On Solution Functions of Optimization: Universal Approximation and Covering Number Bounds

Ming Jin, Vanshaj Khattar, Harshal Kaushik, Bilgehan Sel, and Ruoxi Jia

Electrical and Computer Engineering, Virginia Tech
jinming@vt.edu, vanshajk@vt.edu, harshaldkaushik@vt.edu, bsel@vt.edu, ruoxijia@vt.edu

Abstract

We study the expressibility and learnability of convex optimization solution functions and their multi-layer architectural extension. The main results are: *(1)* the class of solution functions of linear programming (LP) and quadratic programming (QP) is a universal approximant for the smooth model class or some restricted Sobolev space, and we characterize the rate-distortion, *(2)* the approximation power is investigated through a viewpoint of regression error, where information about the target function is provided in terms of data observations, *(3)* compositionality in the form of a deep architecture with optimization as a layer is shown to reconstruct some basic functions used in numerical analysis without error, which implies that *(4)* a substantial reduction in rate-distortion can be achieved with a universal network architecture, and *(5)* we discuss the statistical bounds of empirical covering numbers for LP/QP, as well as a generic optimization problem (possibly nonconvex) by exploiting tame geometry. Our results provide the *first rigorous analysis of the approximation and learning-theoretic properties of solution functions* with implications for algorithmic design and performance guarantees.

1 Introduction

We study the object referred to as *solution function* defined by the following generic optimization:

$$\pi(x,\theta) = \underset{z\in R(x,\theta)}{\arg\min}\ \ g(z;x,\theta), \tag{1}$$

where $g(\cdot;x,\theta):\mathbb{R}^{n_z}\to\mathbb{R}$ and $R(x,\theta)\subseteq\mathbb{R}^{n_z}$ are the objective function and feasible set (with $R:\mathbb{R}^{n_x}\times\mathbb{R}^{n_\theta}\rightrightarrows\mathbb{R}^{n_z}$ being a set-valued function), respectively, characterized by both $x\in\mathbb{R}^{n_x}$ and $\theta\in\Theta\subseteq\mathbb{R}^{n_\theta}$. We use semicolon in $g(z;x,\theta)$ to separate optimization variables from parameters. To make a further distinction, in the context of decision making, x can be the input/state, θ is the parameter, and the output is the decision/action. Since the optimization solution can be a set, we make proper assumptions to ensure uniqueness (Dontchev and Rockafellar 2009).

Historically, the solution function (and optimal value function) has been an important basis for local sensitivity/stability and parametric analysis in optimization theory (Dontchev and Rockafellar 2009; Fiacco 2020); see (Amos 2022) for renewed interests. However, to the best of the authors' knowledge, a mathematical theory characterizing the global properties of solution functions is still missing. Two of the basic questions are:

(1) which classes of functions can they approximate well?

(2) what is the statistical complexity of the class of solution functions?

The first question pertains to the expressivity of the function class. Without proper restrictions, one can easily obtain a construct with $g(z;x,\theta):=\|z-\mu(x,\theta)\|$ and $R(x,\theta):=\mathbb{R}^{n_z}$ so that $\pi(x,\theta)$ can represent any function $\mu(x,\theta)$ despite being nonconvex or even discontinuous; however, such construct is neither interesting nor practically relevant. To prevent such degenerate cases, the optimization in (1) is, in general, assumed to be convex. In fact, as we show later in the analysis, further restrictions to LPs or QPs can still preserve the universal approximation property. Perhaps an even more intriguing and related question is concerning the role of "depth" by drawing an analogy to contemporary studies on deep neural networks (DNNs) (see, e.g., (Hanin 2019; Lu et al. 2021)). Indeed, the idea of concatenating optimizations as "layers" seems to be catching on (Amos and Kolter 2017; Agrawal et al. 2019; Kotary et al. 2021). Beyond the current knowledge in the multi-parametric programming literature that the solution function of an LP/QP is piecewise affine (PWA) (Grancharova and Johansen 2012), we provide a simple construction with two layers of LPs/QPs that can reproduce nonlinear functions such as product operator with no error; repeated stacking such structures by adding depth can thus reconstruct any polynomial functions of arbitrary order. Such blessings of compositionality appear in a very different form than DNNs, and will be exploited to reduce the complexity of construction (measured in terms of the number of variables and constraints). A key issue in training multi-layer compositions is the ability to backpropagate, for which we ensure compliance with the disciplined parametric programming (DPP) rules introduced in (Agrawal et al. 2018). Hence, in the first part of the study, we examine the expressivity and role of depth from the perspective of approximation theory (DeVore and Lorentz 1993). The second question has a bearing on learnability, which, ironically, has not been well-established despite recent advancements in related fields (Amos 2022; Kotary et al. 2021; Ebert et al. 2021;

Hewing et al. 2020). We provide partial answers by focusing on the notion of covering numbers, which is fundamental to furnishing generalization error bounds and characterizing sample complexity (Cucker and Zhou 2007).

1.1 Why Should We Study Solution Functions?

Optimization is crucial in modeling complex phenomena and decision-making processes with a wide range of real-world applications (Boyd, Boyd, and Vandenberghe 2004). In the following, we contextualize this study by connecting to adjacent problems in machine learning, control, and operations research.

Bi-level formulations of decision-making. In machine learning, a lot of complex problems with a hierarchical structure are amenable to a bi-level formulation (Liu et al. 2021), where the inner-level solution function may correspond to a learned model (in hyperparameter optimization (Lorraine, Vicol, and Duvenaud 2020)), task-adaptive features (in multi-task and meta-learning (Hospedales et al. 2020)), attacked model (in adversarial learning (Zeng et al. 2022)), or critic network (in reinforcement learning (Hong et al. 2020)).

Inverse problems. There are also a variety of problems with an *inverse* nature, where the decisions are taken as input, and the goal is to infer an objective and/or constraints that render these decisions approximately or exactly optimal (Adams, Cody, and Beling 2022); therein, the solution function of the corresponding optimization represents some (near-)optimal policies (in inverse reinforcement learning (Adams, Cody, and Beling 2022)), optimal controller (in inverse control (Ab Azar, Shahmansoorian, and Davoudi 2020)), and Nash equilibrium of noncooperative agents (in inverse game theory (Bertsimas, Gupta, and Paschalidis 2015; Jia et al. 2018)), and identifying the parameters of the optimization is tasked to infer the hidden reward function or utility functions.

