UNIVERSITY OF CALIFORNIA
RIVERSIDE

Learning-Augmented Online Decision Making With Guaranteed Trustworthiness

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Jianyi Yang

September 2023

Dissertation Committee:

Dr. Shaolei Ren, Chairperson
Dr. Amit K. Roy-Chowdhury
Dr. Salman Asif

The Dissertation of Jianyi Yang is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to express my sincere gratitude to all those who have helped me a lot during my PhD study and the completion of this thesis.

First and foremost, I would like to extend my deepest gratitude to my Advisor, Dr. Shaolei Ren, for his continuous guidance and support during my PhD study. Dr. Ren's insightful ideas consistently inspire me, particularly when I encounter difficulties in my research. Without his invaluable mentorship, I would not have been able to navigate through my dissertation work with such smoothness and success. Dr. Ren is a true role model for me in my following academic career. His dedication to research has been a tremendous source of encouragement for me. I am truly fortunate to have him as my PhD advisor in the past five years and profoundly grateful for his mentorship. Additionally, I would like to express my immense gratitude to Dr. Adam Wierman from Caltech, with whom Dr. Ren and I have closely collaborated over the past year. Dr. Wierman's invaluable advice and generous support have significantly contributed to several aspects of this dissertation. I am deeply appreciative of his contributions and would like to extend my sincere thanks to him.

I am also immensely grateful to Dr. Amit K Roy-Chowdhury and Dr. Salman Asif for being in my dissertation committee and qualification committee. Their valuable suggestions and feedback have played an important role in enhancing the quality of my dissertation. I am truly grateful for their expertise and guidance throughout this process. Additionally, I am deeply appreciative of Dr. Samet Oymak for his involvement in my qualification committee. Although Dr. Oymak was unable to join my dissertation committee due to availability, his encouragement have been invaluable to me during my PhD journey.

Also, I am deeply appreciative of my collaborator, Dr. Tongxin Li from CUHK, Shenzhen. With his extensive expertise in learning augmented control, he has played an important role in shaping some aspects of this dissertation. Furthermore, I am immensely grateful to my mentors, Dr. Jianshu Chen, and Dr. Xiaoman Pan during my research intern at Tencent AI Lab. I learned a lot of knowledge and improved my skills during the intern thanks to their guidance and expertise.

I would like to extend my gratitude to my excellent labmates, Pengfei Li, Bingqian Lu, Zhihui Shao, Luting Yang, Fangfang Yang, Yejia Liu, and Mohammad Jaminur Islam at UC, Riverside who have been great collaborators and friends during my PhD journey. The dissertation is a product of collaborations and shared efforts with many of them.

Finally, I extend my heartfelt thanks to my family. This dissertation is dedicated to my parents for their unconditional love and unwavering support.

Funding Acknowledgement. I appreciate the funding support from National Science Foundation under grant number CNS-1910208, CNS-1551661 and ECCS-1610471.

Publication Acknowledgement. The text of this dissertation, in part or in full, is a reprint of the material as it appears in list of publications below. The co-author Dr. Shaolei Ren listed in that publication directed and supervised the research which forms the basis for this dissertation.

- Jianyi Yang, and Shaolei Ren, "Robust Bandit Learning with Imperfect Context," AAAI Conference on Artificial Intelligence (AAAI), 2021.

  Added to the dissertation as Chapter 2. Jianyi Yang is the major contributor of the project.

- Jianyi Yang, and Shaolei Ren, Learning-Assisted Algorithm Unrolling for Online Optimization with Inventory Constraints, AAAI Conference on Artificial Intelligence (AAAI), 2023.

  Added to the dissertation as Chapter 5. Jianyi Yang is the major contributor of the project.

- Jianyi Yang, and Shaolei Ren, Towards Understanding Informed Deep Neural Networks: Convergence, Generalization, and Sampling Complexity, International Conference on Machine Learning (ICML),2022.

  Added to the dissertation as Chapter 6. Jianyi Yang is the major contributor of the project.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Learning-Augmented Online Decision Making With Guaranteed Trustworthiness

by

Jianyi Yang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, September 2023
Dr. Shaolei Ren, Chairperson

Many mission-critical systems need to solve online decision-making problems such as workload scheduling in datacenters, power allocation in edge computing, battery management for EV charging, demand response in power systems, etc. Decision-making algorithms in these systems are expected to achieve a high expected reward while guaranteeing some important trustworthiness metrics such as robustness, safety, fairness, etc. Recently, machine learning (ML) for decision processes, utilizing available statistical information to achieve a high expected reward, has been attracting growing interests. However, ML algorithms usually suffer from the lack of trustworthiness guarantees, which hinders their deployments in real systems. On the other hand, domain expert algorithms have been programmed in many real systems for a long time and can be trusted in terms of some performance metrics, but they may not achieve a high enough expected reward. In this dissertation, given various decision processes, we design algorithms to exploit corresponding expert knowledge to achieve both high expected reward and provably-guaranteed worst-case performances, and validate the performance of the proposed algorithms on applications of computing systems.

Specifically, the dissertation includes learning-augmented algorithms and theory in the following aspects. First, the dissertation consider bandit decision making with imperfect context and proposes robust algorithms to maximize the worst-case reward minimize the worst-case regret, respectively. The simulations of the algorithms on online edge datacenter selection validate our theoretical analysis. Then, the dissertation considers online optimization/control problems with known dynamic models and proposes expert calibrated ML algorithms with provable guarantees for anytime competitiveness. The theoretical analysis highlights the tradeoff between any-time competitiveness and average performance. The empirical results on electric vehicle charging station management are used to demonstrate the performance. Furthermore, without the knowledge of dynamic models, the dissertation designs reinforcement learning algorithms to optimize the expected reward while guaranteeing the anytime cost constraints for any episode. Experiments on the application of carbon-intelligent computing verify the reward performance and cost constraint guarantee for the proposed algorithm. Beyond that, the dissertation considers online decision making with multiple budget constraints and proposes machine learning (ML) assisted unrolling approach which unrolls the online decision pipeline and leverages an ML model for updating the Lagrangian multiplier online. For efficient training via backpropagation, we derive gradients of the decision model. Finally, the dissertation gives a theoretical analysis on general domain knowledge informed learning, quantitatively demonstrating the two benefits of do- main knowledge in informed learning — regularizing the label-based supervision and supplementing the labeled samples.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Challenges

Machine learning (ML) especially deep neural networks has achieved great success in multiple fields, such as computer vision, natural language processing, etc [396]. In recent year, the potential of ML in intelligently solving many difficult problems in mission-critical computing systems has attracted increasing attention from both academia and industry. However, this presents new challenges for algorithm design. First, it is well-known that ML based on statistical learning lacks robustness/trustworthiness guarantee and can have unsatisfactory performance especially for adversarial examples or out of distribution (OOD) testing [141]. This impedes the applicability of ML for real computing systems. Besides, domain knowledge can come from various sources in multiple forms. This brings challenges about utilizing domain knowledge and integrating it into ML models to improve the ML performance for computing systems. To solve these challenges, a series of learning-augmented algorithms are being developed to promote the applicability of ML in computing systems.

In mission critical computing systems, there is always the need to solve sequential/online decision-making problems. Examples include online workload scheduling and resource management in sustainable computing [326, 213], battery management for electrical vehicle (EV) charging [374], online demand response with renewables [275, 376], etc. In these problems, the actions at each time round rely on future unavailable information, which makes them difficult to be solved by traditional algorithms. Despite this, ML can solve these online problems well by exploiting enough training data to approximate the relationships between actions and available inputs. However, computing systems usually have a high requirement for robustness/trustworthiness to guarantee safety, quality of service, or fairness. Pure ML-based solutions lack the robustness/trustworthiness guarantee, which limits their applicability for online decision making problems in real systems. In this dissertation, we focus on the learning-augmented algorithm design for online decision making problems in mission-critical computing systems.

1.2 Learning-Augmented Decision Making

The learning augmented decision making design in this dissertation is related to the following research directions.

1.2.1 Learning to Optimize

The dissertation is closely related to the quickly expanding area of learning to optimize (L2O), which pre-trains an ML-based optimizer to directly solve optimization problems [238, 96, 406]. Most commonly, L2O focuses on speeding up the computation for

otherwise computationally expensive problems, such as as DNN training [28], nonconvex optimization in interference channels [257, 116] and combination optimization [117]. Moreover, ML-based optimizers have also been integrated into traditional algorithmic frameworks for faster and/or better solutions [96, 227].

Studies that apply L2O to solve difficult online optimization problems where the key challenge comes from the lack of complete offline information have been generally under-explored. In [228], an ML model is trained as a standalone end-to-end solution for a small set of classic online problems, such as online knapsack. In this dissertation, we give L2O algorithms for various online decision making problems and design algorithms to provably improve performance robustness.

1.2.2 Decision-Focused Learning

This dissertation is relevant to the recent decision-focused learning framework [7, 23, 407, 399]. Decision-focused learning train an ML model by and end-to-end manner to improve the decision-making performance. Decision-focused learning algorithms have been designed for convex optimization [7], combinatorial optimizations [407], and general optimizations [347]. In this dissertation, we apply the methodology of decision-focused learning and train the ML models aware of the downstream calibration/robustification procedure.

1.2.3 Learning-Augmented Algorithms

More recently, by combining ML-predicted actions with expert knowledge, ML-augmented algorithm designs have been emerging in the context of online optimization,

control and decision making [402, 70, 284, 106, 31, 339, 110]. The goal of ML-augmented online algorithms is to combine potentially untrusted ML predictions with robust policies (i.e., control priors). ML-augmented algorithms have been developed for online control/optimization by combining ML predictions and control priors through online switching [340, 31] or adaptively setting a confidence on the ML prediction [248, 246]. However, many learning-augmented algorithms for online decision making lacks the robustness guarantee given a sequence. Although some of the existing studies [248, 340, 111, 244, 241] provide provable competitive bounds, they cannot guarantee any-step trustworthiness constraints that is needed for real decision making [282]. Moreover, these studies typically assume a pre-trained ML model as a black box, whereas we study the policy learning process subject to the competitiveness constraint against a control prior.

1.2.4 Constrained/Conservative RL

Constrained RL algorithms have been designed to solve various Constrained MDP (CMDP) with reward objective and cost constraints. Among them, some are designed to guarantee the expected cost constraints [139, 392], some can guarantee the cost constraints with a high probability [97], and others guarantee a bounded violation of the cost constraints [168, 130, 131, 5]. In addition, conservative RL algorithms [157, 423, 217, 422, 411] also compare the cost performance with a policy prior, but they only guarantee the cost constraints against a policy prior in expectation. In real mission-critical systems, however, the cost constraints are often required to be satisfied at each round in any instance even in the worst case, which hinders the deployment of these constrained/conservative RL policies. This dissertation provide

1.3 Machine Learning With Domain Knowledge

The dissertation also contributes to machine learning with domain knowledge whose preliminaries are given as below.

1.3.1 Preliminaries of Knowledge Informed Learning

Domain knowledge informed machine learning is rapidly emerging as a broad paradigm that incorporates domain knowledge, either directly or indirectly, to augment the purely data-driven approach and better accomplish a machine learning task. We provide a summary of how domain knowledge is integrated with machine learning [396].

- *Training Dataset.* A straightforward approach to utilizing domain knowledge is to generate (sometimes synthetic) data and enlarge the otherwise limited training dataset. For example, based on the simple knowledge of image invariance, cropping[154], scaling[436], flipping[59] and many other image pre-processing methods have been used to augment the training data for image classification tasks. As another example, in reinforcement learning (e.g., robot control and autonomous driving) where initial pre-training is crucial to avoid arbitrarily bad decisions in the real world, simulated environments can be built based on domain knowledge, providing simulations or demonstrations to generate training data [156, 193]. Additionally, generative models constructed based on specific knowledge have been shown useful for increasing training data to improve model performance and robustness [154, 177].

- *Hypothesis Set.* The goal of a machine learning task is to search for an optimal hypothesis that correctly expresses the relationships between input and output. To

reduce the training complexity, the target hypothesis set (decided by, e.g., different neural architectures) should contain the optimal hypothesis and preferably be small enough. Thus, domain knowledge can be employed for hypothesis set selection. For example, [99] makes use of the prior knowledge from the existing neural architectures to design new architectures (and hence, new hypothesis sets) for DNNs. As implicit domain knowledge, long short-term memory recurrent neural networks are commonly used for time series prediction [177]. Also, the structure of a knowledge graph helps to determine the hypothesis set of graph learning [289, 56], while [385] maps the domain knowledge represented in propositional logic into neural networks.

- *Model Training.* Domain knowledge can be integrated, either implicitly or explicitly, with the model training procedure in various ways. First, domain knowledge can assist with the initialization of training. For example, [332] provides a case-based method to initialize genetic algorithms (i.e., generating the initial population based on different cases), while [199, 230, 198] initialize neural network training with various domain knowledge such as label co-occurrence and decision trees. Second, domain knowledge can be used to better tune the hyper-parameters [52, 389, 285, 50]. In [52], implicit knowledge from previous training is incorporated to improve hyper-parameter tuning, and [389] extracts knowledge from multiple datasets to determine the most important hyper-parameters. In addition, a more explicit way to integrate domain knowledge is to directly modify the training objective function (i.e., risk function) based on rigorous characterization of the model output [396]. For example, in [302], the knowledge of constraints is incorporated into neural networks expressing the knowledge based loss

by the ReLu function. For another example, when learning to optimally schedule transmissions for rate maximization in multi-user wireless networks, the communication channel capacity can be added as domain knowledge to the standard label-based loss to guide scheduling decisions; in physics, the analytical expression of a partial differential equation can be utilized as domain knowledge on top of labeled data to better learn the solution to the equation given different inputs

Such integration of explicit and rigorous domain knowledge can significantly benefit machine learning tasks (e.g., fewer labels needed than otherwise). Thus, it is crucial and being actively studied in informed machine learning [396, 408], which is also the focus of our work. Note that using domain knowlege to generate pseudo labeled data to augment the training dataset is a special case of integrating domain knowledge into the training risk function (i.e., the knowledge-based risk is the same as the data-based risk, except that its labels are generated based on domain knowledge).

- *Final Hypothesis.* Domain knowledge can also be used for consistency check on the final learnt hypothesis or model [396]. For example, [216] employs physics domain knowledge to construct the final model, [318] builds simulators to validate results of learned model, and [145] leverages semantic consistency is used to refine the predicted probabilities.

In this dissertation, we utilize the methodology of informed machine learning for online decision making problems, provide a theoretical analysis of informed machine learning to understand the benefits of domain knowledge.

1.3.2 Algorithm Unrolling

Many aspects in this dissertation rely on the techniques of algorithm unrolling which integrate ML into traditional algorithmic frameworks for better trustworthiness, better generalization, lower sampling complexity and/or smaller ML model size [6, 96, 227, 300, 272]. Algorithm unrolling has been used for sparse coding [183], signal and image processing [300, 255], and solving inverse problems [226] and ordinary differential equations (ODEs) [94]. Also, algorithm unrolling is important for Learning to Optimize (L2O) [96, 406]. These studies have their own challenges orthogonal to our problem where the key challenge is the lack of complete offline information. Thus, in this dissertation, we apply the idea of algorithm unrolling to online decision making problems with the aim to achieve guaranteed trustworthiness/robustness and better generalization than generic RL-based optimizers to directly obtain end solutions [17, 134, 228].

1.4 Dissertation Outline

The dissertation focuses on learning augmented algorithms for online decision making problems in computing systems and include the following aspects.

In Chapter 2, the dissertation considers bandit problems with imperfect context and designs robust algorithms to optimize the worst-case performances. Bandit decision problems model many practical problems in recommendation systems, mobile health, cloud resource provisioning, etc. In many cases, however, the context observed by the agent is imperfect and uncertain. This presents challenges for robust decision making. We design robust bandit learning algorithms for different robustness objectives with provably bounded

regrets. The regret analysis shows that the proposed robust bandit algorithms achieves the optimal worst-case performances as time goes to infinity.

In Chapter 3, we design Learning-augmented algorithms for online optimization/control with known dynamic models. The considered settings including smoothed online optimization and online control, is important to numerous applications such as robot tracking, datacenter resource provisioning, battery management for EV charging station, online renewable aggregation, etc. ML algorithms have been increasingly applied in various online decision problems to achieve high average performance, but they lack worst-case performance guarantee (competitiveness). We addresses this challenge by a learning-augmented algorithm which minimizes the average cost under the provable anytime performance guarantee comparing with a trusted prior for each instance. Our results formally highlight the tradeoff between the worst-case and average cost performance.

