k-1} \in \mathcal{Z}_P$, $\mathbf{N}_1(z_{k-1}) := \frac{1}{2L}\mathbf{1}$ *and* $\mathbf{N}_2(z_{k-1}) := \frac{\nabla f(x_{k-1})}{2L}$.

Based on the Lemma 1 and Corollary 1, Assumption 1 leads to an L2O model with best robustness on all InD instances. In the following theorem, we quantify the diminution in convergence rate instigated by the virtual feature s' defined in Sec. 3.

Theorem 1. *Under Assumption 1, there exists virtual Jacobian matrices* $\mathbf{J}_{1,k-1}, \mathbf{J}_{2,k-1}, k = 1, 2, \ldots, K$ *that the per iteration convergence improvement in the OOD scenario is upper bounded by:*

$$\begin{aligned} & F'(x_k + s_k) - F'(x_{k-1} + s_{k-1}) \\ \leq & -\frac{\|\nabla f'(x_{k-1} + s_{k-1})\|^2}{2L} \\ & + L\|\operatorname{diag}(\mathbf{J}_{1,k-1}s')\nabla f'(x_{k-1} + s_{k-1})\|^2 \\ & + L\|\frac{\nabla f'(x_{k-1} + s_{k-1}) - \nabla f(x_{k-1})}{2L} - \mathbf{J}_{2,k-1}s'\|^2. \end{aligned}$$

Proof. See Sec. 8.4 in Appendix. □

Theorem 1 discloses that for a single iteration, the convergence improvement of OOD is bounded by the gradient descent with a step size of $1/L$, resulting in $-|\nabla f|^2/2L$ convergence improvement. Hence, when Math-L2O is adequately trained, any OOD will dampen convergence. Additionally, given that the expression on the right-hand side is not strictly non-positive, we cannot unequivocally affirm that convergence will transpire within a single iteration. Further investigation also intimates that, even in the context of convex optimization problems, scenarios may arise where the value of the objective function deteriorates.

While the existence of virtual Jacobian matrices in Theorem 1 is assured, their specific values remain unknown. Given that boundedness is a defined characteristic of these matrices, we relax this constraint in Theorem 1 and introduce Corollary 2.

Corollary 2. *Under Assumption 1, the per iteration convergence improvement in the OOD scenario can be upper bounded w.r.t.* $\|s'_{k-1}\|$ *by:*

$$\begin{aligned} & F'(x_k + s_k) - F'(x_{k-1} + s_{k-1}) \\ \leq & -\frac{\|\nabla f'(x_{k-1} + s_{k-1})\|^2}{2L} \\ & + \frac{\|\nabla f'(x_{k-1} + s_{k-1}) - \nabla f(x)\|^2}{2L} \\ & + \left(LC_1^2 n\|\nabla f'(x_{k-1} + s_{k-1})\|^2 + 2LC_2^2 n\right)\|s'\|^2. \end{aligned}$$

Proof. See Sec. 8.5 in Appendix. □

Corollary 2 further elucidates that the decline in the convergence improvement of OOD is determined by the magnitude of the input (virtual) feature s' of the L2O model, as outlined in equation 7. This magnitude is intrinsically related to the vector's dimensionality, which relies on the feature construction of the L2O model. For example, to reduce its magnitude, we can eliminate s_{k-1} in equation 7. We achieve this feature shrinking and propose a novel gradient-only L2O model in Sec. 5.

Multi-Iteration Convergence Rate

Building upon Theorem 1, we extrapolate the convergence rate across numerous iterations, as delineated in Theorem 2.

Theorem 2. *Under Assumption 1, the K iterations' convergence rate in the OOD scenario is upper bounded by:*

$$\begin{aligned} & \min_{k=1,\ldots,K} F'(x_k + s_k) - F'(x^* + s^*) \\ \leq & \frac{L}{2}\|x_0 - x^* + s_0 - s^*\|^2 - \frac{L}{2}\|x_K - x^* + s_K - s^*\|^2 \\ & + \frac{L}{K}\sum_{k=1}^{K}(x_k + s_k - x^* - s^*)^\top \\ & \left(x_k + s_k - \left(x_{k-1} + s_{k-1} - \frac{\nabla f'(x_{k-1} + s_{k-1})}{L}\right)\right). \end{aligned}$$

Proof. See Sec. 8.6 in Appendix. □

The first two terms on the right-hand side of the above inequality represent the gradient descent convergence rate characterized by a step size of $1/L$. However, the third term is unbounded and could be either non-positive or positive. This suggests that there is no guaranteed global convergence in OOD situations, even with homogeneous robustness in InD scenarios.