End-to-end optimization. The solution function can be used directly as a predictor or control policy. In decision-focused learning (Wilder, Dilkina, and Tambe 2019; Feber et al. 2020), smart predict-then-optimize (Elmachtoub and Grigas 2020; Loke, Tang, and Xiao 2021), and end-to-end optimization learning (Kotary et al. 2021), a constrained optimization model is integrated into the decision-making pipeline to create a hybrid architecture, where the parameters of the solution function are trained in an end-to-end fashion (often through implicit gradients (Agrawal et al. 2019)). Such approaches have been demonstrated for stochastic programming (Donti, Amos, and Kolter 2017), combinatorial optimization (Wilder, Dilkina, and Tambe 2019), and reinforcement learning (Wang et al. 2021), with various applications in operations research (e.g., vehicle routing, inventory management, and portfolio optimization) (Elmachtoub and Grigas 2020), and hold the promise to enable structural inference and decision-making under constraints.

Model-predictive control. The solution function has a long tradition being used as a policy in model-predictive control (MPC) (Grancharova and Johansen 2012); recent advancements in learning-based MPC aim to infer the parameterization of the MPC policy, i.e., the cost and constraints, that lead to better closed-loop performance and account for safety specifications (Hewing et al. 2020).

None of the above problems can be satisfactorily understood or solved with existing theories (Dontchev and Rockafellar 2009; Fiacco 2020), revealing a fundamental need to study the approximation and statistical properties beyond local perturbations.

1.2 Contributions

Key contributions are summarized below:

- We develop a new perspective on approximation through the lens of regression with a fixed design, and establish a universal approximation theorem of the solution functions of LPs with constructive proof. The complexity of the construction is analyzed in terms of the total number of variables and constraints to obtain an ϵ accuracy (i.e., rate-distortion) (Theorem 1).
- Illuminate the role of depth. Compositionality in the form of multi-layer LPs/QPs is shown to reconstruct polynomials without error (Lemma 2 in appendix). We characterize complexity with the additional depth measure and show a substantial reduction in complexity with a universal network architecture in approximating all smooth functions in some restricted Sobolev space (Theorem 2).
- We discuss statistical bounds using empirical covering numbers. For LPs/QPs, we provide bounds that depend on the number of constraints and some condition number associated with constraints (Theorem 4). For a generic convex optimization problem, we crucially leverage the development in tame geometry by showing that the solution map is Whitney stratifiable (Theorem 5). Our proof technique is broadly applicable to piecewise smooth functions with a bounded number of pieces. The result also has direct implications for provable convergence guarantees when training these functions with subgradient descent.

1.3 Related Work

Deep architecture with optimization as a layer. Inspired by the remarkable effectiveness of DNNs, a line of work considers architectures with differentiable optimization as a layer (Amos and Kolter 2017; Agrawal et al. 2019; Kotary et al. 2021). Since conventional activation functions such as ReLU and max pooling can be reconstructed as LP solution functions, such an architecture can capture more complex structures and richer behaviors (Amos and Kolter 2017). However, a systematic study of approximation or statistical complexity is lacking in the literature.

Approximation and learning theory for DNNs. The universal approximation capacity of neural networks has been well-known (Hornik, Stinchcombe, and White 1989); for instance, to achieve ϵ approximation accuracy of a C^k smooth function with input dimension n_x, one needs $\mathcal{O}(\epsilon^{n_x/k})$ number of neurons (Pinkus 1999). But that alone does not explain why neural networks are so effective in practice, since functions such as polynomials, splines, and wavelets also produce universal approximants. Recent papers aim to elucidate this matter, with a particular focus on the role of depth (Allen-Zhu and Li 2020; Lu et al. 2021). The role of depth has also been examined from the perspective of approximation theory (Yarotsky 2018; Chen et al. 2019), along with various other

measures such as the number of linear regions (Serra, Tjandraatmadja, and Ramalingam 2018) and Betti numbers (Bianchini and Scarsell 2014) (see (DeVore, Hanin, and Petrova 2021) for a recent survey). We also mention a very general approach to expressiveness, in the context of approximation, named the method of nonlinear widths (DeVore, Howard, and Micchelli. 1989). Existing results on statistical complexity of DNNs include bounds for the Vapnik-Chervonenkis (VC) dimension (Bartlett, Maiorov, and Meir 1998; Anthony and Bartlett 1999) and fat-shattering dimension (Anthony and Bartlett 1999), with some recent developments on tighter characterizations (Bartlett et al. 2019). The present study can be seen as parallel development for the solution function and its multi-layer architecture.

Explicit MPC and inverse optimality. Explicit MPC exploits multiparametric programming techniques to compute the solution function offline and has been investigated for LP/QP, nonlinear convex programming, and mixed-integer programming (Grancharova and Johansen 2012). Another closely related topic studied in the control community is inverse MPC, a.k.a., inverse parametric programming, which aims to construct an optimization such that its optimal solution is equivalent to a given function (Baes, Diehl, and Necoara 2008). This inverse optimality problem has led to interesting results for general nonlinear continuous functions (Baes, Diehl, and Necoara 2008) and more recently for continuous PWA functions based on techniques such as difference-of-convex (DC) decomposition (Hempel, Goulart, and Lygeros 2014) and convex lifting (Nguyen et al. 2018). When the target function is only accessible through samples, inverse optimization can be applied to determine an optimization model that renders the set of sampled decisions approximately or exactly optimal (see, e.g., (Ebert et al. 2021) for a recent survey). Despite these developments, the questions of approximation or estimation errors of solution functions have largely eluded attention.

Max-affine and PWA regression. Max-affine regression, originated in (Hildreth 1957), aims to recover an optimal piecewise linear approximant to a convex function through either parametric (Magnani and Boyd 2009; Hannah and Dunson 2013; Ghosh, Pananjady, and A. Guntuboyia 2019) or nonparametric regression (Hildreth 1957; Balázs, György, and Szepesvári 2015; Balázs 2022). Recent studies provide theoretical guarantees on near-optimal minimax rate (Ghosh, Pananjady, and A. Guntuboyia 2019; Balázs, György, and Szepesvári 2015) and adaptive partitioning (Balázs 2022). PWA regression generalizes the function class to nonconvex candidates; an affine fit is computed separately for each partition of the space, which can be either predefined (Toriello and Vielma 2015) or adaptively determined (Siahkamari et al. 2000). These two lines of work are closely linked through DC modeling (Bačák and Borwein 2011).