In Chapter 4, we design learning-augmented algorithms for online decision making with unknown dynamic models. The goal is to optimize the average reward while guaranteeing the anytime cost constraints comparing with a policy prior. This problem models many real decision-making problems in mission critical systems without an exact dynamic model. The problem is challenging since only the information regarding the selected actions is fed back. With some assumptions on the dynamics, we propose the Anytime-Constrained Reinforcement Learning to provably guarantee the anytime constraints.

In Chapter 5, we consider online optimization with budget constraints which models problems including online virtual machine resource allocation, resource management in wireless networks, and data center server provisioning. The problem is challenging due

to the strict budget constraints and the online nature. Existing online learning algorithms like dual mirror descent have bounded average and worst-case performance only when the contexts are independently distributed and the episodic length is long enough. Existing competitive algorithms like CR-Pursuit have worst-case guarantee but are too conservative to achieve satisfactory performance. This dissertation gives a deep unrolling model with better average performance under general assumptions about context distribution and episode length.

In Chapter 6, We give a rigorous analysis about the role of domain knowledge in machine learning and how the quality of domain knowledge affect the generalization. We define the metrics of the imperfectness of labels and domain knowledge and show the dependency of the generalization bound on the imperfectness. Also, the analysis of sampling complexity for informed ML establishes a quantitative equivalence between domain knowledge and labeled samples. Based on the analysis, we provide a generalized training objective to better exploit the benefits of knowledge and balance the label and knowledge imperfectness. The study highlights the two benefits of do- main knowledge in informed learning — regularizing the label-based supervision and supplementing the labeled samples.

Chapter 2

Robust Bandit Decision Making With Imperfect Context

2.1 Introduction

Contextual bandits [280, 112] concern online learning scenarios such as recommendation systems [239], mobile health [236], cloud resource provisioning [89], wireless communications [344], in which arms (a.k.a., actions) are selected based on the underlying context to balance the tradeoff between exploitation of the already learnt knowledge and exploration of uncertain arms [40, 39, 73, 118].

The majority of the existing studies on contextual bandits [112, 388, 344] assume that a perfectly accurate context is known before each arm selection. Consequently, as long as the agent learns increasingly more knowledge about reward, it can select arms with lower and lower average regrets. In many cases, however, the perfect (or true) context is not

available to the agent prior to arm selection. Instead, the true context is revealed after taking an action at the end of each round [223], but can be predicted using predictors, such as time series prediction[72, 164], to facilitate the agent's arm selection. For example, in wireless communications, the channel condition is subject to various attenuation effects (e.g., path loss and small-scale multi-path fading), and is critical context information for the transmitter configuration such as modulation and rate adaption (i.e., arm selection) [175, 344]. But, the channel condition context is predicted and hence can only be coarsely known until the completion of transmission. For another example, the exact workload arrival rate is crucial context information for cloud resource management, but cannot be known until the workload actually arrives. Naturally, context prediction is subject to prediction errors. Moreover, it can also open a new attack surface — an outside attacker may adversarially modify the predicted context. For example, a recent study [100] shows that the energy load predictor in smart grid can be adversarially attacked to produce load estimates with higher-than-usual errors. More motivating examples are provided in [419]. In general, imperfectly predicted and even adversarially presented context is very common in practice.

As motivated by practical problems, we consider a bandit setting where the agent receives imperfectly predicted context and selects an arm at the beginning of each round and the context is revealed after arm selection. We focus on robust arm optimization given imperfect context, which is as crucial as robust reward function estimation or exploration in contextual bandits [138, 305, 446]. Concretely, with imperfect context, our goal is to select arms online in a robust manner to optimize the worst-case performance in a neighborhood domain with the received imperfect context as center and a defense budget as radius. In

this way, the robust arm selection can defend against the imperfect context error (from either context prediction error or adversarial modification) constrained by the budget.

Importantly and interestingly, given imperfect context, maximizing the worst-case reward (referred to as type-I robustness objective) and minimizing the worst-case regret (referred to as type-II robustness objective) can lead to different arms, while they are the same under the setting of perfect context [344, 239, 365]. Given imperfect context, the strategy for type-I robustness is more conservative than that for type-II robustness in terms of reward. The choice of the robustness objective depends on applications. For example, some safety-aware applications [378, 159] intend to avoid extremely low reward, and thus type-I objective is suitable for them. Other applications [239, 90, 184] focus on preventing large sub-optimality of selected arms, and type-II objective is more appropriate. As a distinction from other works on robust optimization of bandits [66, 222, 307], we highlight the difference of the two types of robustness objectives.

We derive two algorithms — MaxMinUCB (Maximize Minimum UCB), which maximizes the worst-case reward for type-I objective, and MinWD (Minimize Worst-case Degradation), which minimizes the worst-case regret for type-II objective. The challenge of algorithm designs is that the agent has no access to exact knowledge of reward function but the estimated counterpart based on history collected data. Thus, in our design, MaxMinUCB maximizes the lower bound of reward, while MinWD minimizes the upper bound of regret.

We analyze the robustness of MaxMinUCB and MinWD by deriving both regret and reward bounds, compared to a strong oracle that knows the true context for arm selection as well as the exact reward function. Importantly, our results show that, while a linear

regret term exists for both MaxMinUCB and MinWD due to imperfect context, the added linear regret term is actually the same as the amount of regret incurred by respectively optimizing type-I and type-II objectives with perfect knowledge of the reward function. This implies that as time goes on, MaxMinUCB and MinWD will asymptotically approach the corresponding optimized objectives from the reward and regret views, respectively.

Finally, we apply MaxMinUCB and MinWD to the problem of online edge datacenter selection and run synthetic simulations to validate our theoretical analysis.

2.2 Related Work

Contextual Bandits. Linear contextual bandit learning is considered in LinUCB by [239]. . The study [4] improves the regret analysis of linear contextual bandit learning, while the studies [8, 9] solve this problem by Thompson sampling and give a regret bound. There are also studies to extend the algorithms to general reward functions like non-linear functions, for which kernel method is exploited in GP-UCB [369], Kernel-UCB [388], IGP-UCB and GP-TS [109, 124]. Nonetheless, a standard assumption in these studies is that perfect context is available for arm selection, whereas imperfect context is common in many practical applications [222].

Adversarial Bandits and Robustness. The prior studies on adversarial bandits [41, 214, 18, 271] have primarily focused on that the adversary maliciously presents rewards to the agent or directly injects errors in rewards. Moreover, many studies [38, 162] consider the best constant policy throughout the entire learning process as the oracle, while in our setting the best arm depends on the true context at each round.

Recently, robust bandit algorithms have been proposed for various adversarial settings. Some focus on robust reward estimation and exploration [18, 184, 138], and others train a robust or distributionally robust policy [411, 380, 362, 361]. Our study differs from the existing adversarial bandits by seeking two different robust algorithms given imperfect (and possibly adversarial) context.

Optimization and Bandits With Imperfect Context. [331] considers online optimization with predictable sequences and [205] focuses on adaptive online optimization competing with dynamic benchmarks. Besides, [88, 211] study the robust optimization of mini-max regret. These studies assume perfectly known cost functions without learning. A recent study [66] considers Bayesian optimization and aims at identifying a worst-case good input region with input perturbation (which can also model a perturbed but fixed environment/context parameter). The study [398] considers the linear bandit where certain context features are hidden, and uses iterative methods to estimate hidden contexts and model parameters. The relevant papers [222] and [307] consider robust Bayesian optimizations where context distribution information is imperfectly provided, and propose to maximize the worst-case expected reward for distributional robustness. Although the objective of MaxMinUCB in our paper is similar to the robust optimization objectives in the two papers, we additionally derive a lower bound for the true reward in our analysis, which provides another perspective on the robustness of arm selection. More importantly, considering that the objectives in the two relevant papers are equivalent to minimizing a pseudo robust regret, we propose MinWD and derive an upper bound for the incurred true regret.

2.3 Problem Formulation

In this section, we give the problem formulation for the contextual bandit with imperfect context. We model the context imperfectness captured by an error budget. Also, we provide the preliminaries for reward estimation by kernel ridge regression,

Assume that at the beginning of round t, the agent receives imperfect context $\hat{x}_t \in \mathcal{X}$ which is exogenously provided and not necessarily the true context x_t. Given the imperfect context $\hat{\mathbf{x}}_t \in \mathcal{X}$ and an arm set $\mathcal{A}$, the agent selects an arm $a_t \in \mathcal{A}$ for round t. Then, the reward y_t along with the true context x_t is revealed to the agent at the end of round t. Assume that $\mathcal{X} \times \mathcal{A} \subseteq \mathbb{R}^d$, and we use $x_{a_t,t}$ to denote the d-dimensional concatenated vector $[x_t, a_t]$.

The reward y_t received by the agent in round t is jointly decided by the true context x_t and selected arm a_t, and can be expressed as follows

$$y_t = f(x_t, a_t) + n_t, \tag{2.1}$$

where $f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{X}$ is the context domain, and n_t is the noise term. We assume that the reward function f belongs to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ generated by a kernel function $k : (\mathcal{X} \times \mathcal{A}) \times (\mathcal{X} \times \mathcal{A}) \to \mathbb{R}$. In this RKHS, there exists a mapping function $\phi : (\mathcal{X} \times \mathcal{A}) \to \mathcal{H}$ which maps context and arm to their corresponding feature in $\mathcal{H}$. By reproducing property, we have $k([x, a], [x', a']) = \langle \phi(x, a), \phi(x', a') \rangle$ and $f(x, a) = \langle \phi(x, a), \theta \rangle$ where θ is the representation of function $f(\cdot, \cdot)$ in $\mathcal{H}$. Further, as commonly considered in the bandit literature [365, 239], the noise n_t follows b-sub-Gaussian distribution for a constant $b \geq 0$, i.e. conditioned on the filtration $\mathcal{F}_{t-1} = \{x_\tau, y_{a,\tau}, a_\tau, \tau = 1, \cdots, t-1\}$, $\forall \sigma \in \mathbb{R}$, $\mathbb{E}[e^{\sigma n_t} | \mathcal{F}_{t-1}] \leq \exp\left(\frac{\sigma^2 b^2}{2}\right)$.

Without knowledge of reward function f, bandit algorithms are designed to decide an arm sequence $\{a_t, t = 1, \cdots, T\}$ to minimize the cumulative regret

$$R_T = \sum_{t=1}^{T} f(x_t, A^*(x_t)) - f(x_t, a_t), \tag{2.2}$$

where $A^*(x_t) = \arg\max_{a \in \mathcal{A}} f(x_t, a)$ is the oracle-optimal arm at round t given the true context x_t. When the received contexts are perfect, i.e. $\hat{x}_t = x_t$, minimizing the cumulative regret is equivalent to maximizing the cumulative reward $F_T = \sum_{t=1}^{T} f(x_t, a_t)$.

2.3.1 Context Imperfectness

The context error can come from a variety of sources, including imperfect context prediction algorithms and adversarial corruption [222, 100] on context. We simply use context error to encapsulate all the error sources without further differentiation. We assume that context error $\|x_t - \hat{x}_t\|$, where $\|\cdot\|$ is a certain norm [66], is less than Δ. Also, Δ is referred to as the defense *budget* and can be considered as the level of robustness/safeguard that the agent intends to provide against context errors: with a larger Δ, the agent wants to make its arm selection robust against larger context errors (at the possible expense of its reward). A time-varying error budget can be captured by using Δ_t. Denote the neighborhood domain of context x as $\mathcal{B}_\Delta(x) = \{y \in \mathcal{X} \mid \|y - x\| \leq \Delta\}$. Then, we have the true context $x_t \in \mathcal{B}_\Delta(\hat{x}_t)$, where $\hat{x}_t$ is available to the agent.

2.3.2 Reward Estimation

Reward estimation is critical for arm selection. Kernel ridge regression, which is widely used in contextual bandits [365] serves as the reward estimation method in our

algorithm designs. By kernel ridge regression, the estimated reward given arm a and context x is expressed as

$$\hat{f}_t(x,a) = \mathbf{k}_t^T(x,a)(\mathbf{K}_t + \lambda \mathbf{I})^{-1}\mathbf{y}_t \tag{2.3}$$

where $\mathbf{I}$ is an identity matrix, $\mathbf{y}_t \in \mathbb{R}^{t-1}$ contains the history of y_τ, $\mathbf{k}_t(x,a) \in \mathbb{R}^{t-1}$ contains $k([x,a],[x_\tau,a_\tau])$, and $\mathbf{K}_t \in \mathbb{R}^{(t-1)\times(t-1)}$ contains $k([x_\tau,a_\tau],[x_{\tau'},a_{\tau'}])$, for τ,$\tau'\in\{1,\cdots,t-1\}$.

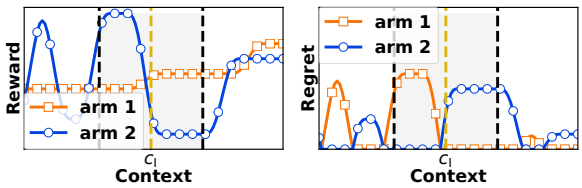
The confidence width of kernel ridge regression is given in the following concentration lemma followed by a definition of reward UCB.

Lemma 1 (Concentration of Kernel Ridge Regression)**.** *Assume that the reward function $f(x,a)$ satisfies $|f(x,a)| \leq B$, the noise n_t satisfies a sub-Gaussian distribution with parameter b, and kernel ridge regression is used to estimate the reward function. With a probability of at least $1-\delta, \delta \in (0,1)$, for all $a \in \mathcal{A}$ and $t \in \mathbb{N}$, the estimation error satisfies $|\hat{f}_t(x,a) - f(x,a)| \leq h_t s_t(x,a)$, where $h_t = \sqrt{\lambda}B + b\sqrt{\gamma_t - 2\log(\delta)}$, $\gamma_t = \log\det(\mathbf{I}+\mathbf{K}_t/\lambda) \leq \bar{d}\log(1+\frac{t}{\bar{d}\lambda})$ and $\bar{d}$ is the rank of $\mathbf{K}_t$. Let $\mathbf{V}_t = \lambda\mathbf{I} + \sum_{s=1}^{t}\phi(x,a)\phi(x,a)^\top$, the squared confidence width is given by $s_t^2(x,a) = \phi(x,a)^\top\mathbf{V}_{t-1}^{-1}\phi(x,a) = \frac{1}{\lambda}k([x,a],[x,a]) - \frac{1}{\lambda}\mathbf{k}_t(x,a)^T(\mathbf{K}_t + \lambda\mathbf{I})^{-1}\mathbf{k}_t(x,a)$.*

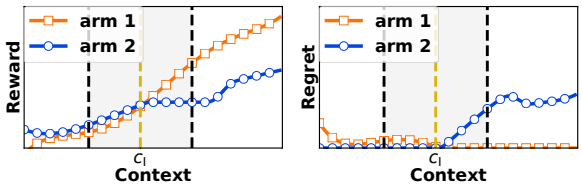
Definition 2. *Given arm $a \in \mathcal{A}$ and context $x \in \mathcal{X}$, the reward UCB (Upper Confidence Bound) is defined as $U_t(x,a) = \hat{f}_t(x,a) + h_t s_t(x,a)$.*

The next lemma shows that the term $s_t(x_t,a_t)$ has a vanishing impact on regret over time.

Lemma 3. *The sum of confidence widths given x_t for $t \in \{1,\cdots,T\}$ satisfies $\sum_{t=1}^{T} s_t^2(x_t,a_t) \leq 2\gamma_T$, where $\gamma_T = \log\det(\mathbf{I}+\mathbf{K}_T/\lambda) \leq \bar{d}\log(1+\frac{T}{\bar{d}\lambda})$ and $\bar{d}$ is the rank of $\mathbf{K}_T$.*



(a) Type-I Robustness



(b) Type-II Robustness

Figure 2.1: Illustration of reward and regret functions that Type-I and Type-II robustness objectives are suitable for, respectively. The golden dotted vertical line represents the imperfect context c_{I}, and the gray region represents the defense region $\mathcal{B}_{\Delta}(c_{\mathrm{I}})$.

Then, we give the definition of *UCB-optimal* arm which is important in our algorithm designs.

Definition 4. *Given context* $x \in \mathcal{X}$*, the* UCB-optimal *arm is* $A_t^{\dagger}(x) = \arg\max_{a \in \mathcal{A}} U_t(x, a)$.

Note that if the received contexts are perfect, i.e. $\hat{x}_t = x_t$, the standard contextual UCB strategy selects arm at round t as $A_t^{\dagger}(x_t)$. Under the cases with imperfect context, a naive policy (which we call SimpleUCB) is simply oblivious of context errors, i.e. the agent selects the UCB-optimal arm regarding imperfect context $\hat{x}_t$, denoted as $a_t^{\dagger} = A_t^{\dagger}(\hat{x}_t)$, by simply viewing the imperfect context $\hat{x}_t$ as true context. Nonetheless, if we want to guarantee the arm selection performance even in the worst case, robust arm selection that accounts for context errors is needed.