The inequation above offers a direct approach to analyzing distinct cases of convergence. Included in the concluding line of Theorem 1 is a gradient descent equation, $x_{k-1} + s_{k-1} - \nabla f'(x_{k-1} + s_{k-1})/L$. Moreover, $x_k + s_k$ represents the updated solution by the L2O model. The subtraction of the two terms reveals the discrepancy between the updates made by L2O and gradient descent on the objective variable $x_{k-1} + s_{k-1}$, thereby creating a vector directed towards $x_k + s_k$. Similarly, $x_k + s_k - x^* - s^*$ signifies the relative position to the optimal solution, generating another vector directed towards $x_k + s_k$. The resulting inner product will be non-positive if the angle between these two vectors is $\pi/2$ or more. Moreover, if the trajectory of $x_k + s_k - x^* - s^*$ can be extrapolated from domain knowledge, a "trust region" surrounding $x_k + s_k$ can be established to augment the efficacy of gradient descent.

From Theorem 1, we develop a stringent formulation to illustrate the potential uncertainty of convergence in OOD

scenarios. If we know the relative position of the optimal solution, we can fine-tune an L2O model to outperform gradient descent. Based on Theorem 1, we establish an upper bound w.r.t. s'. This mirrors the approach in Corollary 2.

Corollary 3. *Under Assumption 1, L2O model $d(z)$'s OOD convergence rate is upper bounded w.r.t. $\|s'_{k-1}\|$ by:*

$$\begin{aligned}
&\min_{k=1,\dots,K} F'(x_k+s_k)-F'(x^*+s^*)\\
\leq&\frac{L}{2}\|x_0+s_0-x^*-s^*\|^2-\frac{L}{2}\|x_K+s_K-x^*-s^*\|^2\\
&+\frac{1}{2K}\sum_{k=1}^{K}\left(\nabla f'(x_{k-1}+s_{k-1})-\nabla f(x_{k-1})\right)^\top\\
&\qquad(x_k+s_k-x^*-s^*)\\
&+\frac{L}{K}\sum_{k=1}^{K}\left(C_1\sqrt{n}\|\nabla f'(x_{k-1}+s_{k-1})\|\right.\\
&\qquad\left.+C_2\sqrt{n}\|x_k+s_k-x^*-s^*\|\right)\|s'_{k-1}\|.
\end{aligned}$$

Proof. See Sec. 8.7 in Appendix. □

Corollary 3 posits that the overall convergence rate is consistently upper bounded by the magnitude of s'. Based on Corollaries 2 and 3, we endeavor to reduce the magnitude of s' by eliminating variable, leading to the approach of a gradient-only Math-L2O framework in the next section.

4.2. Other Three Cases

We have developed several additional theorems and lemmas for non-smooth, incremental historical modeling, and integrated smooth-non-smooth cases.. Our approach mirrors that employed in the smooth case demonstration. The backbone algorithms of math-inspired L2O fundamentally limit their convergences. For example, the Gradient Descent [19] and Proximal Point [18] algorithms in the smooth case and the non-smooth case, respectively.

We extend the theorems and lemmas in the smooth case to derive formulas for convergence improvement of a single iteration and convergence rate across a sequence. These demonstrate the diminishing effect of OOD on convergence. Our findings conclude that constructing fewer features can mitigate this negative impact. More extensive demonstrations and complete proofs can be found in Appendix.

5. Gradient-Only L2O Model

Informed by the theorems and lemmas posited in Sec. 4, we introduce a gradient-only L2O model, GO-Math-L2O, which aims to enhance robustness in OOD scenarios by eliminate variable-related input features for the L2O model.