2 Preliminaries

2.1 Model Class Assumptions

We consider the uniform error as $\|f - \tilde{f}\|_\infty = \max_{x\in[0,1]^{n_x}} |f(x) - \tilde{f}(x)|$. To provide a meaningful discussion of the approximation rate, we state the assumptions on the function class (commonly referred to as model class assumptions). Consider a Sobolev space $\mathcal{W}^{k,\infty}([0,1]^{n_x})$ defined as the space of functions on $[0,1]^{n_x}$ with derivatives up to order k. The norm in $\mathcal{W}^{k,\infty}([0,1]^{n_x})$ is defined as $\|h\|_{\mathcal{W}^{k,\infty}([0,1]^{n_x})} = \max_{\mathbf{k}:|\mathbf{k}|\le k} \operatorname*{ess\,sup}_{z\in[0,1]^{n_x}} |D^{\mathbf{k}}h(z)|$, here $\mathbf{k} \in \{0,1,\dots\}^{n_x}$, $|\mathbf{k}| = \sum_{i=1}^{n_x} k_i$, and $D^{\mathbf{k}} \coloneqq \frac{\partial^{|\mathbf{k}|}}{\partial x_1^{k_1}\cdots\partial x_{n_x}^{k_{n_x}}}$ is the standard derivative operator. A restrictive subclass of functions $F_{k,n_x} = \left\{h \in \mathcal{W}^k([0,1]^{n_x}) : \|h\|_{\mathcal{W}^k([0,1]^{n_x})} \le 1\right\}$ can be considered as a unit ball in $\mathcal{W}^{k,\infty}([0,1]^{n_x})$ consisting of the functions with all their derivatives up to order k bounded by unity. In addition, the class $C^k([0,1]^{n_x})$ consists of functions continuously differentiable up to order k. Without loss of generality, we assume the input space $X \coloneqq [0,1]^{n_x}$ and omit its dependence in the above definitions. We use $\|\cdot\|$ for the standard Euclidean norm.

2.2 Disciplined Parametrized Programming

The DPP rule, as a subset of Disciplined Convex Programming (DCP), places mild restrictions on how parameters can enter expressions. We briefly describe the DPP rule and refer the reader to (Agrawal et al. 2019, Sec. 4.1) for more details.

Let us begin with some basic terminologies. We refer to x and θ in (1) as *parameters*, which, once instantiated with values, are treated as constants by optimization algorithms; by contrast, z is referred to as *variables*, the value of which will be searched for optimal solutions. Suppose that the feasible set is defined by a finite set of constraints:

$$R(x,\theta) = \{z \in \mathbb{R}^{n_z} : \; g_i(z;x,\theta) \le 0, \quad i \in [m_1] \\ h_i(z;x,\theta) = 0, \quad i \in [m_2]\},$$

where we use the shorthand $[m] = \{1,...,m\}$. As in DCP, we assume that the objective function g and constraints $\{g_i\}_{i\in[m_1]}$ and $\{h_i\}_{i\in[m_2]}$ are constructed from a given library of base functions, i.e., *expressions*. In DPP, an expression is said to be parameter-affine if it does not involve variables and is affine in its parameters, and it is parameter-free if it does not have parameters. Under DPP, all parameters are classified as affine, just like variables. Also, the product of two expressions is affine when at least one of the expressions is constant, or when one of the expressions is parameter-affine and the other is parameter-free. For example, let $\{A, a, \lambda\}$ be parameters and z be variable. Then, $Az - a = 0$ is DPP because Az is affine (A is parameter-affine and z is parameter-free), $-g$ is affine, and the sum of affine expressions is affine. Similarly, $\lambda\|z\|_2 \le 0$ is DPP because $\lambda\|z\|_2$ is affine (λ is parameter-affine and $\|z\|_2$ is parameter-free). It is often possible to re-express non-DPP expressions in DPP-compliant ways. For instance, let θ_1, θ_2 be parameters, then $\theta_1\theta_2$ is not DPP because both arguments are parametrized; it can be rewritten in a DPP-compliant way by introducing a variable z, replacing $\theta_1\theta_2$ with the expression $\theta_1 z$ while adding the constraint $z = p_2$. Similarly, if A_1 is a parameter representing a positive semidefinite matrix, the expression $z^\top Az$ is not DPP; it can be rewritten as $\|A_2 z\|_2^2$, where A_2 is a new parameter representing $A_1^{1/2}$. The set of

DPP-compliant optimizations is very broad, including many instances of cone programs. We make sure to follow the DPP rule throughout the paper so that the result is practically relevant to end-to-end optimization that requires differentiation through the solution function for backpropagation (see (Agrawal et al. 2019)).

3 Approximation through the Lens of Regression

In classical approximation theory, approximation rates are obtained assuming full access to the target function f (DeVore and Lorentz 1993; Yarotsky 2018; DeVore, Hanin, and Petrova 2021). In this section, we develop a new viewpoint of approximation through the lens of regression with experimental design, where we leverage an estimation procedure that learns the target function through a dataset to reason about the complexity of approximation.

We formulate the approximation problem in a setting closely related to fixed-design regression (Györfi et al. 2002). Here, let $\mathcal{D}_n = \{(x_1, f(x_1)), \cdots, (x_n, f(x_n))\}$ be a dataset of $n \in \mathbb{N}$ points, where the locations $\{x_i\}_{i\in[n]} \in X^n$ to evaluate the target function f can be arbitrarily selected.[1] The key idea of our proof technique is to first construct an estimator $\mathcal{A} : (X \times \mathbb{R})^n \to \Pi$, where Π is the class of functions that we are analyzing (i.e., the set of solution functions in our setting).[2] We then characterize the approximation error based on the regression error of the estimator; meanwhile, we can reason about the rate-distortion by examining the complexity of the constructed function $\mathcal{A}(\mathcal{D}_n)$.

In the following theorem, we establish the *first* universal approximation theorem for the class of solution functions corresponding to LPs. Readers are referred to the supplementary material for details of proof in the main document.

Theorem 1 (Approximation of C^2 by max-affine regression)**.** *For any target function $f \in C^2$ and $\epsilon > 0$, there exists a solution function π of an LP with $\mathcal{O}\Big(\big(\frac{n_x}{\epsilon}\big)^{\frac{n_x}{2}}\Big)$ constraints and $n_x + 1$ variables, such that $\|f - \pi\|_\infty \leq \epsilon$.*

The proof exploits the fact that any C^2 function can be approximated by a DC function (Bačák and Borwein 2011); we then construct a numerical procedure by extending the algorithm from (Balázs, György, and Szepesvári 2015) for max-affine regression to learn the potentially nonconvex target function from some dataset $\mathcal{D}_n$.