2.4 Robustness Objectives

In the existing bandit literature such as [41, 189, 239], maximizing the cumulative reward is equivalent to minimizing the cumulative regret, under the assumption of perfect context for arm selection. In this section, we will show that maximizing the worst-case reward is equivalent to minimizing a *pseudo* regret and is different from minimizing the worst-case true regret.

2.4.1 Type-I Robustness

With imperfect context, one approach to robust arm selection is to maximize the worst-case reward. With perfect knowledge of reward function, the oracle arm that maximizes the worst-case reward at round t is

$$\bar{a}_t = \arg\max_{a\in\mathcal{A}} \min_{x\in\mathcal{B}_\Delta(\hat{x}_t)} f(x,a). \tag{2.4}$$

For analysis in the following sections, given $\bar{a}_t$, the corresponding context for the worst-case reward is denoted as

$$\bar{x}_t = \arg\min_{x\in\mathcal{B}_\Delta(\hat{x}_t)} f\left(x,\bar{a}_t\right), \tag{2.5}$$

and the resulting optimal worst-case reward is denoted as

$$MF_t = f\left(\bar{x}_t,\bar{a}_t\right). \tag{2.6}$$

Next, Type-I robustness objective is defined based on the difference $\sum_{t=1}^{T} MF_t - F_T$, where $F_T = \sum_{t=1}^{T} f(x_t,a_t)$ is the actual cumulative reward.

Definition 5. *If, with an arm selection strategy* $\{a_1, \cdots, a_T\}$*, the difference between the optimal cumulative worst-case reward and the cumulative true reward* $\sum_{t=1}^{T} MF_t - F_T$ *is sub-linear with respect to* T*, then the strategy achieves Type-I robustness.*

If an arm selection strategy achieves Type-I robustness, the lower bound for the true reward $f(x_t, a_t)$ approaches the optimal worst-case reward MF_t in the defense region as t increases. Therefore, a strategy achieving type-I robustness objective can prevent very low reward. For example, in Fig. 2.1(a), arm 1 is the one that maximizes the worst-case reward, which is not necessarily optimal but always avoids extremely low reward under any context in the defense region.

Note that maximizing the worst-case reward is equivalent to minimizing the robust regret defined in [222], which is written using our formulation as

$$\bar{R}_T = \sum_{t=1}^{T} \min_{x \in \mathcal{B}_\Delta(\hat{x}_t)} f(x, \bar{a}_t) - \min_{x \in \mathcal{B}_\Delta(\hat{x}_t)} f(x, a_t). \tag{2.7}$$

However, this robust regret is a *pseudo* regret because the rewards of oracle arm $\bar{a}_t$ and selected arm a_t are compared under different contexts (i.e., their respective worst-case contexts), and it is not an upper or lower bound of the true regret R_T. To obtain a robust regret performance, we need to define another robustness objective based on the true regret.

2.4.2 Type-II Robustness

To provide robustness for the regret with imperfect context, we can minimize the cumulative worst-case regret, which is expressed as

$$\tilde{R}_T = \sum_{t=1}^{T} \max_{x \in \mathcal{B}_\Delta(\hat{x}_t)} \left[f(x, A^*(x)) - f(x, a_t) \right]. \tag{2.8}$$

Clearly, the true regret $R_T \leq \tilde{R}_T$, and minimizing the worst-case regret is equivalent to minimizing an upper bound for the true regret. Define the instantaneous regret function with respect to context x and arm a as $r(x, a) = f(x, A^*(x)) - f(x, a)$. Since given the reward function the optimization is decoupled among different rounds, the robust oracle arm to minimize the worst-case regret at round t is

$$\tilde{a}_t = \arg\min_{a \in \mathcal{A}} \max_{x \in \mathcal{B}_t(\hat{x}_t)} r(x, a). \tag{2.9}$$

For analysis in the following sections, given $\tilde{a}_t$, the corresponding context for the worst-case regret is denoted as

$$\tilde{x}_t = \arg\max_{x \in \mathcal{B}_\Delta(\hat{x}_t)} r(x, \tilde{a}_t), \tag{2.10}$$

and the resulting optimal worst-case regret is

$$MR_t = r(\tilde{x}_t, \tilde{a}_t). \tag{2.11}$$

Now, we can give the definition of Type-II robustness as follows.

Definition 6. *If, with an arm selection strategy $\{a_1, \cdots, a_T\}$, the difference between the cumulative true regret and the optimal cumulative worst-case regret $R_T - \sum_{t=1}^{T} MR_t$ is sub-linear with respect to T, then the strategy achieves Type-II robustness.*

If an arm selection strategy achieves Type-II robustness, as time increases, the upper bound for the true regret $r(x_t, a_t)$ also approaches the optimal worst-case regret MR_t. Hence, a strategy achieving type-II robustness objective can prevent a high regret. As shown in Fig. 2.1(b), arm 1 is selected by minimizing the worst-case regret, which is a robust arm selection because the regret of arm 1 under any context in the defense region is not too high.

2.4.3 Comparison of Two Robustness Objectives

The two types of robustness correspond to the algorithms maximizing the worst-case reward and minimizing the worst-case regret, respectively. In many cases, they result in different arm selections. Take the two scenarios in Fig. 2.1 as examples. In the scenario of Fig. 2.1(a), given the defense region, arm 1 is selected by maximizing the worst-case reward and arm 2 is selected by minimizing the worst-case regret. It can be observed that the worst-case regrets of the two arms are very close, but the worst-case reward of arm 2 is much lower than that of arm 1. Thus, the strategy of maximizing the worst-case reward is more suitable for this scenario. Differently, in the scenario of Fig. 2.1(b), arm 2 is selected by maximizing the worst-case reward and arm 1 is selected by minimizing the worst-case regret. Since the worst-case rewards of the two arms are very close and the worst-case regret of arm 2 is much larger than arm 1, it is more suitable to minimize the worst-case regret.

2.5 Robust Bandit Arm Selection

In this section, we propose two robust arm selection algorithms: (1) MaxMinUCB (Maximize Minimum Upper Confidence Bound), which aims to maximize the minimum reward (Type-I robustness objective); and (2) MinWD (Minimize Worst-case Degradation), which aims to minimize the maximum regret (Type-II robustness objective). We derive the regret and reward bounds for both algorithms and the proofs are available in [419]. The regret and reward bounds show that both algorithms achieve the corresponding types of robustness.

Algorithm 1 Robust Arm Selection with Imperfect Context

Require: Context error budget Δ

for $t = 1, \cdots, T$ **do**

Receive imperfect context $\hat{x}_t$.

Select arm a_t^{I} to solve Eqn. (2.12) in MaxMinUCB; or select arm a_t^{II} to solve Eqn. (2.16) in MinWD

Observe the true context x_t and the reward y_t.

end for

2.5.1 MaxMinUCB: Maximize Minimum UCB

To achieve type-I robustness, MaxMinUCB in Algorithm 1 selects an arm a_t^{I} by maximizing the minimum UCB within the defense region $\mathcal{B}_\Delta(\hat{x}_t)$:

$$a_t^{\mathrm{I}} = \arg\max_{a \in \mathcal{A}} \min_{x \in \mathcal{B}_\Delta(\hat{x}_t)} U_t(x, a). \tag{2.12}$$

The context that attains the minimum UCB in Eqn.(2.12) is $x_t^{\mathrm{I}} = \min_{x \in \mathcal{B}_\Delta(\hat{x}_t)} U_t\left(x, a_t^{\mathrm{I}}\right)$.

Analysis

The next theorem gives a lower bound of the cumulative true reward of MaxMinUCB in terms of the optimal worst-case reward and a sub-linear term.

Theorem 2.5.1. *If* MaxMinUCB *is used to select arms with imperfect context, then for any true contexts $x_t \in \mathcal{B}_\Delta(\hat{x}_t)$ at round $t, t = 1, \cdots, T$, with a probability of $1 - \delta, \delta \in (0, 1)$, we have the following lower bound on the worst-case cumulative reward*

$$F_T \geq \sum_{t=1}^{T} MF_t - 2h_T\sqrt{2T\bar{d}\log(1 + \frac{T}{\bar{d}\lambda})} \tag{2.13}$$

where MF_t is the optimal worst-case reward in Eqn. (2.6)*, $\bar{d}$ is the rank of $\mathbf{K}_t$ and h_T is given in Lemma 1.*

Remark 7. *Theorem 2.5.1 shows that by* MaxMinUCB*, the difference between the optimal cumulative worst-case reward and the cumulative true reward is sub-linear and thus effectively achieves Type-I robustness according to Definition 5. This means that the reward by* MaxMinUCB *has a bounded sub-linear gap compared to the optimal worst-case reward $\sum_{t=1}^{T} MF_t$ obtained with perfect knowledge of the reward function.* □

We are also interested in the cumulative true regret of MaxMinUCB which is given in the following corollary.

Corollary 8. *If* MaxMinUCB *is used to select arms with imperfect context, then for any true contexts $x_t \in \mathcal{B}_\Delta(\hat{x}_t)$ at round $t, t = 1, \cdots, T$, with a probability of $1-\delta, \delta \in (0,1)$, we have the following bound on the cumulative true regret defined in Eqn.* (2.2)*:*

$$R_T \leq \sum_{t=1}^{T} \overline{MR}_t + 2h_T \sqrt{2T\bar{d}\log(1+\frac{T}{\bar{d}\lambda})} \tag{2.14}$$

where $\overline{MR}_t = \max_{x \in \mathcal{B}_\Delta(\hat{x}_t)} f(x, A^*(x)) - MF_t$, MF_t is the optimal worst-case reward in Eqn. (2.6) *.*

Remark 9. *Corollary 8 shows that the worst-case regret by* MaxMinUCB *can be quite larger than the optimal worst-case regret MR_t given in Eqn.* (2.11) *(Type-II robustness objective). Actually, despite being robust in terms of rewards, arms selected by* MaxMinUCB *can still have very large regret as shown in Fig. 2.1(b). Thus, to achieve type-II robustness, it is necessary to develop an arm selection algorithm that minimizes the worst-case regret.*

2.5.2 MinWD: Minimize Worst-Case Degradation

MinWD is designed to asymptotically minimize the worst-case regret. Without the oracle knowledge of reward function, MinWD performs arm selection based on the upper bound of regret. Denote $D_a(x) = U_t\left(x, A_t^\dagger(x)\right) - U_t(x,a)$ referred to as UCB *degradation* at context x. By Lemma 1, the instantaneous true regret can be bounded as

$$\begin{aligned} r(x_t, a_t) &\leq [D_{a_t}(x_t) + 2h_t s_t(x_t, a_t)] \\ &\leq \overline{D}_{a_t} + 2h_t s_t(x_t, a_t), \end{aligned} \tag{2.15}$$

where $\overline{D}_{a_t} = \max_{x \in \mathcal{B}_\Delta(\hat{x}_t)} D_a(x)$ is called the *worst case degradation*, and $2h_t s_t(x_t, a_t)$ has a vanishing impact by Lemma 3. Thus, to minimize worst-case regret, MinWD minimizes its upper bound $\overline{D}_{a_t}$ excluding the vanishing term $2h_t s_t(x_t, a_t)$, i.e.

$$a_t^{\mathrm{II}} = \min_{a \in \mathcal{A}} \max_{\mathbf{x} \in \mathcal{B}_\Delta(\hat{x}_t)} \left\{ U_t\left(x, A_t^\dagger(x)\right) - U_t(x,a) \right\}. \tag{2.16}$$

The context that attains the worst case in Eqn. (2.16) is written as $x_t^{\mathrm{II}} = \arg\max_{x \in \mathcal{B}_\Delta(\hat{x}_t)} D_{a_t^{\mathrm{II}}}(x)$.

Analysis

Given arm a_t^{II} selected by MinWD, the next lemma gives an upper bound of worst-case degradation.

Lemma 10. *If* MinWD *is used to select arms with imperfect context, then for each* $t = 1, 2, \cdots, T$*, with a probability at least* $1 - \delta, \delta \in (0,1)$*, we have*

$$\overline{D}_{a_t^{\mathrm{II}}, t} \leq MR_t + 2h_t s_t\left(\dot{x}_t, A_t^\dagger(\dot{x}_t)\right), \tag{2.17}$$

where MR_t *is the optimal worst-case regret defined in Eqn.* (2.11)*,* $\dot{x}_t = \arg\max_{x \in \mathcal{B}_\Delta(\hat{x}_t)} D_{\tilde{a}_t}(x)$ *is the context that maximizes the degradation given the arm* $\tilde{a}_t$ *defined for the optimal worst-case regret in Eqn.* (2.10)*.*

Then, in order to show that $\overline{D}_{a_t^{\mathrm{II}},t}$ approaches MR_t, we need to prove that $2h_t s_t\left(\dot{x}_t, A_t^{\dagger}\left(\dot{x}_t\right)\right)$ vanishes as t increases. But, this is difficult because the considered sequence $\left\{\dot{x}_t, A_t^{\dagger}\left(\dot{x}_t\right)\right\}$ is different from the actual sequence of context and selected arms $\left\{x_t, a_t^{\mathrm{II}}\right\}$ under MinWD. To circumvent this issue, we first introduce the concept of $\epsilon-$ covering [412]. Denote $\Phi = \mathcal{X} \times \mathcal{A}$ as the context-arm space. If a finite set Φ_ϵ is an $\epsilon-$ covering of the space Φ, then for each $\varphi \in \Phi$, there exists at least one $\bar{\varphi} \in \Phi_\epsilon$ satisfying $\|\varphi - \bar{\varphi}\|_2 \leq \epsilon$. Denote $\mathcal{C}_\epsilon\left(\bar{\varphi}\right) = \{\varphi \mid \|\varphi - \bar{\varphi}\|_2 \leq \epsilon\}$ as the cell with respect to $\bar{\varphi} \in \Phi_\epsilon$. Since the dimension of the entries in Φ is d, the size of the Φ_ϵ is $|\Phi_\epsilon| \sim O\left(\frac{1}{\epsilon^d}\right)$. Besides, we assume the mapping function ϕ is Lipschitz continuous, i.e. $\forall x, y \in \Phi$, $\|\phi(x) - \phi(y)\| \leq L_\phi \|x - y\|$. Next, we prove the following proposition to bound the sum of confidence widths.

Proposition 11. *Let $\mathcal{X}_T = \{x_{a_1,1}, \cdots, x_{a_T,T}\}$ be the sequence of true contexts and selected arms by bandit algorithms and $\dot{\mathcal{X}}_T = \{\dot{x}_{\dot{a}_1,1}, \cdots, \dot{x}_{\dot{a}_T,T}\}$ be the considered sequence of contexts and actions. Suppose that both $x_{a_t,t}$ and $\dot{x}_{\dot{a}_t,t}$belong to Φ. Besides, with an $\epsilon-$ covering $\Phi_\epsilon \subseteq \Phi$, $\epsilon > 0$, there exists $\kappa \geq 0$ such that two conditions are satisfied: First, $\forall \bar{\varphi} \in \Phi_\epsilon$, $\exists t \leq \left\lceil \kappa/\epsilon^d \right\rceil$ such that $x_{a_t,t} \in \mathcal{C}_\epsilon\left(\bar{\varphi}\right)$. Second, if at round t, $x_{a_t,t} \in \mathcal{C}_\epsilon\left(\bar{\varphi}\right)$ for some $\bar{\varphi} \in \Phi_\epsilon$, then $\exists t \leq t' < t + \left\lceil \kappa/\epsilon^d \right\rceil$ such that $x_{a_t',t'} \in \mathcal{C}_\epsilon\left(\bar{\varphi}\right)$. If the mapping function ϕ is Lipschitz continuous with constant L_ϕ, the sum of squared confidence widths is bounded as*

$$\sum_{t=1}^{T} s_t^2\left(\dot{x}_{\dot{a}_t,t}\right) \leq \sqrt{T}\left(4\tilde{d}\log\left(1+\frac{T}{\tilde{d}\lambda}\right)+\frac{1}{\lambda}\right)+\frac{8L_\phi^2\kappa^{2/d}}{\lambda}T^{1-1/d},$$

where d is the dimension of $x_{a_t,t}$, $\tilde{d}$ is the effective dimension defined in the proof, $s_t^2\left(\dot{x}_{\dot{a}_t,t}\right) = \phi(\dot{x}_{\dot{a}_t,t})^\top \mathbf{V}_{t-1}^{-1}\phi(\dot{x}_{\dot{a}_t,t})$ and $\mathbf{V}_t = \lambda\mathbf{I} + \sum_{s=1}^{t}\phi(x_{a_s,s})\phi(x_{a_s,s})^\top$.