To derive the formulation of GO-Math-L2O, we employ the workflow delineated in [14]. Let T denote the history length. At the k-th iteration, suppose there exists an operator $d_k \in \mathcal{D}_C(\mathbb{R}^{3n})$, we formulate the input of our GO-Math-L2O as follows:

$$x_k = x_{k-1} - d_k(\nabla f(x_{k-1}), g_k, v_{k-1}), \tag{8}$$

where g_k denotes the implicit subgradient vector of x_k to invoke the proximal gradient method [14]. Moreover, we eliminate all variable-related features and define v_k as the result of historical modeling [14]. Different from the variable approach in [14], we propose to utilize gradient (and subgradient) to model the historical information of the optimization process since gradient sufficiently and necessarily indicates optimality in convex optimization scenarios. Such an approach reduces the magnitude of L2O's input feature (defined in Sec. 3) by $1/3$, which facilitates convergence based on our proposed corollaries in Sec. 4.

Suppose there exists an operator $u_k \in \mathcal{D}_C(\mathbb{R}^{Tn})$, we define the following model to generate v_k from the gradient and subgradient of T historical iterations:

$$v_k = d_k(\nabla f(x_{k-1})+g_{k-1},\dots,\nabla f(x_{k-T})+g_{k-T}). \tag{9}$$

where each g represents a subgradient vector. For subgradient selection, we should carefully choose an instance from the subgradient set of each non-smooth point since an arbitrary selection may lead to poor convergence [25].

We achieve a lightweight subgradient selection based on the gradient map method [28] and our following model constructions. From the objective definition in Sec. 2.1, the non-smooth objective r is trivially solvable by $\arg\min$. Thus, at k-th iteration, we can recover an implicit subgradient vector g_k of $\arg\min$ by k-th solution x_k and k_{-1}-th solution x_{k-1} if the L2O operator d_k in equation 8 is inversible. Next, we achieve an inversible d_k based on the workflow proposed in [14].

With the above feature and component constructions, we start to define the structures and learnable parameters of our L2O operator d_k in equation 8. We formulate d_k as the necessary condition of convergence [14], which means the formulation that d_k should follow if convergence is achieved. First, denote a candidate optimal solution as x^*, we construct two sufficient conditions (Asymptotic **F**ixed **P**oint and **G**lobal **C**onvergence) of convergence for our L2O operator d_k in equation 8:

$$\lim_{k\to\infty} d_k(\nabla f(x^*), -\nabla f(x^*), 0) = \mathbf{0}, \tag{FP}$$

$$\lim_{k\to\infty} x_k = x^*. \tag{GC}$$

As discussed in [14], such two conditions are essential for optimization algorithms.

Then, we present the following Theorem 3 to construct d_k's parameters. Theorem 3 shows that if d_k converges, it should be in the form of equation 10. Then, if we add a further assumption on some of the parameters, the solution on each iteration can be uniquely obtained by equation 11.

Theorem 3. *Suppose $T = 2$, given $f \in \mathcal{F}_L(\mathbb{R}^n)$ and $r \in \mathcal{F}(\mathbb{R}^n)$, we pick an operators from $\mathcal{D}_C(\mathbb{R}^{3n})$ and $\mathcal{D}_C(\mathbb{R}^{2n})$. If Condition FP and Condition GC hold, there exist $\mathbf{R}_k \succ 0, \mathbf{Q}_k, \mathbf{B}_k \in \mathbb{R}^{n\times n}$ and $b_{1,k}, b_{2,k} \in \mathbb{R}^n$ and satisfying:*

$$\begin{aligned} x_k =& x_{k-1} - \mathbf{R}_k \nabla f(x_{k-1}) - \mathbf{R}_k g_k - \mathbf{Q}_k v_{k-1} - b_{1,k}, \\ v_k =& (\mathbf{I} - \mathbf{B}_k) G_k + \mathbf{B}_k G_{k-1} - b_{2,k}, \\ G_k :=& \mathbf{R}_k^{-1}(x_{k-1} - x_k - \mathbf{Q}_k v_{k-1} - b_{1,k}), \end{aligned} \tag{10}$$