Note that in statistical learning theory, it is uncommon to impose a model class assumption on the target function that gives rise to the data, so the generalization error is compared with the best-in-class; while the generalization error may vanish as more data is collected, the approximation error may always be bounded away from zero because the target function may not lie in the function class of estimators (Cucker and Zhou 2007). The critical implication of the above result is that we can approximate any smooth function to arbitrary precision by constructing an LP with enough constraints and variables. This is not without surprise, as LPs have arguably the simplest form within the broad classes of optimization (Boyd, Boyd, and Vandenberghe 2004).

If we count a "neuron" in a neural network the same way we count a constraint in optimization, then the above approximation scheme gives the same order of complexity in terms of ϵ as a one-layer neural network (DeVore, Hanin, and Petrova 2021); however, the authors admit that a head-to-head comparison may not be fair (indeed, we later refer to an entire optimization program as a generalized neuron). Interestingly, complexity is mainly reflected in the number of constraints; the number of variables can be kept at the same level as the input dimension. Lastly, it is not our intention to exhaust all possible construction methods to derive the rate-distortion; other methods may also apply (He et al. 2020; DeVore, Hanin, and Petrova 2021).

4 The Role of Depth

In this section, we illuminate the role of depth in using deep architectures with solution functions. Interestingly, depth plays a very different role herein compared to DNNs (Yarotsky 2018; DeVore, Hanin, and Petrova 2021). We begin by introducing some formalities to characterize the architecture, which may be of independent interest.

4.1 Optimization-Induced Network Architecture

We consider a deep network as a directed acyclic graph (DAG), $\mathcal{N} \coloneqq (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are finite sets of vertices (a.k.a., nodes) and directed edges. The set $\mathcal{V}$ consists of the set $\mathcal{V}_i$ of input vertices as placeholders for independent variables (i.e., inputs x), the set $\mathcal{V}_o$ of output vertices (i.e., corresponding output), and the set $\mathcal{V}_h \coloneqq \mathcal{V} \setminus \{\mathcal{V}_i, \mathcal{V}_o\}$ of hidden vertices, which store certain intermediate values to compute the output. The output of each $v \in \mathcal{V} \setminus \mathcal{V}_i$ is given by the solution function (parameterized by θ_v), $\pi_v(\cdot; \theta_v)$, which takes as input from incoming edges; for each edge $e \in \mathcal{E}$, an affine transformation $h_e(\cdot; \theta_e)$ is applied to the output of the incident node. Analogous to DNNs, we define a general notion of a neuron as a computational unit associated with each node $v \in \mathcal{V}$, which takes as input the (possibly vector-valued) outputs $x_{v'}$ from the incident nodes $v' \in \mathcal{V} \setminus \mathcal{V}_o$ with an edge $e = (v', v) \in \mathcal{E}$ directed to v, and produces the output

$$x_v \coloneqq \pi_v(\{h_e(x_{v'}; \theta_e)\}_{e=(v',v)\in\mathcal{E}}, \theta_v). \tag{2}$$

As a convention, outputs from input nodes $v \in \mathcal{V}_i$ are externally provided function inputs; outputs from neurons associated with output nodes $v \in \mathcal{V}_o$ are given by affine transformation of values from adjacent incoming nodes. Thus, we define the output function $f_\mathcal{N} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_o}$ of the network $\mathcal{N}$ by

$$f_\mathcal{N}(x) \coloneqq (x_v, v \in \mathcal{V}_o). \tag{3}$$

Since a vector-valued function can be regarded as a concatenation of scalar-valued functions, for simplicity, we will only

[1] Note that, for the purpose of analyzing approximation power, the labels received in the dataset are assumed to be accurate (i.e., noiseless observations). Noisy data can be processed by combining the proposed method with standard regression techniques.

[2] In general, we can allow the estimator to have infinite computational power to solve a nonconvex optimization to arbitrary accuracy; however, for practical purposes, we restrict it to being a computationally efficient procedure so that we can obtain an approximant with a reasonable amount of time by learning from a finite dataset $\mathcal{D}_n$.

consider the case where $n_o = 1$. The collection of $\{\theta_v, \theta_e\}$ for $v \in \mathcal{V}$ and $e \in \mathcal{E}$ are referred to as the trainable parameters of $\mathcal{N}$. For a fixed architecture, the set of output functions forms a parameterized nonlinear manifold.

For the exposition, we can also organize the nodes of $\mathcal{N}$ into layers. The zeroth layer, called the input layer, consists of all n_x input vertices in $\mathcal{V}_i$. The input layer is followed by hidden vertices organized into L hidden layers, with the j-th layer $\mathcal{H}_j$ consisting of all n_j vertices that are j-hop away from the input layer excluding the output vertices, for $j \in [L]$. Finally, the output layer consists of all output nodes $\mathcal{V}_o$, which contains at least one node that is $L+1$-hop neighbor of the input layer (otherwise, the depth must be less than L). The main distinction with conventional DNNs is that the computation of a neuron is given by some solution function instead of the usual coordinate-wise activation function (e.g., ReLU). Since the computational complexity of an optimization family (e.g., LPs/QPs) can usually be characterized by the number of variables and constraints, we measure the width of each layer by the total number of variables and constraints among nodes therein, due to a simple fact that we state without proof.

Proposition 1 (Concatenation rule)**.** *The concatenation of solution functions $\{\pi_v(\cdot, \theta_v)\}_{v\in\mathcal{V}'}$, where $\pi_v(\cdot, \theta_v)$ is associated with an optimization with n_v^z variables and n_v^c constraints, can be written as a solution function of some optimization with $\sum_{v\in\mathcal{V}'} n_v^z$ variables and $\sum_{v\in\mathcal{V}'} n_v^c$ constraints (up to some additive constants no larger than $|\mathcal{V}'| + 1$).*

Henceforth, we refer to the integers $W_j^v \coloneqq \sum_{v\in\mathcal{H}_j} n_v^z$ and $W_j^c \coloneqq \sum_{v\in\mathcal{H}_j} n_v^c$ as the *variable width* (v-width) and *constraint width* (c-width) of the j-th layer, respectively, and L as the *depth* of the network. We usually use the maximum v-width and c-width among all hidden layers, denoted by W^v and W^c respectively, to characterize the network width.