Remark 12. *The conditions in Proposition 11 guarantee that the time interval between the events that true context-arm feature lies in the same cell is not larger than $\left\lceil \kappa/\epsilon^d \right\rceil$, which is*

proportional to the size of the ϵ-covering $|\Phi_\epsilon|$. That means, similar contexts and selected arms occur in the true sequence repeatedly if T is large enough. If contexts are sampled from a bounded space $\mathcal{X}$ with some distribution, then similar contexts will occur repeatedly. Also, note that the arm in our considered sequence $A_t^\dagger(\dot{x}_t)$ is the UCB-optimal arm, which becomes close to the optimal arm for $\dot{x}_t$ if the confidence width is sufficiently small. Hence, there exists some context error budget sequence $\{\Delta_t\}$ such that, starting from a certain round T_0, the two conditions are satisfied. The two conditions in Proposition 11 are mainly for theoretical analysis of MinWD.

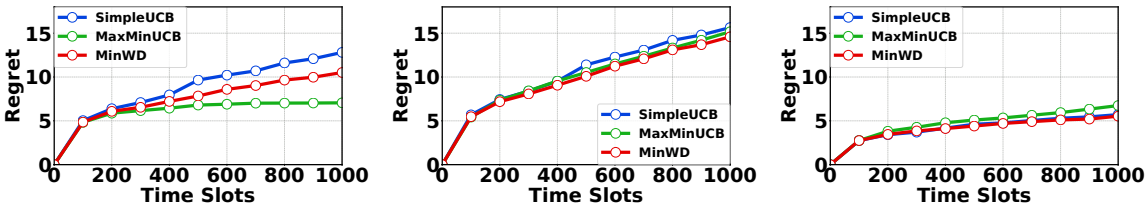
By Lemma 10 and Proposition 11, we bound the cumulative regret of MinWD.

Theorem 2.5.2. *If* MinWD *is used to select arms with imperfect context and as time goes on, and the conditions in Proposition 11 are satisfied, then for any true context $x_t \in \mathcal{B}_\Delta(\hat{x}_t)$ at round $t, t = 1, \cdots, T$, with a probability of $1-\delta, \delta \in (0,1)$, we have the following bound on the cumulative true regret:*

$$R_T \leq \sum_t^T MR_t + 2h_T T^{\frac{3}{4}} \sqrt{\left(4\tilde{d}\log\left(1+\frac{T}{\tilde{d}\lambda}\right)+\frac{1}{\lambda}\right)} + 4\sqrt{\frac{2}{\lambda}} L_\phi \kappa^{\frac{1}{d}} h_T T^{1-\frac{1}{2d}} + 2h_T\sqrt{2T\bar{d}\log(1+\frac{T}{\bar{d}\lambda})},$$

where MR_t is the optimal worst-case regret for round t in Eqn. (2.11), d is the dimension of $x_{a_t,t}$, $\tilde{d}$ is the effective dimension defined in the proof of Proposition 11, $\bar{d}$ is the rank of $\mathbf{K}_t$ and h_T is given in Lemma 1.

Remark 13. *Theorem 2.5.2 shows that by* MinWD, *$R_T - \sum_{t=1}^T MR_t$ is sub-linear w.r.t. T and thus Type-II robustness is effectively achieved according to Definition 6. This means the true regret bound approaches $\sum_t^T MR_t$, the optimal worst-case regret, asymptotically.*



(a) Robust regret. (b) Worst-case regret. (c) True regret.

Figure 2.2: Different cumulative regret objectives for different algorithms.

Next, in parallel with MaxMinUCB, we derive the bound of true reward for MinWD.

Corollary 14. *If* MinWD *is used to select arms with imperfect context and as time goes on, and the true sequence of context and arm obeys the conditions in Proposition 11, then for any true contexts* $x_t \in \mathcal{B}_\Delta(\hat{x}_t)$ *at round* $t, t = 1, \cdots, T$*, with a probability of* $1-\delta, \delta \in (0,1)$*, we have the following lower bound of the cumulative reward*

$$F_T \geq \sum_{t=1}^{T} [MF_t - MR_t] - 2h_T T^{\frac{3}{4}} \sqrt{\left(4\tilde{d}\log\left(1+\frac{T}{\tilde{d}\lambda}\right)+\frac{1}{\lambda}\right)}$$
$$- 4\sqrt{\frac{2}{\lambda}} L_\phi \kappa^{\frac{1}{d}} h_T T^{1-\frac{1}{2d}} - 2h_T\sqrt{2T\bar{d}\log(1+\frac{T}{\bar{d}\lambda})},$$

where MR_t *is the optimal worst-case regret for round* t *in Eqn.* (2.11)*,* d *is the dimension of* $x_{a_t,t}$*,* $\tilde{d}$ *is the effective dimension defined in the proof of Proposition 11,* $\bar{d}$ *is the rank of* $\mathbf{K}_t$*, and* h_T *is given in Lemma 1.*

Remark 15. *Corollary 14 shows that as* t *becomes sufficiently large, the difference between the optimal worst-case reward and the true reward of the selected arm is no larger than the optimal worst-case regret* MR_t*. With perfect context, we have* $MR_t = 0$*, and hence* MaxMinUCB *and* MinWD *both asymptotically maximize the reward, implying that these two types of robustness are the same under perfect context.*

Table 2.1: Summary of Analysis

Algorithms	**Regret**	**Reward**
MaxMinUCB	$\sum_{t=1}^{T}\overline{MR}_t$ + $O(\sqrt{T\log T})$	$\sum_{t=1}^{T}MF_t$ − $O(\sqrt{T\log T})$
MinWD	$\sum_{t=1}^{T}MR_t$ + $O(T^{\frac{3}{4}}\sqrt{\log T}$ + $T^{1-\frac{1}{2d}}+\sqrt{T\log T})$	$\sum_{t}^{T}[MF_t-MR_t]$− $O(T^{\frac{3}{4}}\sqrt{\log T}$ + $T^{1-\frac{1}{2d}}+\sqrt{T\log T})$

2.5.3 Summary of Main Results

We summarize our analysis of MaxMinUCB and MinWD in Table 2.1, while the algorithms details are available in Algorithm 1. In the table, d is the dimension of context-arm vector $[x, a]$, $\overline{MR}_t = \max_{x\in\mathcal{B}_\Delta(\hat{x}_t)} f\left(x, A^*\left(x\right)\right) - MF_t$, and MF_t and MR_t are defined in Eqn. (2.6) and (2.11), respectively. Type-I and type-II robustness objectives are achieved by MaxMinUCB and MinWD respectively.

2.6 Simulation

Edge computing is a promising technique to meet the demand of latency-sensitive applications [357]. Given multiple heterogeneous edge datacenters located in different locations, which one should be selected? Specifically, each edge datacenter is viewed as an arm, and the users' workload is context that can only be predicted prior to arm selection. Our goal is to learn datacenter selection to optimize the latency in a robust manner given imperfect workload information. We assume that the service rate of the edge datacenter a, $a \in \mathcal{A}$, is μ_a, the computation latency satisfies an M/M/1 queueing model and the average communication delay between this datacenter and users is p_a. Hence, the average total

latency cost can be expressed as $l(x, a) = p_a \cdot x + \frac{x}{\mu_a - x}$ which is commonly-considered in the literature [262, 416, 261]. The detailed settings are given in [419].

In Fig. 2.2, we compare different algorithms in terms of three cumulative regret objectives: robust regret in Eqn. (2.7), worst-case regret in Eqn. (2.8) and true regret in Eqn. (2.2). We consider the following algorithms: SimpleUCB with imperfect context, MaxMinUCB with imperfect context and MinWD with imperfect context. Given a sequence of true contexts, imperfect context sequence is generated by sampling i.i.d. uniform distribution over $\mathcal{B}_\Delta(x_t)$ at each round. In the simulations, Gaussian kernel with parameter 0.1 is used for reward (loss) estimation. λ in Eqn. (2.3) is set as 0.1. The exploration rate is set as $h_t = 0.04$.

As is shown in Fig. 2.2(a), MaxMinUCB has the best performance of robust regret among the three algorithms. This is because MaxMinUCB targets at type-I robustness objective which is equivalent to minimizing the robust regret. However, MaxMinUCB is not the best algorithm in terms of true regret as is shown in Fig. 2.2(c) since robust regret is not an upper or lower bound of true regret. Another robust algorithm MinWD is also better than SimpleUCB in terms of robust regret, and it has the best performance among the three algorithms in terms of the worst-case regret, as shown in Fig. 2.2(b). This is because the regret of MinWD approaches the optimal worst-case regret (Theorem 2.5.2). MinWD also has a good performance of true regret, which coincides with the fact that the worst-case regret is the upper bound of the true regret. By comparing the three algorithms in terms of the three regret objectives, we can clearly see that MaxMinUCB and MinWD achieve performance robustness in terms of the robust regret and worst-case regret, respectively.

2.7 Conclusion

In this chapter, considering a bandit setting with imperfect context, we propose: MaxMinUCB which maximizes the worst-case reward; and MinWD which minimizes the worst-case regret. Our analysis of MaxMinUCB and MinWD based on regret and reward bounds shows that as time goes on, MaxMinUCB and MinWD both perform as asymptotically well as their counterparts that have perfect knowledge of the reward function. Finally, we consider online edge datacenter selection and run synthetic simulations for evaluation.

Chapter 3

Learning-Augmented Online Decision Making With Known Dynamic Models

3.1 Introduction

We consider an *online* control problem with time-varying dynamics and cost functions that are revealed sequentially to the control agent. This model has practical applications to numerous problems such as control of cooling systems [282], online renewable aggregation [246], battery management for electrical vehicle (EV) charging [374], workload scheduling in datacenters [342], among others.

By exploiting the statistical information, machine learning (ML) has been widely applied for various online decision problems [17, 134, 243, 421]. In the context of online

control/optimization, ML has been leveraged to discover novel policies that often exceed the performance of manually-designed policies [246, 249, 251, 242]. While ML can progressively improve its policy for online control and achieve a low average cost in the long run, it suffers from the lack of performance guarantees in each step of an episode, especially when the amount of training data is insufficient and/or the ML model is not well designed. This potentially hinders the deployment of learning-based controller in real systems.

On the other hand, control priors, such as human-designed online algorithms or rule-based heuristic controllers, have provably worst-case performance guarantees or certified performances in real-world control systems. For example, competitive online control/optimization algorithms [233, 103, 356, 355, 170, 427, 172, 356, 171, 316] can achieve bounded worst-case cost ratio relative to the offline optimal control policy (a.k.a competitive ratio). In addition, heuristic controllers have been widely adopted in real systems (e.g., Sequence of Operations for industrial/commercial cooling [282, 104]), providing certified performance as the ground truth. That being said, these competitive algorithms or herustics typically cannot achieve as good average cost performance as ML-based algorithms due to the limited ability to utilize available statistical information.

To address the critical need for worst-case guarantees and good average performance in control systems, learning-augmented control/optimization algorithms [248, 246, 435, 340, 111, 244, 241], which combine the ML output with a robust control prior, have been developed in recent years. Among them, [248] provides a learning-augmented algorithm for Linear Quadratic Control (LQC) with performance bounds under both imperfect and perfect ML predictions. In addition, learning-augmented algorithms [340, 111, 244, 241] are also designed

for a finite-memory control problem, called Smoothed Online Convex Optimization (SOCO), where switching costs are included in the cost function and the state dynamic is a known and fixed function of the previous finite-step actions. Nonetheless, these studies typically focus on simple settings where the system dynamic follows a linear or fixed model completely known in advance, which may not capture real complex systems (e.g., industrial cooling [104]). Moreover, they only guarantee performance robustness for an entire episode, whereas the performance within an episode can still be much worse compared to running the control prior alone.

This paper considers a more general and challenging online control problem with non-linear and time-varying dynamic models that are sequentially revealed online. Importantly, to provide stronger performance robustness, we consider any-step competitiveness (Definition 16): given any $\lambda > 0$, the total cost cannot exceed that of the control prior scaled by $(1 + \lambda)$ at any step within any episode. The key challenges for utilizing ML-based controllers to improve the average performance while still being able to guarantee any-step competitiveness come from uncertainties of time-varying dynamics and cost functions that are revealed sequentially over the online control process. To address these challenges, we design a novel competitiveness-constrained online policy learning algorithm, called `LACC` (Learning-Augmented Online Competitive Control). Concretely, `LACC` leverages construction of novel reservation costs to provably guarantee any-step $(1 + \lambda)$-competitiveness and meanwhile trains the ML-based policy based on available data to improve the average performance. Our analysis quantifies highlights the impact of the any-step competitiveness constraint on the average cost performance of `LACC`, revealing the tradeoff between the

average performance with worst-case competitiveness governed by $\lambda > 0$. Finally, we also show the convergence of `LACC` as the number of episodes increases and empirically validate `LACC` using battery management in charging stations.

3.2 Related Work

Online Competitive Control/Optimization. Our work is relevant to the literature of online competitive control/optimization. In our problem setting, the target is to minimize the cumulative cost in the nonlinear dynamics, which is different from the traditional control literature that uses measures for stabilization purposes [321, 147, 148, 218]. Like the recent works on competitive control [170, 169, 427, 172, 356, 171, 316], our work considers guarantees on the worst-case competitiveness, but our main focus is different — we leverage ML to explore policies with low average cost while enforcing competitiveness guarantees for any step in any episode. This enables the use of the existing competitive control policies as priors. Achieving our objective requires novel design of safe action sets and new analysis techniques to find the trade-off between the average performance and worst-case competitiveness.

Conservative Learning. The literature on conservative learning uses a policy prior to guide the exploration process [423, 5, 422, 84, 97, 20, 130, 103]. They consider an *average* constraint or a total T-episode constraint (with respect to a policy prior) with a high probability. Additionally, any-step constraints have also been considered in other contexts [274, 314], but these studies only require any-step constraints in expectation instead of our any-step competitiveness for any instance. Thus, our any-step competitiveness

provides stronger robustness that is desired in real mission-critical control systems such as industrial cooling. Other studies consider orthogonal constraints such as safe states or per-step (non-commulative) constraints [84, 20, 97].

ML-Augmented Online Algorithms. Our study is most relevant to the emerging ML-augmented online algorithms [402, 70, 284, 106, 110]. The goal of ML-augmented online algorithms is to combine potentially untrusted ML predictions with robust policies (i.e., control priors). ML-augmented algorithms have been developed for online control/optimization by combining ML predictions and control priors through online switching [340, 31] or adaptively setting a confidence on the ML prediction [248, 246]. Compared to these studies, we make contributions by considering a more challenging setting, i.e., non-linear and time-varying dynamic models that are sequentially revealed online. Although some of the existing studies [248, 340, 111, 244, 241] provide provable competitive bounds, they cannot guarantee any-step competitiveness that is needed for real control [282]. Moreover, these studies typically assume a pre-trained ML model as a black box, whereas we study the policy learning process subject to the competitiveness constraint against a control prior.

3.3 Problem Formulation

3.3.1 Finite-Horizon Control Model

We consider a general finite-horizon control problem with time-varying nonlinear dynamics. More specifically, at time $h \in [H-1]$, the control agent observes the current state $x_h \in \mathcal{X} \in \mathbb{R}^n$, chooses an action $u_h \in \mathcal{U} \subseteq \mathbb{R}^d$ and incurs a non-negative system cost $c_h(x_h, u_h) : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}_+$. Given x_h and u_h, the system transitions to x_{h+1} at time $h+1$

following system dynamics as:

$$x_{h+1} = f_h(x_h, u_h) + w_h, \quad h = 0, \ldots, H-1, \tag{3.1}$$

where $f_h : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ denotes a time-varying function, and w_h is an additive perturbation noise. At the last time H, the terminal cost $c_H(x_H)$ only relies on the final state x_H.