where for $k = 0, 1, 2, \ldots$, $g_{k+1} \in \partial r(x_{k+1})$ represents implicit subgradient vector, $\mathbf{R}_k$, $\mathbf{Q}_k$, and $\mathbf{B}_k$ are bounded parameter matrices and $b_{1,k} \to 0, b_{2,k} \to 0$ as $k \to \infty$. Since $\mathbf{R}_k$ is symmetric positive definite, x_{k+1} is uniquely determined through:

$$\arg\min_{x\in\mathbb{R}^n} r(x) + \frac{1}{2}\|x - \mathbf{R}_k \nabla f(x_k) - \mathbf{Q}_k v_k - b_{1,k}\|^2_{\mathbf{R}_k^{-1}}, \tag{11}$$

where $\|\cdot\|_{\mathbf{R}_k^{-1}}$ is defined as $\|x\|_{\mathbf{R}_k^{-1}} = \sqrt{x^\top \mathbf{R}_k^{-1} x}$.

Proof. See Sec. 8.8 in Appendix. □

As a necessary condition for convergence, Theorem 3 suggests that our gradient-only L2O model should construct parameters $\mathbf{R}$, $\mathbf{Q}$, $\mathbf{B}$, b_1, and b_2. It is worth noting that this model does not guarantee satisfaction of conditions FP and GC. The convergence is promoted by training.

We learn to construct the parameters in Theorem 3. First, the proof elucidates that the bias terms approach zero upon convergence. Thus, we set $b_1, b_2 := 0$ and learn to construct $\mathbf{R}$, $\mathbf{Q}$, and $\mathbf{B}$. We take the construction in [14] to implement our GO-Math-L2O model with a two-layer LSTM cell. Then, we utilize three one-layer linear neural network models with Sigmoid activation function [17] to generate $\mathbf{R}$, $\mathbf{Q}$, and $\mathbf{B}$ at each iteration, respectively, which ensures that all the matrices are bounded.

6. Experiments

We perform experiments with Python 3.9 and PyTorch 1.12 on an Ubuntu 18.04 system equipped with 128GB of memory, an Intel Xeon Gold 5320 CPU, and a pair of NVIDIA RTX 3090 GPUs. We strictly follow the experimental setup presented in [14] for constructing InD evaluations. Due to the page limit, the implementation details are in Sec. 12.

We use the Adam optimizer [13] to train our proposed model and learning-based baselines on datasets of 32,000 optimization problems with randomly sampled parameters and optimal solutions. We generate a test dataset of 1,000 iterations' objective values, averaging over 1,024 pre-generated optimization problems. We evaluate different training configurations and loss functions to select the best setting. Details are in Sec. 12.5, Appendix.

Baselines. We compare our GD-Math-L2O (Section 5) against both learning-based methods and non-learning algorithms. Our main competater is the state-of-the-art (SOTA) math-inspired L2O model in [14]. Specifically, we select the best variant from this study, L2O-PA. Consistent with the outlined methodology, we also compare our approach with several hand-crafted algorithms: ISTA, FISTA [5], Adam [13], and AdamHD [4], which is Adam complemented by an adaptive learning rate. Moreover, we assess our model against two black-box L2O models, namely L2O-DM[3] and L2O-RNNprop [15], and one Ada-LISTA [1] that unrolls the gradient descent algorithm with learning.

Optimization Objective. We choose the two regression problems in [14]: *LASSO Regression* and *Logistic Regression*, defined as follows:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} F(x) =& \frac{1}{2}\|\mathbf{A}x - b\|^2 + \lambda\|x\|_1, \\ \min_{x\in\mathbb{R}^n} F(x) =& -\frac{1}{m}\sum_{i=1}^{m}\big[b_i \log(h(a_i^\top x)) \\ &+ (1 - b_i)\log(1 - h(a_i^\top x))\big] + \lambda\|x\|_1, \end{aligned}$$

where $m := 1000$. $\mathbf{A} \in \mathbb{R}^{250\times 500}$ and $b \in \mathbb{R}^{500}$, $\{(a_i, b_i) \in \mathbb{R}^{50} \times \{0, 1\}\}_{i=1}^m$ are given parameters. $h(x) := 1/(1 + e^{-x})$ is sigmoid function. We utilize the standard normal distribution to generate samples and set $\lambda := 0.1$ for both scenarios [14].