Definition 1. *We define $\Upsilon^{W^v,W^c,L}$ as the family of functions $f_{\mathcal{N}}$ in* (3) *with width and depth bounded by W^v, W^c, and L.*

The set of solution functions $\Pi = \{\pi(\cdot;\theta) : \mathbb{R}^{n_x} \to \mathbb{R} \mid \theta \in \Theta\}$ can be regarded as a single-layer network where the output transformation is identity. In the sections pertaining to approximation property, we will focus exclusively on LP/QP solution functions. Also, to make a distinction between *networks* and *network architectures*: We define the latter as the former with unspecified weights. The universal approximation property of a network architecture is discussed in the sense that we can approximate any function from a model class $\mathcal{F}$ with error ϵ by simply some weight assignment.

4.2 Exact Construction of Taylor Polynomials

In the recent work of (Yarotsky 2018), it is shown that DNNs can approximate some elementary functions such as the multiplication operator with progressive accuracy by increasing depth, which is then used to establish the improvement of approximation due to depth. The first intriguing role of depth for optimization-induced architecture is that we can *exactly construct* the product function with a network of only two layers. This is based on the following simple observations:

- The solution to $\{\min_z -z \quad \text{s.t.} \quad x_1 z \le x_2\}$, where $x_1, x_2 \in (0, 1]$ are the parameters of the optimization (i.e., inputs), is exactly x_2/x_1.
- A two-layer architecture, with $\{\min_z -z \ \text{ s.t. } \ x_1 z \le 1\}$ as the first layer, the output of which is provided as input to $\{\min_z -z \ \text{ s.t. } \ \square z \le x_2\}$ as the parameter value for $\square$, has an output of $x_1 x_2$ for any $x_1, x_2 \in (0, 1]$.

Both observations can be directly verified by writing the Karush–Kuhn–Tucker (KKT) conditions.[3] Note that in the above, we do not consider the measure-zero event that any coordinate of x can be 0. Most importantly, observe that the above constructions comply with the DPP rule (e.g., $x_1 z$ is affine because x_1 is parameter-affine and z is parameter-free). In the following, we use $\binom{k}{n}$ to denote the binomial coefficient n choose k and $\lceil\cdot\rceil$ as the ceiling function.

Theorem 2. *There exists a universal multilayer LP architecture that:*

(a) can approximate any function from F_{k,n_x} with uniform error bounded by $\epsilon > 0$;

(b) has a depth of at most $2k$ and the widest layer has at most $\left(2n_x + 2 + 2\binom{n_x}{k+n_x}\right)N^{n_x}$ constraints and $\left(1 + \binom{n_x}{k+n_x}\right)N^{n_x}$ variables, where $N = \lceil n_x(\frac{1}{k!\epsilon})^{1/k}\rceil$.

The theorem above provides an upper bound for approximation complexity with a universal network architecture to approximate all functions in F_{k,n_x}. For DNNs to achieve the same error ϵ (Yarotsky 2017), one needs a depth of $\mathcal{O}(\ln(1/\epsilon))$ with $\mathcal{O}(\epsilon^{-n_x/k}\ln(1/\epsilon))$ weights; in contrast, for optimization-induced networks, we only need a fixed depth that does not grow with the accuracy requirement ϵ and $\mathcal{O}(\epsilon^{-n_x/k})$ constraints and variables. The removal of the dependence of depth on ϵ is due to the fact that instead of approximating some Taylor polynomial, as is done for DNNs with ReLU activation, we are able to *exactly reconstruct* the polynomial function with a deep optimization-induced network, which eliminates any approximation error due to reconstruction. We can also remove a factor of $2^{n_x^2/k}$ from the number of constraints and variables relative to what would yield without exploiting the special advantage of solution functions. This is based on another simple observation:

- The solution to $\{\min_{z\in[0,1]} -z \ \text{ s.t. } \ (x-1)z \le 0, (x+1)z \ge 0\}$, where $x \in \mathbb{R}$ is the optimization parameter (i.e., input), is exactly the bump function

$$\psi(x) = \begin{cases} 1, & |x| \le 1 \\ 0, & \text{otherwise} \end{cases};$$

[3] The first observation, in particular, may be contradictory to the common belief held in the multi-parametric programming literature that the solution function of an LP is always piecewise linear (Grancharova and Johansen 2012). A close examination of their argument reveals that it holds true only when the LP is free of any expression in its constraints where parameters multiply with variables.

- More generally, for an arbitrary union of intervals $\cup_{i\in I}[a_i, b_i]$, the solution to $\{\min_{z\in[0,1]} -z \quad \text{s.t.} \quad (x - b_i)z \leq 0, (x - a_i)z \geq 0, \forall i \in I\}$, where $x \in \mathbb{R}$, is exactly the multi-bump function that is 1 if $x \in \cup_{i\in I}[a_i, b_i]$ and 0 otherwise.

The proof relies on partitioning the input space into a grid of $(N + 1)^{n_x}$ functions. With the above bump functions, we can decrease the size of the grid N from $\lceil(\frac{\epsilon k!}{2^{n_x} n_x^k})^{-1/k}\rceil$ in (Yarotsky 2018) to $\lceil(\frac{\epsilon k!}{n_x^k})^{-1/k}\rceil$ by a factor of $2^{n_x/k}$, resulting in an overall reduction of $2^{n_x^2/k}$ in complexity, which can be substantial for high dimensions. Interestingly, from the construction of bump functions, the solution function can act as some switching mechanism or logical expressions (e.g., if-then-else), similar to the role of a binary variable in mixed-integer programming. To sum up, Sections 3 and 4 answered the first question posed in the introduction, namely, the approximation properties of the solution functions. We now proceed to answer the second question, namely, the statistical complexity of the solution functions.

5 Definability and Whitney Stratification

Real-world optimization often has some nice structures that can be exploited (Ioffe 2009). Given some mild assumptions about objective/constraint functions, we can show that the solution function enjoys a nice property, namely, Whitney stratification, which induces desirable computational guarantees (Ioffe 2009; Davis et al. 2020). In this section, we resume the generality of a convex optimization problem.

First, let us recall some fundamental concepts in tame geometry (Van den Dries and Miller 1996; Ioffe 2009).

Definition 2 (Whitney Stratification)**.** *A Whitney C^k stratification of a set I is a partition of I into finitely many nonempty C^k manifolds, called strata, with the following conditions:*

1) For any two strata I_a and I_b, $I_a \cap I_b \neq \emptyset$ implies that $I_a \subset \mathrm{cl} I_b$ holds, where $\mathrm{cl} I_b$ is the closure of the set I_b.
2) For any sequence of points x_k in a stratum I_a, converging to a point $x^\star$ in a stratum I_b, if the corresponding normal vectors $v_k \in N_{I_a}(x_k)$ converge to a vector v, then the inclusion $v \in N_{I_b}(x^\star)$ holds. Here, $N_{I_a}(x_k)$ denotes the normal cone to I_a at x_k.