Unlike the prior literature [316, 340, 248, 325, 249, 435] that focuses on simple linear or fixed models completely known a priori, we consider a more general setting and assume that the dynamics f_h and cost function c_h are sequentially revealed to the control agent before choosing u_h, while the additive noise w_h is not revealed or observed and can only be calculated as $w_h = x_{h+1} - f_h(x_h, u_h)$ after x_{h+1} is observed at time $h+1$. For convenience, we denote $y_h \coloneqq (f_h, c_h, w_h)$ and $y_{0:H} = (y_1, \cdots, y_H)$ as the information of an entire episode. Importantly, $y_{0:H} \in \mathcal{Y}$ is sampled from an unknown distribution $\mathbb{P}_{y_{0:H}}$. An online control policy denoted by π is a function which outputs the action $u_h \in \mathcal{U}$. With an initial state x_0, the cumulative cost up to round h is expressed as

$$J_h^\pi = \sum_{i=0}^{h} c_i(x_i, u_i) \text{ for } h \in [H-1], \text{ and } J_H^\pi(y_{0:H}) \coloneqq \sum_{h=0}^{H} c_h(x_h, u_h) + C_H(x_H), \tag{3.2}$$

We summarize the standard assumptions on the system dynamics and costs below. Our first assumption is the Lipschitz continuity assumption on the functions $(f_h : h = 0, \ldots, H-1)$ and it is common in finite-horizon control models [248, 249, 427].

Assumption 1 (Lipschitz continuity of dynamics)**.** *For each time h, the function f_h is Lipschitz continuous with respect to x_h and u_h with Lipschitz constants $\sigma_x \geq 0$ and $\sigma_u \geq 0$, respectively, i.e., for any (x, u) and (x', u'), f_h satisfies $\|f_h(x, u) - f_h(x', u)\| \leq \sigma_x \|x - x'\|$ and $\|f_h(x, u) - f_h(x, u')\| \leq \sigma_u \|u - u'\|$.*

The second assumption is the smoothness of the costs $(c_h : h = 0, \ldots, H)$, which is a common regularity condition on control system costs [249, 269, 355, 267].

Assumption 2 (Well-conditioned costs)**.** *For each time h, the cost function c_h is non-negative, convex, and β-smooth with respect to (x_h, u_h).*

Note that, if the Lipschitz continuous f_h is the linear dynamics $x_{h+1} = D_x x_h + D_u u_h + w_h$ where D_x and D_u are controllable matrices, and the smooth cost functions c_h is the quadratic function, i.e. $c_h(x_h, u_h) = x_h^\top Q x_h + u_h^\top R u_h$, $C_H(x_H) = x_H^\top Q x_H$ where Q, $R \succ 0$ are all positive definite, our control model reduces to the classic setting of *linear quadratic control* [248, 325, 249, 435].

3.3.2 Objective and Performance Metrics

We seek to design a control policy π to minimize the average cost $\mathbb{E}_{y_{0:H}}\left[J_H^\pi(y_{0:H})\right]$ while ensuring a competitiveness guarantee for any step in each instance/episode against a control prior $\pi^\dagger$. Let $J_h^\pi = \sum_{i=0}^h c_i(x_i, u_i)$ and $J_h^{\pi^\dagger} = \sum_{i=0}^h c_i(x_i^\dagger, u_i^\dagger)$ are respectively the cumulative costs of the policy π and control prior $\pi^\dagger$ defined in Eqn.(3.2). We first give the formal definition of competitiveness.

Definition 16 (Any-step competitiveness)**.** *A policy π is $(1+\lambda)$-competitive against a control prior $\pi^\dagger$ for $\lambda > 0$ if $J_h^\pi \le (1+\lambda) J_h^{\pi^\dagger}$ for all $h \in [H]$ is satisfied for any episode $y_{0:H} \in \mathcal{Y}$.*

Any-step competitiveness requires that, given any instance $y_{0:H} \in \mathcal{Y}$, the cost of a controller never exceeds $(1+\lambda)$ times the cost of a control prior $\pi^\dagger$ at any time h. Compared to the per-episode constraint considered in the literature [248], any-step competitiveness is

stricter and also more practical in practical control systems, because the control process might stop at any round $h \leq H$ while the agent does not know the stopping time in advance.

Our competitiveness constraint is also strongly motviated by real-world control systems. For example, in industrial cooling, a control agent decides to turn on/off various pieces of equipment to optimize the energy efficiency [282, 104, 409]. Control priors, such as Sequence of Operations (SOO), have been programmed into systems with a long history and provide certified performance. To deploy a new learning-based controller, strong performance robustness compared to the control prior is required. In another example of workload scheduling in datacenters, the learning-based control agent needs to decide the number of active servers for carbon efficiency while ensuring that its performance always stays competitive against a control prior. For this application, there exist both empirically-strong [326] and theoretically-competitive [171, 356] online control policies that can readily serve as control priors. Thus, by using a control policy that is competitive against the offline-optimal cost, our any-step competitiveness also immediately translates into guaranteed competitiveness against the offline-optimal algorithm with a scaling of $(1 + \lambda)$.

Under any-step competitiveness constraints, a learning-based controller is utilized to optimize the average performance over the distribution of $y_{0:H} \in \mathcal{Y}$. Thus, our objective is to find an online policy that optimizes the following:

$$\min_{\pi} \mathbb{E}_{y_{0:H}} \left[J_H^{\pi}(y_{0:H}) \right], \quad s.t., \ J_h^{\pi} \leq (1+\lambda) J_h^{\pi^\dagger}, \ \ \forall h \in [H], \forall y_{0:H} \in \mathcal{Y}. \tag{3.3}$$

For convenience, we define the collection of all $(1 + \lambda)$-competitive control policies as $\Pi_\lambda = \{\pi \mid J_h^{\pi} \leq (1+\lambda) J_h^{\pi^\dagger}, \forall h \in [H], \ \forall y_{0:H} \in \mathcal{Y}\}$. If λ is larger, then the size of Π_λ is also larger and we have more flexibility to optimize the average cost. With episodic data

$\{y_{0:H} \in \mathcal{Y}\}$ collected online, we use ML to learn a control policy $\pi_\lambda \in \Pi_\lambda$ while satisfying the any-step competitiveness. In the online setting, we update our policy π_λ^t in each episode $t = 1, \ldots, T$, with each episode corresponding to an instance of our finite-horizon control problem in Section 3.3.1.

We define the unconstrained-optimal policy as $\pi^* = \arg\max_{\pi \in \Pi_\infty} \mathbb{E}_{y_{0:H}} [J_T^\pi(y_{0:H})]$, which minimizes the expected cost without considering any-step competitiveness. Next, we define the expected regret defined as follows.

Definition 17 (Expected regret)**.** *Given the any-step competitiveness constraint in* (3.3) *for each episode, the expected regret of an online competitive control policy π_λ with respect to the unconstrained-optimal policy $\pi^* = \arg\max_{\pi \in \Pi_\infty} \mathbb{E}_{y_{0:H}} [J_T^\pi(y_{0:H})]$ is defined as*

$$R(\pi_\lambda, \pi^*) = \mathbb{E}_{y_{0:H}} \left[J_T^{\pi_\lambda}(y_{0:H}) - J_T^{\pi^*}(y_{0:H}) \right]. \tag{3.4}$$

The expected regret $R(\pi_\lambda, \pi^*)$ quantifies the tradeoff between the average cost performance and the competitiveness requirement with respect to a control prior. The tradeoff between average and worst-case performance is inevitable and well-known in the online optimization literature [110, 284], but algorithm designs are needed to achieve a lower $R(\pi_\lambda, \pi^*)$.

3.4 Algorithm Design

3.4.1 Safe Action Set for Per-Episode Competitiveness

Our goal is to find a policy satisfying any-time competitiveness with a low expected cost. In online control, however, it is challenging to guarantee the any-step competitiveness

constraint in (3.3) for any episode. This is because the competitiveness constraint relies on the cost J_h^{π} of the control policy and the cost $J_h^{\pi^\dagger}$ of the control prior $\pi^\dagger$ which can both be evaluated only after the information $(f_{0:h}, c_{0:h}, x_{0:h})$ is observed at step h. If the competitiveness constraint is violated at some step h, the selected online actions before h cannot be revised. As a result, we must construct a safe action set $\mathcal{U}_{\lambda,h}$ for each round $h \in [0, H-1]$ within an episode: if actions are selected from the safe action sets, any-step competitiveness is always satisfied even in the worst case.

A naive design is to directly ensure the competitiveness constraint (3.3) for each step h, i.e.

$$\overline{\mathcal{U}}_{\lambda,h} = \left\{ u_h \mid \sum_{i=0}^{h} c_i(x_i, u_i) \leq (1+\lambda) \sum_{i=0}^{h} c_i(x_i^\dagger, u_i^\dagger) \right\}. \tag{3.5}$$

In online control, however, the naive safe action set $\overline{\mathcal{U}}_{\lambda,h}$ can be empty at some step h, resulting in no feasible actions to meet any-step competitiveness, which we explain as follows. Suppose that $\sum_{i=0}^{h} c_i(x_i, u_i) = (1+\lambda) \sum_{i=0}^{h} c_i(x_i^\dagger, u_i^\dagger)$ is satisfied at time h. If $x_{h+1} = x_{h+1}^\dagger$ holds at round $h+1$, the agent can always choose $u_{h+1} = u_{h+1}^\dagger$ to satisfy (3.5) at round $h+1$. However, when $x_{h+1} \neq x_{h+1}^\dagger$, it is possible that the control prior has a low cost for its state $x_{h+1}^\dagger$ at time $h+1$ such that for any action $u \in \mathcal{U}$ the true cost $c_{h+1}(x_{h+1}, u)$ is lager than the prior cost $(1+\lambda)c_{h+1}(x_{h+1}^\dagger, u_{h+1}^\dagger)$. In such a case, the naive safe action set $\overline{\mathcal{U}}_{\lambda,h}$ is empty and the control agent cannot maintain the inequality in (3.5), thus potentially violating the subsequent any-step competitiveness constraints.

Consequently, if an action in the safe action set $\mathcal{U}_{\lambda,h}$ leads to a state x_{h+1} different from the state $x_{h+1}^\dagger$ of the control prior, the resulting cumulative cost should satisfy $\sum_{i=0}^{h} c_i(x_i, u_i) + \phi_h \leq (1+\lambda) \sum_{i=0}^{h} c_i(x_i^\dagger, u_i^\dagger)$ with an added reservation cost $\phi_h > 0$ for

hedging. For the design of the reservation cost, we must consider all the possible future control environments $y_{h+1:H}$ before the decision at each time $h =\in [H-1]$ and hedge against the worst case for competitiveness. To this end, we design a reservation cost in the next proposition.

Proposition 18. *With Assumptions 1 and 2, if the action u_h at round h is selected from the safe set $\mathcal{U}_{\lambda,h}, \lambda > 0$ defined as*

$$\mathcal{U}_{\lambda,h} \coloneqq \left\{ u \in \mathcal{U} \mid J_{h-1} + c_h(x_h, u) + \phi_h(u) \leq (1+\lambda) J_h^{\pi^\dagger} \right\} \tag{3.6}$$

where $J_{h-1} = \sum_{i=0}^{h-1} c_i(x_i, u_i)$ and $J_h^{\pi^\dagger} = \sum_{i=0}^{h} c_i(x_i^\dagger, u_i^\dagger)$ are the true costs and the cost of control prior, respectively, with the reservation cost function defined as

$$\phi_h(u) = q_h \| f_h(x_h, u) - f_h(x_h^\dagger, u_h^\dagger) \|^2 \tag{3.7}$$

where $q_h = C_1(1+\frac{1}{\lambda})\frac{\beta}{2}\sum_{h'=0}^{H-h-1}(C_2\sigma_x^2)^{h'}$ for constants $C_1 \geq 1$ and $C_2 \geq 1$, then the any-step $(1+\lambda)$-competitiveness (3.3) is guaranteed.

The key insight for the reservation cost $\phi_h(u)$ in (3.7) is to account for the worst-case future cost gap between the control policy π_λ and the control prior $\pi^\dagger$ resulting from the current state/action difference at each round $h = 0, \ldots, H-1$. By adding the reservation cost at each time h, there always exists a non-empty safe action set $\mathcal{U}_{\lambda,h}$ in the subsequent steps, ensuring any-step competitiveness.

3.4.2 Learning-Augmented Competitive Control

By Proposition 18, any-step competitiveness in (3.3) is strictly satisfied if the action at each time h is chosen from the safe action set $\mathcal{U}_{\lambda,h}$. Therefore, given an ML policy $\tilde{\pi}$

Algorithm 2 Online Competitive Control with `LACC`

Require: ML Policy $\tilde{\pi}$ and policy prior $\pi^\dagger$

for time horizon $h = 0, \cdots, H-1$ **do**

  Observe state x_h and information $\{c_h, f_h\}$, and calculate last-step noise w_{h-1}.

  Update the policy prior's state $x_h^\dagger = f_{h-1}(x_{h-1}^\dagger, u_{h-1}^\dagger) + w_{h-1}$

  Obtain an action $u_h^\dagger$ by the prior $\pi^\dagger$, and update prior cost $J_h^{\pi^\dagger} = J_{h-1}^{\pi^\dagger} + c_h(x_h^\dagger, u_h^\dagger)$

  Obtain the ML action $\tilde{u}_h$ via the ML model $\tilde{\pi}$

  if the ML action $\tilde{u}_h \in \mathcal{U}_{\lambda,h}$ **then** take $u_h = \tilde{u}_h$

  else take $u_h = m(\tilde{u}_h)$ **end if** *// Map to a safe action set $\mathcal{U}_{\lambda,h}$ (3.6) using m*

  Update true cost $J_h = J_{h-1} + c_h(x_h, u_h)$

end for

and a control prior $\pi^\dagger$, we design a learning-augmented competitive controller as shown in Algorithm 2.

At each round h within an episode, the control agent first evaluates the cost of the control prior. To achieve this, after observing the true state x_h and the dynamics f_{h-1}, we first inversely calculate the noise at the last round as $w_{h-1} = x_h - f_{h-1}(x_{h-1}, u_{h-1})$, which implies a "virtual state" corresponding to the control prior for the same online information $y_{0:h-1}$, denoted by $x_h^\dagger = f_{h-1}(x_{h-1}^\dagger, u_{h-1}^\dagger) + w_{h-1}$. Next, we query the control prior $\pi^\dagger$ with a state $x_h^\dagger$ and obtain an action $u_h^\dagger$, which can be used to update the cumulative cost at round h, denoted by $J_h^\dagger = J_{h-1}^\dagger + c_h(x_h^\dagger, u_h^\dagger)$. By doing so, a safe action set $\mathcal{U}_{\lambda,h}$ can be constructed as in Proposition 18.

It remains to select an action from the safe action set. If the ML action $\tilde{u}_h$ is in the safe action set $\mathcal{U}_{\lambda,h}$, then we simply select $u_h = \tilde{u}_h$. Otherwise, we can use a projection

Algorithm 3 Online Policy Training

1: Initialize the ML policy $\tilde{\pi}^{(0)}$

2: **for** episode $t = 1, \ldots, T$ **do**

3: Implement Algorithm 2 with $\tilde{\pi}^{(t-1)}$ and record $y_{0:H}^{(t)}$.

4: Append $y_{0:H}^{(t)}$ to the replay buffer $\mathcal{D}_t$

5: Update the ML policy $\tilde{\pi}^{(t)}$ by (3.9).

6: **end for**

function $m : \mathbb{R}^d \to \mathcal{U}_{\lambda,h}$ that projects the ML action $\tilde{u}_h$ into an action in the safe action set. Thus, the selected action can be represented as

$$u_h = m(\tilde{u}_h, \mathcal{U}_{\lambda,h}) = \arg \min_{u \in \mathcal{U}_{\lambda,h}} \|\tilde{u}_h - u\|. \tag{3.8}$$

When the safe action set is a convex set (e.g. the dynamic functions $\{f_h : h \in [H]\}$ are linear [170, 427, 169, 435]), the projection can be efficiently solved by convex optimization. Otherwise, projection can be performed by alternative low-complexity algorithms [256, 87].

Algorithm 2 strictly guarantees any-time $(1+\lambda)$-competitiveness against the control prior $\pi^\dagger$ even when the ML policy $\tilde{\pi}$ has an arbitrarily bad performance (e.g., when only limited data is collected), thus ensuring a strong any-step competitiveness than the existing constrained learning-based control that focuses on *average* constraints or constraints over a horizon.