We implement the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [5], executing 5,000 iterations to generate labels (optimal objective values) [14]. Due to page limit, we confine our presentation to *LASSO Regression*. The results of *Logistic Regression* are in Sec. 12.8, Appendix.

OOD Scenarios. We aim to quantify the effect of OOD on convergence rates. We specifically formulate two types of OOD trajectories triggered by different actions. It is crucial to note that both OOD and InD scenarios maintain an identical optimality on both objective and solution.

1) $s_0 \neq 0, s_0 \in \mathbb{R}^n$. x_0 is altered by an adjustment factor s_0 that x_0' falls within the OOD set $\mathcal{S}_O$. Assuming the objective remains consistent, we expect x' to move from the OOD $\mathcal{S}_O$ to the InD $\mathcal{S}_P$.
2) $F'(x) = F(x + t), t \in \mathbb{R}^n$. The OOD perturbation introduces a translation t along the axes of the objective variable to the objective function. Thus, the optimal solution x'^* diverges from that obtained under the original InD domain, even though the optimal value remains. This illustrates a scenario where the domain translates in inference. If the starting point is unchanged, x' is expected to move from InD domain to OOD domain.

We derive the non-smooth function's proximal operator for the OOD scenario, specifically for the ℓ_1-norm. We define $r(x)$ as $\lambda|x|_1$, and define the OOD translation as t on

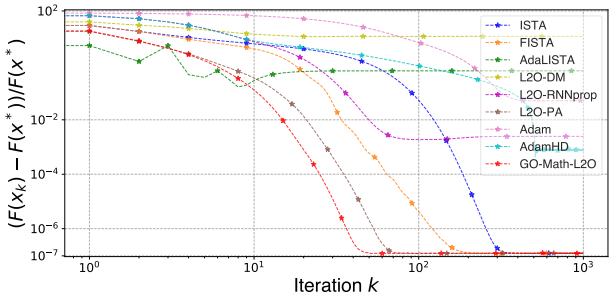

Figure 1. LASSO Regression: InD.

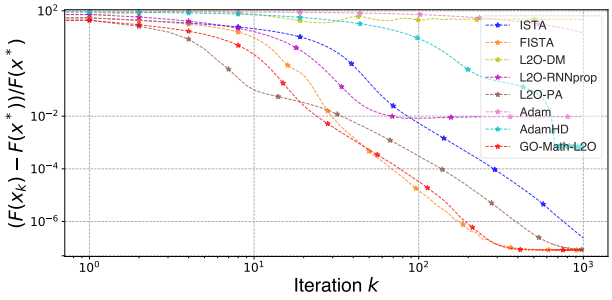

Figure 2. LASSO Regression: Real-World OOD.

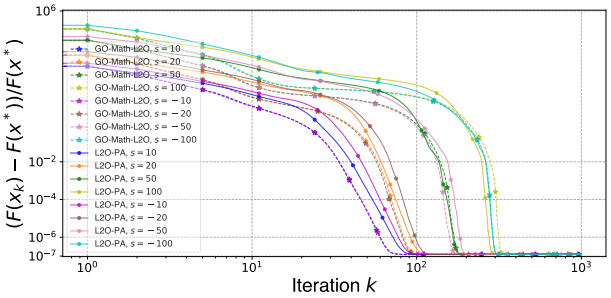

Figure 3. LASSO Regression: OOD by Trigger 1.