Roughly speaking, stratification is a locally finite partition of a given set into differentiable manifolds, which fit together in a regular manner (property *1)* in Def. 2). Whitney stratification as defined above is a special type of stratification for which the strata are such that their tangent spaces (as viewed from normal cones) also fit regularly (property *2)*). There are several ways to verify Whitney stratifiability. For example, one can show that the function under study belongs to one of the well-known function classes, such as semialgebraic functions, whose members are known to be Whitney stratifiable (Van den Dries and Miller 1996). However, to study the solution function of a general convex optimization problem, we need a far-reaching axiomatic extension of semialgebraic sets to classes of functions definable on "o-minimal structures," which are very general classes and share several attractive analytic features as semialgebraic sets, including Whitney stratifiability (Van den Dries and Miller 1996); the definition of o-minimal structures can be found in the appendix.

Assumption 1. *The function g and the set-valued map R are definable in some o-minimal structure.*

This is a mild assumption as practically all functions from real-world applications, including deep neural networks, are definable in some o-minimal structure (Davis et al. 2020); also, the composition of mappings, along with the sum, inf-convolution, and several other classical operations of analysis involving a finite number of definable objects in some o-minimal structure remains in the same structure (Van den Dries and Miller 1996).

Theorem 3. *The solution function* (1) *is Whitney stratifiable. In addition, any function in the class $\Upsilon^{W^v, W^c, L}$ is Whitney stratifiable for any positive integers W^v, W^c, and L.*

The far-reaching consequence of definability, exploited in this study, is that definable sets and functions admit, for each $k \geq 1$, a C^k–Whitney stratification with finitely many strata (see, for instance, (Van den Dries and Miller 1996, Result 4.8)). This remarkable property, combined with the result that any stratifiable functions enjoy a nonsmooth Kurdyka–Łojasiewicz inequality (Bolte et al. 2007), provides the basis for convergence analysis of many optimization algorithms (Drusvyatskiy and Lewis 2018). In particular, the application of subgradient methods to solution functions or optimization-induced networks is endowed with rigorous convergence guarantees (see, e.g., (Davis et al. 2020)).

6 Covering Number Bounds

In this section, we provide covering number bounds for the solution functions of LPs, QPs, and in general, any convex programs. We focus on the empirical L_1 covering number $\mathcal{N}_1(\epsilon, \Pi, n)$, which is a variant of the covering number for a set of functions Π at ϵ accuracy with respect to the empirical L_1 metric defined over n data points.

Definition 3 ((Zhang 2002), empirical L_1 covering number)**.** *Given observations $\mathcal{D}_n = \{x_1, ..., x_n\}$ and vectors $f(\mathcal{D}_n) = [f(x_1), ..., f(x_n)] \in \mathbb{R}^n$ for any $f \in \mathcal{F}$, the empirical L_1 covering number, denoted as $\mathcal{N}_1(\epsilon, \mathcal{F}, \mathcal{D}_n)$, is the minimum number m of a collection of vectors $g_1, ..., g_m \in \mathcal{F}$, such that $\forall f \in \mathcal{F}$, there exists an g_j such that*

$$\|f - g_j\|_{\mathcal{D}_n} \coloneqq \frac{1}{n}\sum_{i=1}^{n} |f(x_i) - g_j(x_i)| \leq \epsilon.$$

We define $\mathcal{N}_1(\epsilon, \mathcal{F}, n) = \sup_{\mathcal{D}_n} \mathcal{N}_1(\epsilon, \mathcal{F}, \mathcal{D}_n)$. The set $\{g_1, ..., g_m\}$ above is called the (empirical) ϵ-cover of $\mathcal{F}$ and the logarithm of covering number $\log \mathcal{N}_1(\epsilon, \mathcal{F}, n)$ is known as the entropy number.

6.1 The Class of LPs and QPs

Consider the function

$$\pi_{QP}(x; \theta) \coloneqq \underset{z\in R(x,\theta)}{\arg\min} \left(\frac{1}{2}A_0 z + U_0^x x + U_0^\theta \theta + b_0\right)^\top z, \tag{4}$$

with $R(x,\theta) := \{z : A_1 z \leq b_1 + U_1^x x + U_1^\theta \theta, A_2 z = b_2 + U_2^x x + U_2^\theta \theta\}$, where x and θ are the parameters, $z \in \mathbb{R}^{n_z}$ is the optimization variable, and all the rest are fixed hyperparameters of compatible dimensions; in particular, $A_0 \succ 0$ is positive definite, and m_1 and m_2 are the number of inequality and equality constraints, respectively. We can also define $\pi_{LP}(x;\theta)$ by setting $A_0 = 0$ in (4). Now, let us also introduce

$$\kappa^\star_{LP} = \max_{\iota \subseteq \{1,\ldots,m_1\}, |\iota| \leq n_z - m_2} \left\| \tilde{A}_{LP}(\iota) \right\|_2, \tag{5}$$

where

$$\tilde{A}_{LP}(\iota) = \begin{bmatrix} [A_1]_\iota \\ A_2 \end{bmatrix}^{-1} \begin{bmatrix} [U_1^x]_\iota \\ U_2^x \end{bmatrix}. \tag{6}$$

Similarly, define

$$\kappa^\star_{QP} = \max_{\iota \subseteq \{1,\ldots,m_1\}, |\iota| \leq n_z - m_2} \|\tilde{A}_{QP}(\iota)\|_2, \tag{7}$$

where $\tilde{A}_{QP}(\iota)$ is given as

$$\tilde{A}_{QP}(\iota) = M\left([A_1]_\iota A_0^{-1} U_0^x + [U_1^x]_\iota\right) - A_0^{-1}[U_0^x]_\iota \tag{8}$$

with $M = A_0^{-1}[A_1]_\iota^\top \left([A_1]_\iota A_0^{-1}[A_1]_\iota^\top\right)^{-1}$. Here, $\|\cdot\|_2$ is the spectral norm, and $[A_1]_\iota$ is the submatrix formed by the rows of A_1 indexed by ι, and the inverse is understood as pseudo-inverse in the case of a rectangular matrix. The quantities $\kappa^\star_{LP}$ and $\kappa^\star_{QP}$ are some condition numbers associated with the optimization parameters. For instance, under the linear independence constraint qualification (Luo, Pang, and Ralph 1996), the inverse matrix in (6) is always full-rank; the alignment of some constraints, on the other hand, will result in larger values of $\kappa^\star_{LP}$, since it will yield larger dual variables. In this section, we assume standard constraint qualifications, including Mangasarian-Fromovitz constraint qualification, constant rank constraint qualification, and strong coherent orientation condition (Luo, Pang, and Ralph 1996), such that both $\kappa^\star_{LP}$ and $\kappa^\star_{QP}$ are bounded. We also assume that Θ is compact and $m_1 + m_2 \geq n_z$, where n_z is the dimension of the variables in (4); this assumption is easy to be relaxed with a slightly more complicated (but not necessarily more insightful) bound, thus we make the restriction to streamline the presentation. We define $\Pi_\square = \{\pi_\square(\cdot,\theta) : \theta \in \Theta\}$, where $\square$ can be LP or QP.