3.4.3 Training the ML Policy

With the any-step competitiveness constraint, training the ML policy becomes significantly different compared with an unconstrained (or only average-constrained) setting,

and is shown in Algorithm 3. Specifically, the ML policy $\tilde{\pi}$ interacts with the control environment through a mapping onto the safe action set $\mathcal{U}_{\lambda,h}$. Denoting the mapping in (3.8) by $u_h = m(\tilde{u}_h, \mathcal{U}_{\lambda,h})$, given the ML action $\tilde{u}_h$, the dynamics is $x_{h+1} = f_h(x_h, m(\tilde{u}_h, \mathcal{U}_{\lambda,h})) + w_h$ and the cost becomes $c_h = (x_h, m(\tilde{u}_h, \mathcal{U}_{\lambda,h}))$. We update the ML policy $\tilde{\pi}$ by Algorithm 3 in an episodic setting, i.e., whenever an episode is solved by Algorithm 2 after the episode t, we append the information $y_{0:H}$ of the full episode to the replay buffer $\mathcal{D}_t$. Then, the ML policy denoted by $\tilde{\pi}_\lambda^{(t)}$ is updated by minimizing the empirical cost on the replay buffer $\mathcal{D}_t$ as

$$\tilde{\pi}_\lambda^{(t)} = \arg\min_{\tilde{\pi}\in\Pi} \sum_{y_{0:H}\in\mathcal{D}_t} \sum_{h=0}^{H} c_h\big(x_h, m\left(\tilde{\pi}(s_h), \mathcal{U}_{\lambda,h}\right)\big), \tag{3.9}$$

where the ML input s_h consists of the current system state x_h, available environment information $\{f_i, c_i, i \in [0,h]\}$ and $\{w_i, i \in [0,h-1]\}$, and the actions $\{u_i^\dagger, i \in [0,h]\}$ of the control prior $\pi^\dagger$. The updated ML policy $\tilde{\pi}_\lambda^{(t+1)}$ is then used for the $(t+1)-$th episode.

To train an ML policy based on (3.9), we can directly parameterize the ML policy using an ML model with trainable weights (e.g. neural networks) and use gradient descent to optimize the weights in the minimization (3.9). To be more precise, we perform the back-propagation through the online control process where all the operations are differentiable (e.g. the projection operator can be implicitly differentiated by the KKT conditions, as shown in [7]). The ML policy can also be obtained by value-based methods wherein, e.g., a $Q_h(s,u)$ function that evaluates the expected long-term cost given the input s and action u is learned and the ML action is chosen as $\tilde{u}_h = \arg\max_u Q_h(s,u)$.

3.5 Performance Analysis

With any-step competitiveness of `LACC` guaranteed by design, we now bound its expected regret.

3.5.1 Expected Regret

To highlight the impact of λ on the tradeoff between the average performance and any-step competitiveness, we first consider a case where the ML model is optimally trained (with a sufficiently large mumber of training samples) to minimize the expected cost of the online control process, i.e., $\tilde{\pi}^*_\lambda = \arg\min_{\tilde{\pi}\in\Pi} \mathbb{E}_{y_{0:H}} \left[\sum_{h=0}^{H} c_h(x_h, m(\tilde{\pi}(s_h), \mathcal{U}_{\lambda,h}))\right]$. The convergence of online training in Algorithm 3 to $\tilde{\pi}^*_\lambda$ as the number of episodes increases will be studied in Section 3.5.2.

Given the ML model $\tilde{\pi}^*_\lambda(s_h)$, for notation convenience, we denote the corresponding optimal competitive policy as $\pi^*_\lambda : \pi^*_\lambda(s_h) = m(\tilde{\pi}^*_\lambda(s_h), \mathcal{U}_{\lambda,h})$. Due to the irreducible gap introduced by the any-step competitiveness constraint, the expected cost of π^*_λ is no less than that of the optimal-unconstrained policy $\pi^* = \arg\min_{\pi\in\Pi} \mathbb{E}_{y_{0:H}} \left[\sum_{h=0}^{H} c_h(x_h, \pi(s_h))\right]$. Thus, we bound the expected regret between our competitive policy π^*_λ and the optimal policy π^*, showing the tradeoff between the worst-case and the average performances. For the sake of analysis, we first make the following Lipschitz assumption which is not overly restrictive. The policy space Π can be a space of neural networks which satisfy Lipschitz continuity by using spectrum normalization [53, 296, 221].

Assumption 3. *The optimal-unconstrained policy* $\pi^* = \arg\min_{\pi\in\Pi_\infty} \mathbb{E}_{y_{0:H}} [J^\pi_T(y_{0:H})]$ *is* L_π*-Lipschitz continuous in terms of the input* s_h *for any* $h \in [H-1]$.

Theorem 3.5.1. *With Assumption 3, by choosing* $\phi_h(u) = q_h \|f_h(x_h, u_h) - f_t(x_h^\dagger, u_h^\dagger)\|^2$ *with* $q_h = \left(\frac{\sqrt{1+\lambda}+1}{\lambda} + 1\right) \frac{\beta}{2} \sum_{i=0}^{H-h-1} \left(2\sigma_x^2\right)^i$*, the expected regret is bounded by*

$$R(\pi_\lambda^*, \pi^*) \leq \min\left\{ B\mathbb{E}\left[\sum_{h=0}^{H-1} \left[\eta_h - (\sqrt{1+\lambda}-1)^2 G c_h^\dagger\right]^+\right],\ R(\pi^\dagger, \pi^*)\right\} \tag{3.10}$$

where $\eta_h = \|u_h^* - u_h^\dagger\|$ *is the action discrepancy between the optimal unconstrained policy and the control prior,* $G = 2\left(\beta A\left(1 + 2\sigma_u^2(1-(2\sigma_x^2)^{H-h})/(1-2\sigma_x^2)\right)\right)^{-1}$ *and* $B = \beta A\left(1 + (1+2L_\pi)\sigma_u \sum_{i=0}^{H-1}(\sigma_x + 2\sigma_u L_\pi)^{h-i-1}\right)$ *are constants that only depend on the control system, in which* β *is the smoothness parameter of the cost function* c_h*, the size of the state-action set is* $A = \max_{(x,u),(x',u')\in\mathcal{X}\times\mathcal{U}} \|(x,u) - (x',u')\|$*,* L_π *is the Lipschitz constant of the unconstrained-optimal policy* π^**,* σ_x *and* σ_u *are the Lipschitz constants of the dynamics model* f_h *(Assumption 1).*

The results in Theorem 3.5.1 can be interpreted as follows. First, the any-step competitiveness constraint naturally creates a gap of expected cost between the competitive optimal policy π_λ^* and the unconstrained optimal policy π^*. More specifically, given a control prior $\pi^\dagger$, when $\lambda > 0$ becomes smaller, the competitiveness constraint is more stringent, which thus makes the actions of control policy π_λ^* potentially deviate more from those of the unconstrained-optimal policy π^* and increases the first term inside the minimum operator (3.10). On the contrary, when $\lambda > 0$ becomes larger, the competitiveness constraint is more relaxed, reducing the expected regret of the control policy π_λ^* to the optimal unconstrained policy π^*. In particular, if $c_h^\dagger > 0$ for all $h \in [H]$ and λ is sufficiently large, the terms $\left[\eta_h - (\sqrt{1+\lambda}-1)^2 G c_h^\dagger\right]^+$ can reduce to zero, voiding the competitiveness constraint and resulting in a zero expected regret. In any case, the expected regret of π_λ^* is always bounded by the control prior's regret $R(\pi^\dagger, \pi^*)$, because the actions of both π_λ^* and $\pi^\dagger$ are in the safe

action sets $\mathcal{U}_{\lambda,h}$ and π^*_λ is the optimal policy with such constraints. Note that the tradeoff in Theorem 3.5.1 comes from conservativeness to address the worst-case problem instance and is unavoidable for online control and optimization in general [110].

We now discuss the intrinsic impact of the control system parameters on the expected regret bound. The bound increases with the episode length H (inside the constant B) because the competitiveness-induced state perturbation compared to the optimal unconstrained policy π^* accumulates as we use a constrained online policy π^*_λ different from π^*. Another parameter that affects the expected cost ratio is η_h, i.e. the discrepancy between the action of the optimal policy π^* and that of the control prior $\pi^\dagger$. The intuition is that given a larger η, the control prior $\pi^\dagger$ is more different than the unconstrained-optimal policy π^*, naturally making it more difficult for our control policy to approach π^* while meeting any-step competitiveness.

3.5.2 Convergence

In this section, we denote the competitive policy at the end of episode t as $\pi^{(t)}_\lambda(s_h) = m(\tilde{\pi}^{(t)}_\lambda(s_h), \mathcal{U}_{\lambda,t})$ with $\tilde{\pi}^{(t)}_\lambda$ obtained by Algorithm 3 and bound its expected regret.

Theorem 3.5.2. *If ML policy is trained by Eqn.* (3.9)*, with probability at least* $1-\delta, \delta \in (0,1)$*, the expected regret of our competitive policy is bounded by*

$$R\big(\pi^{(T)}_\lambda, \pi^*\big) \leq B\mathbb{E}\left[\sum_{h=0}^{H-1}\left[\eta_h - (\sqrt{1+\lambda}-1)^2 G c^\dagger_h\right]^+\right] + \mathcal{O}\left(\sqrt{\frac{1}{T}\ln\frac{N(\epsilon,\Pi_\lambda,\hat{L}^T_1)}{\delta}}\right),$$

where the system-related parameters B, G *and* η *have the same definition as in Theorem 3.5.1,* $N(\epsilon,\Pi_\lambda,\hat{L}^T_1)$ *is the* ϵ*-covering number of the competitive policy space* Π_λ *with* L_1*-norm as the distance measure (the distance of two policies* π *and* π' *is* $\|\pi-\pi'\|_{\hat{L}^T_1} =$

$\frac{1}{T}\sum_{t=1}^{T}\sum_{h=1}^{H}\|\pi(s_h^{(t)}) - \pi'(s_h^{(t)})\|_1$*), and* $\mathcal{O}$ *indicates the scaling with cost upper bound* $\bar{c}$ *and episode length* H*.*

Theorem 3.5.2 shows that our policy with the online-trained ML model converges with a rate of $\sqrt{1/T}$. In particular, the convergence rate is affected by λ through the covering number $N(\epsilon, \Pi_\lambda, \hat{L}_2^T)$ which indicates the richness of the competitive policy class Π_λ. The covering number of the competitive policy class Π_λ is no larger than the unconstrained policy set Π_∞. This is because with the same ML model, the competitiveness constraint reduces the set of feasible actions — with a smaller $\lambda > 0$, the competitiveness policy space becomes smaller, making it easier for the convergence of `LACC`.

3.6 Simulation Study

In this section, we evaluate the empirical performance of our competitiveness-constrained policy by performing a simulation study for battery management in electric vehicle (EV) charging stations [382].

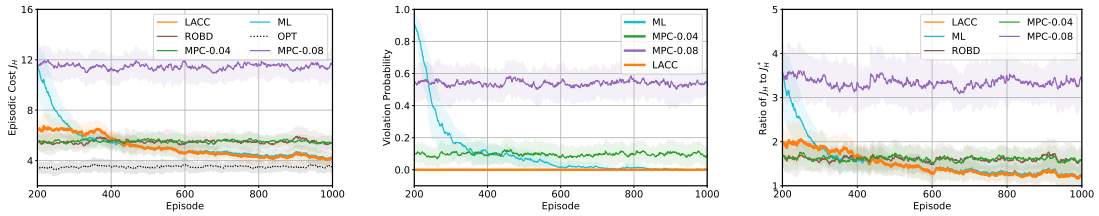
Setup. We consider the EV charging problem described as follows. At each time h, suppose that $x_h \in \mathbb{R}_+^n$ represents the State of Charge (SoC) and $u_h \in \mathbb{R}^d$ represents the decision for battery charging/discharging. The battery dynamics are $x_{h+1} = A_x x_h + A_u u_h - w_h$, where the matrices A_x and A_u are positive definite and capture the effects of self leakage and charging efficiency, respectively; w_h is the additive demand, and u_h represents the charging/discharging rate (kW). The goal is to decide the online charging/discharging rates to minimize the total charging cost while maintaining the battery SoC around a nominal value $\bar{x}$ [248, 316, 247, 382]. Formally, the control objective is $\min_{u_h : h\in[H-1]} \sum_{h=0}^{H-1} \|x_h - \bar{x}\|^2 + b\|u_h\|^2$.

In our case study, we use a recently open-sourced dataset for an EV charging station at Caltech [235]. We use data containing records from April to August of 2018. Each charging record contains the start time and end time of the EV charging session, and the total energy demand during this session. Without details of the batteries, we consolidate the energy demand within each hour as w_h. We consider each problem episode as one day ($H = 24$ hours) and generate 2856 problem episodes with a sliding window moving one hour ahead each time. We set $A_x = A_u = 1$ and $b = 5$ for the control model. For fair comparison, we use the same neural network architecture for different policies with the same initialized weights. Specifically, for learning, we use a neural network with 2 hidden layers, each with 8 neurons, and train the model using PyTorch. After each episode, we sample 14 available history instances from the dataset and train the model in an online manner. We use the reservation function in Theorem 3.5.1. We use the first 200 episodes as a warm-up stage for pre-training the baselines.

Baselines. For empirical comparison, we consider the the baseline algorithms as follows. Offline Optimal Policy (`OPT`) is the optimal offline policy that knows all the information in advance for each episode. Model Predictive Control (`MPC`)[158] solves the control problem by leveraging predictions of the future information. Linear Quadratic Regulator without Predictions (`LQR`) is a classic competitive controller that performs online control without predictions [426]. The control problem for EV charging can be formulated in the form of smoothed online convex optimization, for which Regularized Online Balanced Descent (`ROBD`) is the order-optimal online algorithm with the best-known competitive ratio. We also compare with a standard ML for control (`ML`) without competitiveness constraints.

Results. In Fig. 3.1, we show the simulation results for the setting with the competitiveness parameter $\lambda = 0.5$. The shadow region around the lines indicates the variance for each performance metric for different runs. Fig. 3.1(a) shows the empirical average cost. As `LQR` essentially assumes $w_h = 0$ which is not satisfied in our problem, its episodic cost is too high (greater than 200) to be included in the figure and hence it is omitted. The costs for our competitiveness-constrained policy `LACC` described in Algorithm 3 are highlighted in a thickened blue curve. We can find that the costs of `ML` and `LACC` both decrease when the number of episode increases. Among the early episodes (e.g. 200-300), both `ML` and `LACC` have higher costs than the policy prior `ROBD`, because `ML` and `LACC` need more data samples to learn a good policy. Nevertheless, `LACC` significantly outperforms `ML` at these early episodes due to its guarantee of $(1 + \lambda)$-competitiveness against the prior policy `ROBD` (Proposition 18). Moreover, as the number of episodes increases, both `ML` and `LACC` can achieve a low cost close to that of `OPT`, which shows the empirically good performance of `LACC` while being able to formally guarantee $(1 + \lambda)$-competitiveness against the control prior.

Fig. 3.1(b) shows the violation percentage that an algorithm exceeds the cost of $(1 + \lambda)J_H^{\texttt{ROBD}}$ where `ROBD` is used as the control policy prior due to its order-optimal competitive ratio. Note that `LACC` is guaranteed to have a zero violation rate for all episodes. We see that `ML` has a high violation probability at the beginning. In the training process, when the number of episodes increases, the violation rate of `ML` decreases, but the average violation probability over 800 episodes of `ML` is still as high as 0.212, even though we exclude the first 200 episodes during which `ML` is warmed up. `MPC-0.04` and `MPC-0.08` have average



(a) Average cost per episode J_H (b) Competitiveness violation (c) Episodic ratio of J_H to $J_H^{\texttt{OPT}}$

Figure 3.1: Comparisons of `LACC` with baselines.

violation probabilities of 0.096 and 0.540, respectively. The results shows the necessity to guarantee the competitiveness for each episode, especially when `ML` or ML predictions (for `MPC`) cannot be fully trusted yet.

Fig. 3.1(c) shows the per-episode cost ratios of different policies to `OPT`, complementing the results in Fig. 3.1(a).

3.7 Concluding Remarks

This chapter considers an online control problem with time-varying dynamics and cost functions. We focus on a novel setting where the goal is to minimize the average cost subject to the any-step competitiveness against a given control prior. We design a competitiveness-constrained algorithm, `LACC`, that uses reservation functions to ensure competitiveness. Our analysis formally highlights the impact of the competitiveness factor $\lambda > 0$ that trades off the average cost performance with worst-case competitiveness. We also show the convergence of `LACC` as the number of episodes increases and empirically validate `LACC` using simulations.

Chapter 4

Learning-Augmented Online Decision Making With Unknown Dynamic Models

4.1 Introduction

In mission-critical online decision-making problems such as cloud workload scheduling [326, 122], cooling control in datacenters [409, 104, 282], battery management for Electrical Vehicle (EV) charging [374, 373], and voltage control in smart grids [358, 443], there is always a need to improve the reward performance while meeting the requirements for some important cost metrics. In these mission-critical systems, there always exist some policy priors that meet the critical cost requirements, but they may not perform well in terms of the rewards. In the application of cooling control, for example, some rule-based

heuristics [120, 282] have been programmed into the real system for a long time and have verified performance in maintaining a safe temperature range, but they may not achieve a high energy efficiency. In this paper, we design a Reinforcement Learning (RL) algorithm with the goal of optimizing the reward performance under the guarantee of cost constraints against a policy prior for any round in any episode.