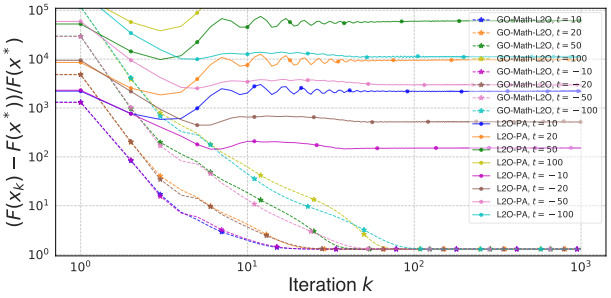

Figure 4. LASSO Regression: OOD by Trigger 2.

variable. The OOD proximal operator with t is given by:

$$(\operatorname{prox}_{r,p_k}(\bar{x}))_i := -t + \operatorname{sign}(\bar{x}_i)\max(0, |\bar{x}_i| - \lambda(p_k)_i + \operatorname{sign}(\bar{x}_i)t).$$

6.1. InD Comparison

The trajectories of solving the *LASSO Regression* problems are shown in Figure 1, where the vertical axis represents the normed objective value at a given iteration (indicated on the horizontal axis) with a label generated by FISTA [5]. Our proposed method (red line) surpasses all other methods, demonstrating better optimality and quicker convergence.

Furthermore, we utilize several ablation studies on model configuration, such as gradient map recovery strategies in Sec. 12.4 and hyperparameter settings for learned parameter matrices in Sec. 12.6, to determine the best model configuration. The details are in the Appendix.

6.2. OOD Comparison

The real-world results in Firgure 2 show that our GO-Math-L2O (converges at 400 iterations) outperforms all other baselines (1,000 iterations). Considering the lackluster performances of other baselines in Figures 1 and 2, we primarily compare our GO-Math-L2O model against SOTA L2O-PA [14]. We construct two synthetic OOD scenarios with the two trigger settings, where the optimal objectives align with those in Figure 1.

Figure 3 portrays the scenario wherein the initial point shifts such that $s_0 \neq 0$, with the legends denoting sixteen cases. Our GO-Math-L2O model (represented by dashed lines) outshines L2O-PA (solid lines) in all instances, asserting its superior robustness.

The observations in Figure 4 for the OOD scenario involve function shifting that $F'(x) = F(x + t)$. The optimal values achieved by both methods deteriorate from 10^{-7} (as seen in Figure 3) to 10^0. However, our GO-Math-L2O still outperforms L2O-PA in all cases. For example, when $t = \pm 10$, our model converges at around 20 steps, but L2O-PA fails to converge.

7. Conclusion

This paper aims to improve the robustness of L2O in OOD scenarios. We derive a general condition to ensure robustness in InD scenarios. We propose virtual features to connect the OOD L2O's outputs with InD L2O's outputs of a whole trajectory. Based on such connections, we prove formulations to demonstrate the convergence performances in OOD scenarios. Based on the observations, we establish that the magnitude of the L2O model's input features intrinsically limits the OOD's convergence. Furthermore, we propose a robust L2O model with concise gradient-only features and modeling historical features with gradient and subgradient. Experiments show our model significantly outperforms SOTA baselines.

Acknowledgements

This work is partly supported by funding from the Research Grants Council of Hong Kong (11209520, CRF C7004-22G) and CUHK (4055199).

References

[1] Aviad Aberdam, Alona Golts, and Michael Elad. Ada-lista: Learned solvers adaptive to varying models. *IEEE TPAMI*, 44(12):9222–9235, 2021. 7

[2] Diogo Almeida, Clemens Winter, Jie Tang, and Wojciech Zaremba. A generalizable approach to learning optimizers. *arXiv preprint arXiv:2106.00958*, 2021. 1

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016. 7

[4] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017. 7

[5] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. 7, 8, 16, 29, 45, 49

[6] Yanmei Cao, Guomei Zhang, Guobing Li, and Jia Zhang. A Deep Q-Network Based-Resource Allocation Scheme for Massive MIMO-NOMA. *IEEE Communications Letters*, 25 (5):1544–1548, 2021. 1

[7] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Zhangyang Wang, Howard Heaton, Jialin Liu, and Wotao Yin. Learning to optimize: A primer and a benchmark. *The Journal of Machine Learning Research*, 23(1):8562–8620, 2022. 1

[8] Karol Gregor and Yann LeCun. Learning Fast Approximations of Sparse Coding. In *ICML*, pages 399–406, 2010. 1

[9] Howard Heaton, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Safeguarded learned convex optimization. In *AAAI*, pages 7848–7855, 2023. 1