Theorem 4. *The empirical L_1 covering number of $\Pi_\square$ over bounded input space is controlled by*

$$\log \mathcal{N}_1(\epsilon, \Pi_\square, n) \lesssim \frac{\kappa^{\star 2}_\square}{\epsilon^2} \sum\nolimits_{0 \leq i \leq n_z - m_2} \binom{m_1}{i},$$

where $\square$ can be either LP or QP.

The above bound implies that the complexity of the class of LPs and QPs increases by the number of inequality constraints (which agrees with our approximation results obtained so far) and also depends on the conditional numbers $\kappa^\star_{LP}$ and $\kappa^\star_{QP}$ that bound the maximum slope of affine functions among all pieces.

6.2 Generic Optimization Class

More generally, we consider any optimization (possibly nonconvex) with a definable objective and constraints. We note that bounding the entropy numbers of the classes C^k with respect to the supremum norm was were among the first results after the introduction of the concept of covering numbers (e.g., (Cucker and Smale 2002)). However, the solution function does not belong to C^k since the function is only piecewise smooth and may be even not differentiable at the boundary between two pieces; besides, each piece may be nonconvex, but existing results assume that the domain is convex.

Theorem 5. *Consider the set $\Pi := \{\pi(\cdot,\theta) : \theta \in \Theta\}$, where π is defined in* (1) *and X and Θ are compact. Then,*

$$\log \mathcal{N}_1(\epsilon, \Pi, n) \lesssim n\,(1/\epsilon)^{1/k} + k^{n_x} \log(1/\epsilon),$$

where the constant depends on the number of strata in the C^k-Whitney stratification, which is always finite.

The proof exploits the result from the last section that the solution function of any optimization given by definable objective and constraints is Whitney stratifiable. This provides us with a starting point to bound the complexity, since any Whitney stratifiable function is piecewise-smooth with bounded pieces (Van den Dries and Miller 1996). We also proved an empirical L_1 covering number bound for C^k functions, which can be of independent interest. Note that (Pontil 2003) proved that the empirical covering number is on the same order of the standard covering number for the smooth function class, in the sense that the empirical covering number can be lower bounded by the standard covering number on a larger scale (see the lower bound in (Pontil 2003, Thm. 1)). However, for their lower bound to be non-vacuous, we need the size of the dataset to be on the same order as a covering set of the entire space. This only applies when the number of data scales exponentially with dimension n_x, which is rarely the case with practical problems. On the other hand, our result makes explicit the dependence on the number of data points n and recovers the bound for the standard covering number when $n = \mathcal{O}\left((1/\epsilon)^{n_x}\right)$ in view of (Pontil 2003).

To summarize this section, the key implication of Theorems 4 and 5 is that the solution function of any definable optimization is statistically learnable.

7 Conclusion

In this paper, we provide definite (but partial) answers to fundamental questions about the approximation and learning-theoretic properties of solution functions. The results provided in the paper can be used to understand questions about sample complexity and approximation capacity in the practice of decision making, and can help guide practice in various engineering domains discussed in the Introduction. Given the importance of this class of functions in the practical and theoretical arena, we expect our results to advance the understanding of algorithm design and spur further research on this problem.

Acknowledgments

The authors acknowledge the generous support by NSF, the Commonwealth Cyber Initiative (CCI), C3.ai Digital Transformation Institute, and the U.S. Department of Energy.

References

Ab Azar, N.; Shahmansoorian, A.; and Davoudi, M. 2020. From inverse optimal control to inverse reinforcement learning: A historical review. *Annual Reviews in Control*, 50: 119–138.

Adams, S.; Cody, T.; and Beling, P. A. 2022. A survey of inverse reinforcement learning. *Artificial Intelligence Review*, 1–40.

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, J. Z. 2019. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32.

Agrawal, A.; Verschueren, R.; Diamond, S.; and Boyd, S. 2018. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1): 42–60.

Allen-Zhu, Z.; and Li, Y. 2020. Backward Feature Correction: How Deep Learning Performs Deep Learning. *arXiv preprint arXiv:2001.04413*.

Amos, B. 2022. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*.

Amos, B.; and Kolter, J. Z. 2017. OptNet: Differentiable optimization as a layer in neural network. In *International Conference on Machine Learning*, 136–145. PLMR.

Anthony, M.; and Bartlett, P. L. 1999. Neural network learning: Theoretical foundations. *Cambridge university press Cambridge*, 9.

Bačák, M.; and Borwein, J. M. 2011. On difference convexity of locally Lipschitz functions. *Optimization*, 60(8-9): 961–978.

Baes, M.; Diehl, M.; and Necoara, I. 2008. Every continuous nonlinear control system can be obtained by parametric convex programming. *IEEE Transactions on Automatic Control*, 53(8): 1963–1967.

Balázs, G. 2022. Adaptively partitioning max-affine estimators for convex regression. In *Proceedings of Machine Learning Research*, 860–874. PMLR.

Balázs, G.; György; and Szepesvári, C. 2015. Near-optimal max-affine estimators for convex regression. In *Proceedings of Machine Learning Research*, 56–64. PMLR.

Bartlett, P. L.; Harvey, N.; Liaw, C.; and Mehrabian, A. 2019. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*, 20(1): 2285–2301.

Bartlett, P. L.; Maiorov, V.; and Meir, R. 1998. Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural computation*, 10(8): 2159–2173.

Bertsimas, D.; Gupta, V.; and Paschalidis, I. C. 2015. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2): 595–633.

Bianchini, M.; and Scarsell, F. 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems,*, 25: 1553–1565.

Bolte, J.; Daniilidis, A.; Lewis, A.; and Shiota, M. 2007. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2): 556–572.