Constrained RL algorithms have been designed to solve various Constrained MDP (CMDP) with reward objective and cost constraints. Among them, some are designed to guarantee the expected cost constraints [139, 392], some can guarantee the cost constraints with a high probability [97], and others guarantee a bounded violation of the cost constraints [168, 130, 131, 5]. In addition, conservative RL algorithms [157, 423, 217, 422, 411] also compare the cost performance with a policy prior, but they only guarantee the cost constraints against a policy prior in expectation. In real mission-critical systems, however, the cost constraints are often required to be satisfied at each round in any instance even in the worst case, which hinders the deployment of these constrained/conservative RL policies. Recently, learning-augmented online control algorithms [248, 246, 435, 340, 111, 244] have been developed to exploit machine learning predictions with the worst-case control performance guarantee. Nonetheless, the learning-augmented control algorithms require the full knowledge of the dynamic models, which limits their application in many systems with unknown random dynamic models.

To fill in this technical blank, we model the mission-critical decision-making problem as a new Markov Decision Process (MDP) which is called the Anytime-Constrained MDP (A-CMDP). In A-CMDP, the environment feeds back a reward and a cost corresponding to

the selected action at each round. The next state is updated based on a random dynamic model which is a function of the current action and state and is not known to the agent. The distribution of the dynamic model is also unknown to the agent and needs to be learned. Different from CMDP, at each round h in any episode, the policy of A-CMDP must guarantee that the cumulative cost J_h is upper bounded by a scaled cumulative cost of the policy prior $\pi^\dagger$ plus a relaxation, i.e. $J_h \leq (1+\lambda)J_h^\dagger + hb$ with $\lambda, b > 0$, which is called an *anytime* constraint. Under these anytime cost constraints for all rounds, the RL agent explores the policy to optimize the expected reward.

The anytime constraint guarantee is much more strict than the requirements of constraints in typical CMDPs, which presents new challenges for RL algorithm design. First of all, the anytime constraints are required to be satisfied for any episode, even for the early episodes when the collected sequence samples are not enough. Also, to guarantee the constraints for each round, we need to design a safe action set for each round to ensure that feasible actions exist to meet the constraints in subsequent rounds. Last but not least, without knowing the full transition model, the agent has no access to the action sets defined by the anytime constraints, which makes it more difficult to guarantee the anytime constraints than the setting with known transition model [248].

Contributions. In this paper, we design algorithms to solve the novel problem of A-CMDP. The contributions are summarized as follows. First, we propose an Anytime-Constrained Decision-making (`ACD`) algorithm to provably guarantee the anytime constraints given any ML policy by projection to safe action sets. The safe action sets are updated at each round according to a designed rule to gain as much flexibility as possible to optimize

the reward. Then, we develop a new model-based RL algorithm (`ACRL`) to learn the optimal ML policy to be used in `ACD`. The proposed model-based RL can effectively utilize the new dynamic defined by `ACD` to reduce the learning complexity. Last but not least, we give rigorous analysis on the reward regret of `ACRL` comparing with the optimal-unconstrained policy. The analysis shows that the learned policy performs as well as the optimal `ACD` policy and these exist a fundamentally-irreducible performance gap between `ACD` policy and the optimal-unconstrained policy which is resulted from the anytime constraint guarantee.

4.2 Related Work

Conservative/Constrained RL. Compared with the existing literature on conservative/constrained RL [423, 5, 422, 84, 20, 130, 168, 139, 131, 392], our study has important differences. Concretely, the existing constrained RL works consider an *average* constraint with or without constraint violation. In addition, existing conservative RL works [423] consider an *average* constraint comparing with a policy prior. However, the constraints can be violated especially at early exploration episodes. In sharp contrast, our anytime constraint ensures a strict constraint for any round in each episode. In fact, with the same policy prior, our anytime-constrained policy can also meet the average constraint without violation in conservative/constrained RL [423, 137].

Our study is also relevant to safe RL. Some studies on safe RL [84, 20, 97] focus on constraining that the system state or action at each time h cannot fall into certain pre-determined restricted regions (often with a high probability), which is orthogonal to our anytime constraint requirement that constrains the cumulative cost at each round of

an episode. Our study is related to RL with safety constraints [84, 250], but is highlighted by the strict constraint guarantee for each round in each episode. In a study on safe RL [84], the number of safety violation events is constrained almost surely by a budget given in advance, but the safety violation value can still be unbounded. By contrast, our work strictly guarantee the anytime constraints by designing the safety action sets. In a recent study [20], the safety requirement is formulated as the single-round cost constraints. Differently, we consider a cumulative cost constraints which has direct motivation from mission-critical applications.

Learning-Augmented Online Decision-Making. Learning-based policies can usually achieve good average performance but suffer from unbounded worst-case performance. To meet the requirements for worst-case performance for learning-based policy, learning-augmented algorithms are developed for online control/optimization problems [248, 340, 111, 244, 241]. To guarantee the performance for each problem instance, learning-augmented algorithm can perform an online switch between ML policy and prior [340], combine the ML policy and prior with an adaptive parameter [248], or project the ML actions into safe action sets relying on the prior actions [244]. Comparing with learning-augmented algorithms, we consider more general online settings with unknown dynamic model and the safe action set design does not rely on any model information. Also, our problem can guarantee the cost performance for each round in any episode comparing with a policy prior, which has not been studied by existing learning-augmented algorithms.

4.3 Problem Formulation

4.3.1 Anytime-Constrained MDP

In this section, we introduce the setting of a novel MDP problem called Anytime-Constrained Markov Decision Process (A-CMDP), denoted as $\mathcal{M}(\mathcal{X}, \mathcal{A}, \mathcal{F}, g, H, r, c, \pi, \pi^\dagger)$. In A-CMDP, each episode has H rounds. The state at each round is denoted as $x_h \in \mathcal{X}, h \in [H]$. At each round of an episode, the agent selects action a_h from an action set $\mathcal{A}$. The environment generates a reward $r_h(x_h, a_h)$ and a cost $c_h(x_h, a_h)$ with $r_h \in \mathcal{R}$ and $c_h \in \mathcal{C}$. We model the dynamics as $x_{h+1} = f_h(x_h, a_h)$ where $f_h \in \mathcal{F}$ is a random transition function drawn from an unknown distribution $g(f_h \mid x_h, a_h)$ with the density $g \in \mathcal{G}$. The agent has no access to the random function f_h but can observe the state x_h at each round h. Note that we model the dynamics in a function style for ease of presentation, and this dynamic model can be translated into the transition probability in standard MDP models [383, 36] as $\mathbb{P}(x_{h+1} \mid x_h, a_h) = \sum_{f_h} \mathbb{1}(f_h(x_h, a_h) = x_{h+1}) g(f_h \mid x_h, a_h)$. A policy π is a function which gives the action a_h for each round $h \in [H]$. Let $V_h^\pi(x_h) = \mathbb{E}\left[\sum_{i=h}^H r_i(x_i, a_i))\right]$ denote the expected value of the total reward from round h by policy π. One objective of A-CMDP to maximize the expected total reward from the first round which is denoted as $\mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right] = \mathbb{E}\left[\sum_{h=1}^H r_h(x_h, a_h))\right]$.

Besides optimizing the expected total reward as in existing MDPs, A-CMDP also guarantees the anytime cost constraints comparing with a policy prior $\pi^\dagger$. The policy prior can be a policy that has verified cost performance in real systems or a heuristic policy with strong empirically-guaranteed cost performance, for which concrete examples will be given in the next section. Denote $y_h = (f_h, c_h, r_h)$, and $y_{1:H} = \{y_h\}_{h=1}^H \in \mathcal{Y} = \mathcal{F} \times \mathcal{R} \times \mathcal{C}$ is

a sampled sequence of the models in an A-CMDP. Let $J_h^\pi(y_{1:H}) = \sum_{i=1}^h c_i(x_i, a_i)$ be the cost up to round $h \in [H]$ with states $x_i, i \in [h]$ and actions $a_i, i \in [h]$ of a policy π. Also, let $J_h^{\pi^\dagger}(y_{1:H}) = \sum_{i=1}^h c_i(x_i^\dagger, a_i^\dagger)$ be the cost of the prior with states $x_i^\dagger, i \in [h]$ and actions $a_i^\dagger, i \in [h]$ of the prior $\pi^\dagger$. The anytime constraints are defined as below.

Definition 19 (Anytime constraints)**.** *If a policy π satisfies $(\lambda, b)-$anytime constraints, the cost of π never exceeds the cost of the policy prior $\pi^\dagger$ relaxed by parameters $\lambda \geq 0$ and $b \geq 0$, i.e. $J_h^\pi(y_{1:H}) \leq (1+\lambda) J_h^{\pi^\dagger}(y_{1:H}) + hb, \forall h \in [H], \forall y_{1:H} \in \mathcal{Y}$.*

Now, we can formally express the objective of A-CMDP with Π being the policy space as

$$\max_{\pi \in \Pi} \mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right], \quad s.t.\ \ J_h^\pi(y_{1:H}) \leq (1+\lambda) J_h^{\pi^\dagger}(y_{1:H}) + hb, \ \ \forall h \in [H], \forall y_{1:H} \in \mathcal{Y}. \tag{4.1}$$

Let $\Pi_{\lambda,b}$ be the collection of policies that satisfy the anytime constraints in (4.1). We design an anytime-constrained RL algorithm that explores the policy space $\Pi_{\lambda,b}$ in K episodes to optimize the expected reward $\mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right]$. Note that different from constrained/conservative MDPs [139, 168, 392, 423, 5, 422], the anytime constraints in (4.1) must be satisfied for any round in any sampled episode $y_{1:H} \in \mathcal{Y}$ given relaxed parameters $\lambda, b \geq 0$. To evaluate the performance of the learned policy $\pi^k \in \Pi_{\lambda,b}, k \in [K]$ and the impact of the anytime constraints, we consider the regret performance metric defined as

$$\text{Regret}(K) = \sum_{k=1}^K \mathbb{E}_{x_1}\left[V_1^{\pi^*}(x_1) - V_1^{\pi^k}(x_1)\right], \text{with } \pi^k \in \Pi_{\lambda,b} \tag{4.2}$$

where $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right]$ is the optimal policy without considering the anytime constraints. When λ or b becomes larger, the constraints get less strict and the algorithm

has more flexibility to minimize the regret in (4.2). Thus, the regret analysis will show the trade-off between optimizing the expected reward and satisfying the anytime cost constraint.

In this paper, we make additional assumptions on the cost functions, transition functions, and the prior policy which are important for the anytime-constrained algorithm design and analysis.

Assumption 4. *All the cost functions in the space $\mathcal{C}$ have a minimum value $\epsilon \geq 0$, i.e. $\forall(x,a), \forall h \in [H], c_h(x,a) \geq \epsilon \geq 0$, and are L_c-Lipschitz continuous with respect to action a_h and the state x_h. All the transition functions in the space $\mathcal{F}$ are L_f-Lipschitz continuous with respect to action a_h and the state x_h. The parameters ϵ, L_c and L_f are known to the agent.*

The Lipschitz continuity of cost function space $\mathcal{C}$ and transition function space $\mathcal{F}$ can also be found in other works on Lipschitz MDP [36, 161]. The Lispchitz assumptions actually apply to many continuous mission-critical systems like cooling systems, power systems and carbon-aware datacenters [104, 98, 326]. In these systems, the agents have no access to concrete cost and transition functions, but they usually have the knowledge of the minimum cost values and the Lipschitz constants of cost and dynamic functions by system evaluation, and thus can utilize the information to satisfy anytime constraints.

Definition 20 (Telescoping policy)**.** *A policy π satisfies the telescoping property if the policy is used from round h_1 to h_2 with initialized states x_{h_1} and x'_{h_1}, the corresponding states x_{h_2} and x'_{h_2} at round h_2 satisfies*

$$\|x_{h_2} - x'_{h_2}\| \leq p(h_2 - h_1)\|x_{h_1} - x'_{h_1}\|, \tag{4.3}$$

where $p(h)$ is called a perturbation function with h and $p(0) = 1$.

Assumption 5. *The prior policy $\pi^\dagger$ satisfies the telescoping property with some perturbation function p. Furthermore, $\pi^\dagger$ is Lipschitz continuous.*

The telescoping property in Definition 20 is useful, because if it is satisfied by a policy, then with an initial state perturbation at a fixed round, the maximum divergence of the states afterwards by employing the policy can be bounded. The telescoping policies are assumed or verified for many policy priors for online control and decision makings [387, 269, 268, 245]. A stable policy typically satisfies the telescoping property with small enough perturbations $p(h), h \in [H]$ [387, 253].

4.3.2 Motivating Examples

The anytime constraints have direct motivations from many mission-critical control systems.

Constrained Cooling Control in Data Centers. In mission-critical infrastructures like data centers, the agent needs to make decisions on cooling equipment management to maintain temperature range and achieve high energy efficiency. Over many years, rule-based policies have been used in cooling systems and have verified cooling performance in maintaining a suitable temperature for computing [282]. Recently, RL algorithms are developed for cooling control in data centers to optimize the energy efficiency [409, 104, 282]. The safety concerns of RL policies, however, hinder their deployment in real systems. In data centers, an unreliable cooling policy can overheat devices and denial critical services, causing a huge loss [120, 282]. The safety risk is especially high at the early exploration stage of RL in the real environment. Therefore, it is crucial to guarantee the constraints

on cooling performance at anytime in any episode for safety. With the reliable rule-based policies as control priors, A-CMDP can accurately model the critical parts of the cooling control problem, opening a path towards learning reliable policy for cooling of data centers.

Workload Scheduling in Carbon-Intelligent Computing. The world is witnessing a growing demand for computing power due to new computing applications like AI. The large carbon footprint of computing has become a problem that cannot be ignored [326, 410, 345, 132]. Studies find that the amount of carbon emission per kilowatt-hour on electricity grid varies with time and locations due to the various types of electricity generation [240, 219, 77]. Exploiting the time-varying property of carbon efficiency, recent studies are developing workload scheduling policies (e.g. delay some temporally flexible workloads) to optimize the total carbon efficiency [326]. However, an unreliable workload scheduling policy in warehouse-scale computers can cause a large computing latency, resulting in an unsatisfactory Quality of Service (QoS). Thus, to achieve a high carbon efficiency while guaranteeing a low computing latency, we need to solve an A-CMDP which leverages RL to improve the carbon efficiency while always guaranteeing the QoS requirement compared with a policy prior focusing on computing latency [122, 174, 95, 442, 441]. This also resembles the practice of carbon-intelligent computing adopted by Google [326].

4.4 Methods

In this section, we first propose a projection-based algorithm to guarantee the anytime constraints given any RL policy. Then, we model a new MDP based on the projection-based algorithm and give an RL algorithm to solve it.

4.4.1 Guarantee the Anytime Constraints

It is challenging to guarantee the anytime constraints in (4.1) for an RL policy in any episode due to the following. First of all, in MDPs, the agent can only observe the *real* states $\{x_h\}_{h=1}^H$ corresponding to the truly-selected actions $\{a_h\}_{h=1}^H$. The agent does not select the actions $a_h^\dagger$ of the prior, so the states of the prior $x_h^\dagger$ are *virtual* states that cannot be observed. Thus, the agent cannot evaluate the costs of the prior $j_h^{\pi^\dagger}$ which is in the anytime constraint at each round h. Also, the action at each round h has an impact on the costs in the future rounds $i, i > h$ based on the random transition models $f_i, i \geq h$. Thus, besides satisfying the constraints in the current round, we need to have good planning for the future rounds to avoid any possible constraint violations even without exact knowledge of transition and/or cost models. Additionally, the RL policy may be arbitrarily bad in the environment and can give high costs (especially when very limited training data is available), making constraint satisfaction even harder.

Despite the challenges, we design safe action sets $\{\mathcal{A}_h, h \in [H]\}$ to guarantee the anytime constraints: if action a_h is strictly selected from $\mathcal{A}_h$ for each round h, the anytime constraints for all rounds are guaranteed. As discussed above, the anytime constraints cannot be evaluated at any time since the policy prior's state and cost information is not available. Thus, we propose to convert the original anytime constraints into constraints that only depend on the action differences between the real policy and the policy prior and the known information. We give the design of the safe action sets based on the next proposition. For the ease of presentation, we denote $c_i = c_i(x_i, a_i)$ as the real cost and $c_i^\dagger = c_i(x_i^\dagger, \pi(x_i^\dagger))$ as the cost of the policy prior at round i.