[10] Michael (https://math.stackexchange.com/users/155065/michael). Proof of convergence for the proximal point algorithm. Mathematics Stack Exchange, 2015. URL:https://math.stackexchange.com/q/1303325 (version: 2015-05-30). 14, 17

[11] Qiyu Hu, Yunlong Cai, Qingjiang Shi, Kaidi Xu, Guanding Yu, and Zhi Ding. Iterative Algorithm Induced Deep-Unfolding Neural Networks: Precoding Design for Multiuser MIMO Systems. *IEEE TWC*, 20(2):1394–1410, 2020. 1

[12] B.L. Kalman and S.C. Kwasny. Why tanh: choosing a sigmoidal function. In *IJCNN International Joint Conference on Neural Networks*, pages 578–581 vol.4, 1992. 4

[13] D Kinga, Jimmy Ba Adam, et al. A method for stochastic optimization. In *ICLR*, page 6. San Diego, California;, 2015. 7, 50

[14] Jialin Liu, Xiaohan Chen, Zhangyang Wang, Wotao Yin, and HanQin Cai. Towards Constituting Mathematical Structures for Learning to Optimize. In *ICML*, 2023. 1, 2, 3, 4, 6, 7, 8, 14, 15, 16, 17, 29, 30, 31, 48, 49, 50, 52, 53, 54, 55

[15] Kaifeng Lv, Shunhua Jiang, and Jian Li. Learning gradient descent: Better generalization and longer horizons. In *ICML*, pages 2247–2255. PMLR, 2017. 7, 29

[16] Kaifeng Lv, Shunhua Jiang, and Jian Li. Learning gradient descent: Better generalization and longer horizons. In *ICML*, pages 2247–2255. PMLR, 2017. 1

[17] Sridhar Narayan. The generalized sigmoid activation function: Competitive supervised learning. *Information Sciences*, 99(1):69–82, 1997. 4, 7

[18] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976. 6, 17

[19] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. 1, 3, 6

[20] W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 1976. 3

[21] Lukas Schynol and Marius Pesavento. Coordinated Sum-Rate Maximization in Multicell MU-MIMO With Deep Unrolling. *IEEE JSAC*, 41(4):1120–1134, 2023. 1

[22] Yifei Shen, Yuanming Shi, Jun Zhang, and Khaled B. Letaief. Graph Neural Networks for Scalable Radio Resource Management: Architecture Design and Theoretical Analysis. *IEEE JSAC*, 39(1):101–115, 2021. 1

[23] Z Shen, J Liu, Y He, X Zhang, R Xu, H Yu, and P Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2023. 1

[24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. 2012. 1

[25] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient Methods. *Stanford EE392o Optimization Projects*, 2003. 6

[26] Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nicholas D Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE TSP*, 66(20):5438–5453, 2018. 1

[27] Ryan Tibshirani. Lecture 6: September 12. *CMU 10-725: Optimization*, 2013. 2, 3

[28] L. Vandenberghe. Proximal gradient method. *ECE236C (Spring 2022)*, 2022. 6, 17

[29] Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *ICML*, pages 3751–3760. PMLR, 2017. 1

[30] Junjie Yang, Tianlong Chen, Mingkang Zhu, Fengxiang He, Dacheng Tao, Yingbin Liang, and Zhangyang Wang. Learning to Generalize Provably in Learning to Optimize. In *International Conference on Artificial Intelligence and Statistics*, pages 9807–9825, 2023. 1

[31] Yu Zhao, Ignas G. Niemegeers, and Sonia M.Heemstra De Groot. Dynamic Power Allocation for Cell-Free Massive MIMO: Deep Reinforcement Learning Methods. *IEEE Access*, 9:102953–102965, 2021. 1

[32] Xingyu Zhou. On the fenchel duality between strong convexity and lipschitz continuous gradient. *arXiv preprint arXiv:1803.06573*, 2018. 17

[33] Minghe Zhu, Tsung-Hui Chang, and Mingyi Hong. Learning to beamform in heterogeneous massive MIMO networks. *IEEE TWC*, 2022. 1