Boyd, S.; Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Chen, M.; Jiang, H.; Liao, W.; and Zhao, T. 2019. Efficient Approximation of Deep ReLU Networks for Functions on Low Dimensional Manifolds. In *Advances in Neural Information Processing Systems 32*. NeurIPS.

Cucker, F.; and Smale, S. 2002. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1): 1–49.

Cucker, F.; and Zhou, D. X. 2007. *Learning theory: an approximation theory viewpoint*, volume 24. Cambridge University Press.

Davis, D.; Drusvyatskiy, D.; Kakade, S.; and Lee, J. D. 2020. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1): 119–154.

DeVore, R.; Hanin, B.; and Petrova, G. 2021. Neural network approximation. *Acta Numerica*, 30: 327–444.

DeVore, R. A.; Howard, R.; and Micchelli., C. 1989. Optimal nonlinear approximation. *Manuscipta Mathematica*, 64(4): 469–478.

DeVore, R. A.; and Lorentz, G. G. 1993. *Constructive approximation*, volume 303. Springer Science & Business Media.

Dontchev, A. L.; and Rockafellar, R. T. 2009. *Implicit functions and solution mappings*, volume 543. Springer.

Donti, P.; Amos, B.; and Kolter, J. Z. 2017. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30.

Drusvyatskiy, D.; and Lewis, A. S. 2018. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3): 919–948.

Ebert, F.; Finn, C.; Dasari, S.; Xie, A.; Lee, A.; and Levine, S. 2021. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:2109.03920*.

Elmachtoub, A. N.; and Grigas, P. 2020. Smart "Predict, then Optimize". *Management Science*, 68(1): 2–26.

Feber, A.; Wilder, B.; Dilkina, B.; and Tambe, M. 2020. MIPaaL: Mixed Integer Program as a Layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(2): 1504–1511.

Fiacco, A. V. 2020. *Mathematical programming with data perturbations*. CRC Press.

Ghosh, A.; Pananjady, A.; and A. Guntuboyia, a. K. 2019. Maxaffine regression: Provable, tractable, and near-optimal statistical estimation. *arXiv preprint arXiv:1906.09255*.

Grancharova, A.; and Johansen, T. A. 2012. Explicit nonlinear model predictive control: Theory and applications. *Springer Science and Business Media*, 429.

Györfi, L.; Kohler, M.; Krzyzak, A.; Walk, H.; et al. 2002. *A distribution-free theory of nonparametric regression*, volume 1. Springer.

Hanin, B. 2019. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10): 992.

Hannah, L. A.; and Dunson, D. B. 2013. Multivariate convex regression with adaptive partitioning. *Journal of Machine Learning Research*, 14(1): 3261–3294.

He, J.; Li, L.; Xu, J.; and Zheng, C. 2020. Relu deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3): 502–527.

Hempel, A. B.; Goulart, P. J.; and Lygeros, J. 2014. Inverse parametric optimization with an application to hybrid system control. *IEEE Transactions on Automatic Control*, 60(4): 1064–1069.

Hewing, L.; Wabersich, K. P.; Menner, M.; and Zeilinger, M. N. 2020. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3: 269–296.

Hildreth, C. 1957. Point estimates of ordinates of concave functions. *Journal of the American Statistical Association*, 49: 598–619.

Hong, M.; Wai, H.-T.; Wang, Z.; and Yang, Z. 2020. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5): 359–366.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2020. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Ioffe, A. D. 2009. An invitation to tame optimization. *SIAM Journal on Optimization*, 19(4): 1894–1917.

Jia, R.; Konstantakopoulos, I. C.; Li, B.; and Spanos, C. 2018. Poisoning Attacks on Data-Driven Utility Learning in Games. *Annual American Control Conference (ACC)*, 5774–5780.

Kotary, J.; Fioretto, F.; Van Hentenryck, P.; and Wilder, B. 2021. End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Liu, R.; Gao, J.; Zhang, J.; Meng, D.; and Lin, Z. 2021. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Loke, G. G.; Tang, Q.; and Xiao, Y. 2021. Decision-Driven Regularization: A Blended Model for Predict-then-Optimize. *Available at SSRN 3623006*.

Lorraine, J.; Vicol, P.; and Duvenaud, D. 2020. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, 1540–1552. PMLR.

Lu, J.; Shen, Z.; Yang, H.; and Zhang, S. 2021. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5): 5465–5506.

Luo, Z.-Q.; Pang, J.-S.; and Ralph, D. 1996. Mathematical Programs with Equilibrium Constraints. *Cambridge University Press*.

Magnani, A.; and Boyd, S. P. 2009. Convex piecewise-linear fitting. *Optimization and Engineering*, 10(1): 1–17.

Nguyen, N.; Gulan, M.; Olaru, S.; and Rodriguez-Ayerbe, P. 2018. Convex lifting: Theory and control applications. *IEEE Transactions on Automatic Control*, 63(5): 1243–1258.

Pinkus, A. 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8: 143–195.

Pontil, M. 2003. A note on different covering numbers in learning theory. *Journal of Complexity*, 19(5): 665–671.

Serra, T.; Tjandraatmadja, C.; and Ramalingam, S. 2018. Bounding and counting linear regions of deep neural networks. In *International conference on machine learning*, 4558–4566. PMLR.

Siahkamari, A.; Gangrade, A.; Kulis, B.; and Saligrama, V. 2000. Piecewise linear regression via a difference of convex functions. In *International conference on machine learning*, 8895–8904. PLMR.

Toriello, A.; and Vielma, J. P. 2015. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1): 86–95.

Van den Dries, L.; and Miller, C. 1996. Geometric categories and o-minimal structures. *Duke Mathematical Journal*, 84(2): 497–540.

Wang, K.; Shah, S.; Chen, H.; Perrault, A.; Doshi-Velez, F.; and Tambe, M. 2021. Learning MDPs from Features: Predict-Then-Optimize for Sequential Decision Problems by Reinforcement Learning. *arXiv preprint arXiv:2106.03279*.

Wilder, B.; Dilkina, B.; and Tambe, M. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1658–1665.

Yarotsky, D. 2017. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114.

Yarotsky, D. 2018. Optimal approximation of continuous functions by very deep ReLU networks. In *Conference on learning theory*, 639–649. PMLR.

Zeng, Y.; Chen, S.; Park, W.; Mao, Z.; Jin, M.; and Jia, R. 2022. Adversarial Unlearning of Backdoors via Implicit Hypergradient. In *International Conference on Learning Representations*.

Zhang, T. 2002. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2(Mar): 527–550.