Proposition 21. *Suppose that Assumption 4 and 5 are satisfied. At round h with costs $\{c_i\}_{i=1}^{h-1}$ observed, the anytime constraints $J_{h'}^{\pi} \leq (1+\lambda)J_{h'}^{\pi^\dagger} + h'b$ for rounds $h' = h, \cdots, H$ are satisfied if for all subsequent rounds $h' = h, \cdots, H$,*

$$\sum_{j=h}^{h'} \Gamma_{j,j} \|a_j - \pi^\dagger(x_j)\| \leq G_{h,h'}, \ \forall h' = h, \cdots, H, \tag{4.4}$$

where $\Gamma_{j,n} = \sum_{i=n}^{H} q_{j,i}, (j \in [H], \forall n \geq j)$, with $q_{j,i} = L_c \mathbb{1}(j=i) + L_c(1+L_{\pi^\dagger})L_f p(i-1-j)\mathbb{1}(j<i), (\forall j \in [H], i \geq j)$, relying on known parameters, and $G_{h,h'}$ is called the allowed deviation which is expressed as

$$G_{h,h'} = \sum_{i=1}^{h-1} \Big((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h} d_i \Big) + (h' - h + 1)(\lambda\epsilon + b), \tag{4.5}$$

where $\hat{c}_i^\dagger = \max\Big\{\epsilon, c_i - \sum_{j=1}^{i} q_{j,i} d_j\Big\}, (\forall i \in [H])$, is the lower bound of of $c_i^\dagger$, and $d_j = \|a_j - \pi^\dagger(x_j)\|, \forall j \in [H]$ is the action difference at round j.

At each round $h \in [H]$, Proposition 21 provides a sufficient condition for satisfying all the anytime constraints from round h to round H given in (4.1). The meanings of the parameters in Proposition 21 are explained as follows. The weight $q_{j,i}$ measures the impact of action deviation at round j on the cost difference $|c_i - c_i^\dagger|$ at round $i \geq j$, and the weight $\Gamma_{j,n}$ indicates the total impact of the action deviation at round j on the sum of the cost differences from rounds n to round H. Based on the definition of $q_{j,i}$, we get $\hat{c}_i^\dagger$ as a lower bound of the prior cost $c_i^\dagger$. With these bounds, we can calculate the maximum allowed total action deviation comparing with the prior actions $\pi^\dagger(x_j)$ from round $j = h$ to h' as $G_{h,h'}$

By applying Proposition 21 at initialization, we can guarantee the anytime constraints for all rounds $h' \in [H]$ if we ensure that for all rounds $h' \in [H]$, $\sum_{j=1}^{h'} \Gamma_{j,j} \|a_j - \pi^\dagger(x_j)\| \leq G_{1,h'} = h'(\lambda\epsilon + b)$. This sufficient condition is a long-term constraint relying on

minimum cost value ϵ and the relaxation parameters λ and b. Although we can guarantee the anytime constraints by the sufficient condition obtained at initialization, we apply Proposition 21 at all the subsequent rounds with the cost feedback information to get larger action sets and more flexibility to optimize the average reward. In this way, we can update the allowed deviation according to the next corollary.

Corollary 22. *At round 1, we initialize the allowed deviation as* $D_1 = \lambda\epsilon + b$. *At round* $h, h > 1$, *the allowed deviation is updated as*

$$D_h = \max\left\{D_{h-1} + \lambda\epsilon + b - \Gamma_{h-1,h-1}d_{h-1},\ R_{h-1} + \lambda\epsilon + b\right\} \tag{4.6}$$

where $R_{h-1} = \sum_{i=1}^{h-1}\left((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h}d_i\right)$ *with notations defined in Proposition 21. The* $(\lambda, b)-$*anytime constraints in Definition 19 are satisfied if it holds at each round* h *that* $\Gamma_{h,h}\|a_h - \pi^\dagger(x_h)\| \leq D_h$.

Corollary 22 gives a direct way to calculate the allowed action deviation at each round. In the update rule (4.6) of the allowed deviation, the first term of the maximum operation is based on the deviation calculation at round $h-1$ while the second term is obtained by applying Proposition 21 at round h. By Corollary 22, we can define the safe action set at each round h as

$$\mathcal{A}_h(D_h) = \left\{a \mid \Gamma_{h,h}\|a - \pi^\dagger(x_h)\| \leq D_h\right\}. \tag{4.7}$$

With the safe action set design in (4.7), we propose a projection-based algorithm called `ACD` in Algorithm 4. We first initialize an allowed deviation as $D_1 = \lambda\epsilon + b$. When the output $\tilde{a}_h$ of the ML model is obtained at each round h, it is projected into a safe

Algorithm 4 Anytime-Constrained Decision-making (ACD)

Initialization: Initialize an allowed deviation: $D_1 = \lambda\epsilon + b$.

for $h = 1, \cdots, H$ **do**

Obtain the output of the ML policy $\tilde{\pi}$ as $\tilde{a}_h$.

Select the action a_t by projecting $\tilde{a}_h$ into the safe action set $\mathcal{A}_h(D_h)$ in (4.7).

Update the allowed deviation D_{h+1} by (4.6).

end for

action set $\mathcal{A}_h(D_h)$ depending on the allowed deviation D_h, i.e. $a_h = P_{\mathcal{A}_h(D_h)}(\tilde{a}_h) = \arg\min_{a \in \mathcal{A}_h(D_h)} \|a - \tilde{a}_h\|$. The projection can be efficiently solved by many existing methods on constrained policy learning or machine learning [422, 87, 24, 256, 133]. The allowed deviation is then updated based on Corollary 22. Intuitively, if the actions are closer to the prior actions before h, i.e. the action deviations $\{d_i\}_{i=1}^{h-1}$ get smaller, then R_{h-1} becomes larger and D_h becomes larger, leaving more flexibility to deviate from $a_i^\dagger, i \geq h$ in subsequent rounds.

4.4.2 Anytime-Constrained RL

The anytime constraints have been satisfied by Algorithm 4, but it remains to design an RL algorithm to optimize the average reward under the anytime cost constraints, which is given in this section.

The anytime-constrained decision-making algorithm in Algorithm 4 defines a new MDP, with an additional set of allowed deviations $\mathcal{D}$ to the A-MDP defined in Section 4.3.1, denoted as $\tilde{\mathcal{M}}(\mathcal{X}, \mathcal{D}, \mathcal{A}, \mathcal{F}, g, H, r, c, \tilde{\pi}, \pi^\dagger)$. In the new MDP, we define an augmented state s_h which include the original state x_h, the allowed deviation $D_h \in \mathcal{D}$, and history

information $\{c_i\}_{i=1}^{h-1}$ and $\{d_i\}_{i=1}^{h-1}$. The transition of x_h is defined by f_h in Section 4.3.1 and needs to be learned while the transition of D_h is defined in (4.6) and is known to the agent. The ML policy $\tilde{\pi}$ gives an output $\tilde{a}_h$ and the selected action is the projected action $a_h = P_{\mathcal{A}_h(D_h)}(\tilde{a}_h)$. Then the environment generates a reward $r_h(x_h, P_{\mathcal{A}_h(D_h)}(\tilde{a}_h))$ and a cost $c_h(x_h, P_{\mathcal{A}_h(D_h)}(\tilde{a}_h))$. Thus, the value function corresponding to the ML policy $\tilde{\pi}$ can be expressed as $\tilde{V}_h^{\tilde{\pi}}(s_h) = \mathbb{E}\left[\sum_{i=h}^{H} r_i(x_i, P_{\mathcal{A}_h(D_h)}(\tilde{a}_h))\right]$ with $\tilde{a}_h$ being the output of the ML policy $\tilde{\pi}$. For notation convenience, we sometimes write the actions of π^* and $\pi^\dagger$ as $\pi^*(s)$ and $\pi^\dagger(s)$ even though they only reply on the original state x in s.

To solve the MDP, we propose a model-based RL algorithm called `ACRL` in Algorithm 5. Different from the existing model-based RL algorithms [312, 42, 440], `ACRL` utilize the dynamic model of A-CMDP and `ACD` (Algorithm 4) to reduce the learning complexity for A-CMDP. Given an estimation of the transition distribution $\hat{g}^k$ at episode k, we perform value iteration to update $\tilde{Q}$ functions for $h = 1, \cdots, H$.

$$\begin{aligned} &\tilde{Q}_h^k(s_h, \tilde{a}_h) = r_h(x_h, a_h) + \max_{g \in \mathcal{G}_k} \mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) \mid s_h, a_h\right],\ \tilde{V}_h^k(s_h) = \max_{a \in \mathcal{A}} \tilde{Q}_h^k(s_h, a), \\ &\mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) \mid s_h, a_h\right] = \sum_{f \in \mathcal{F}} \tilde{V}_{h+1}^k(f(x_h, a_h), D_{h+1}) g(f \mid s_h, a_h), \end{aligned} \tag{4.8}$$

where $a_h = P_{\mathcal{A}_h(D_h)}(\tilde{a}_h)$, $\tilde{Q}_{H+1,k}(s, a) = 0, \tilde{V}_{H+1,k}(s) = 0$, $\mathcal{G}_k$ is the confidence set based on the estimation of the transition model g. The transition model g is fitted as

$$\hat{g}^k = \arg\min_{g \in \mathcal{G}} \sum_{i=1}^{k-1} \sum_{h=1}^{H} \left(\mathbb{E}_g\left[\tilde{V}_{h+1}^i(s_{h+1}) \mid s_h, a_h\right] - \tilde{V}_{h+1}^i(s_{h+1}, a_{h+1})\right)^2. \tag{4.9}$$

Based on the transition estimation, we can calculate the confidence set as

$$\mathcal{G}_k = \left\{ g \in \mathcal{G} \,\middle|\, \sum_{i=1}^{k-1} \sum_{h=1}^{H} \left(\mathbb{E}_g\left[\tilde{V}_{h+1}^i(s_{h+1}) \mid s_h, a_h\right] - \mathbb{E}_{\hat{g}^k}\left[\tilde{V}_{h+1}^i(s_{h+1}) \mid s_h, a_h\right]\right)^2 \le \beta_k \right\}, \tag{4.10}$$

Algorithm 5 Anytime-Constrained Reinforcement Learning (`ACRL`)

Initialization: Transition model set $\mathcal{G}_1 = \{\hat{g}^1\}$.

for each episode $k = 1, \cdots, K$ **do**

  Observe the initial state s_1^k.

  Select $g^k = \arg\max_{g \in \mathcal{G}^k} \mathbb{E}_g \left[V_1(s_1^k)\right]$.

  Perform value iteration in Eqn. (4.8) and update $\tilde{Q}$ functions $\tilde{Q}_1^k \cdots, \tilde{Q}_H^k$.

  for each round $h = 1, \cdots, H$ **do**

    Run `ACD` (Algorithm 4) by ML policy $\tilde{\pi}^k(s_h) = \arg\max_{a \in \mathcal{A}} \tilde{Q}_h^k(s_h, a)$

    Observe state s_{h+1}^k and store values $\tilde{V}_{h+1}^k(s_{h+1}^t)$.

  end for

  Update transition model $\hat{g}^{k+1}$ using (4.9) and calculate confidence set $\mathcal{G}_{k+1}$.

end for

where $\beta_k > 0$ is a confidence parameter.

With a learned ML policy $\tilde{\pi}^k$ at each episode k, the policy used for action selection is the `ACD` policy π^k. Given the optimal ML policy $\tilde{\pi}^* = \arg\max_{\tilde{\pi} \in \tilde{\Pi}} \tilde{V}_1^{\tilde{\pi}}(s_1)$ with $\tilde{\Pi}$ being the ML policy space, the optimal `ACD` policy is denoted as π°. For an augmented state s_h at round h, they select action as

$$\pi^k(s_h) = P_{\mathcal{A}_h(D_h)}(\tilde{\pi}^k(s_h)),\ \pi^\circ(s_h) = P_{\mathcal{A}_h(D_h)}(\tilde{\pi}^*(s_h)). \tag{4.11}$$

4.5 Performance Analysis

In this section, we analyze the regret of `ACRL` to show the impacts of anytime cost constraints as well as the reinforcement learning process on the average reward.

4.5.1 Regret Due to Constraint Guarantee

Intuitively, due to the anytime constraints in Eqn. (4.1), there always exists an unavoidable reward gap between an `ACD` policy and the optimal-unconstrained policy π^*. In this section, to quantify this unavoidable gap, we bound the regret of the optimal `ACD` policy π°, highlighting the impact of anytime cost constraints on the average reward performance.

Theorem 4.5.1. *Assume that the optimal-unconstrained policy π^* has a value function $Q_h^{\pi^*}(x,a)$ which is $L_{Q,h}$-Lipschitz continuous with respect to the action a for all x. The regret between the optimal `ACD` policy π° that satisfies $(\lambda, b)-$anytime constraints and the optimal-unconstrained policy π^* is bounded as*

$$\mathbb{E}_{x_1}\left[V_1^{\pi^*}(x_1) - V_1^{\pi^\circ}(x_1)\right] \leq \mathbb{E}_{y_{1:H}}\left\{\sum_{h=1}^{H} L_{Q,h}\left[\eta - \frac{1}{\Gamma_{h,h}}(\lambda\epsilon + b + \Delta G_h)\right]^+\right\}, \tag{4.12}$$

where $\eta = \sup_{x\in\mathcal{X}} \|\pi^*(x) - \pi^+(x))\|$ is the maximum action discrepancy between the policy prior $\pi^\dagger$ and optimal-unconstrained policy π^*; $\Gamma_{h,h}$ is defined in Proposition 21; $\Delta G_h = [R_{h-1}]^+$ is the gain of the allowed deviation by applying Proposition 21 at round h.

The regret bound stated in Theorem 4.5.1 is intrinsic and inevitable, due to the committed assurance of satisfying the anytime constraints. Such a bound cannot be improved via policy learning, i.e., converge to 0 when the number of episodes $K \to \infty$. This is because to satisfy the $(\lambda, b)-$anytime constraints, the feasible policy set $\Pi_{\lambda,b}$ defined

under (4.1) is a subset of the original policy set Π, and the derived regret is an upper bound of $\max_{\pi\in\Pi}\mathbb{E}_{x_1}\left[V_1^{\pi}(x_1)\right] - \max_{\pi\in\Pi_{\lambda,b}}\mathbb{E}_{x_1}\left[V_1^{\pi}(x_1)\right]$. Moreover, the regret bound relies on the action discrepancy η. This is because if the optimal-unconstrained policy π^* is more different from the prior $\pi^\dagger$, its actions are altered to a larger extent to guarantee the constraints, resulting in a larger degradation of reward performance. More importantly, the regret bound indicates the trade-off between the reward optimization and anytime constraint satisfaction governed by the parameters λ and b. When λ or b becomes larger, we can get smaller regret because the anytime constraints in (4.1) are relaxed to have more flexibility to optimize the average reward. In the extreme cases when λ or b is large enough, all the policies in Π can satisfy the anytime constraints, so we can get zero regret.

Moreover, the regret bound shows that the update of allowed deviation by applying Proposition 21 based on the cost feedback at each round will benefit the reward optimization. By the definition of R_{h-1} in Corollary 22, if the real actions deviate more from the prior actions before h, the gain ΔG_i for $i \geq h$ can be smaller, so the actions must be closer to the prior actions in the subsequent rounds, potentially causing a larger regret. Thus, it is important to have a good planing of the action differences $\{d_i\}_{i=1}^{H}$ to get larger allowed deviation for reward optimization. Exploiting the representation power of machine learning, `ACRL` can learn a good planning of the action differences, and the `ACD` policy π° corresponding to the optimal ML policy $\tilde{\pi}^*$ can achieve the optimal planing of the action differences.

4.5.2 Regret of `ACRL`

To quantify the regret defined in Eqn. (4.2), it remains to bound the reward gap between the `ACD` policy π^k and the optimal `ACD` policy π°. In this section, we show that π^k

by ACRL approaches the optimal one π° as episode $K \to \infty$ by bounding the pseudo regret

$$\mathrm{PReg}(K) = \mathbb{E}_{x_1}\left[\sum_{k=1}^{K}\left(V_1^{\pi^\circ}(s_1) - V_1^{\pi^k}(s_1)\right)\right]. \tag{4.13}$$

Theorem 4.5.2. *Assume that the value function is bounded by $\bar{V}$. Denote a set of function as*

$$\mathcal{Q} = \left\{q \mid \exists g \in \mathcal{G}, \forall (s,a,v) \in \mathcal{S}\times\mathcal{A}\times\mathcal{V}, q(s,a,v) = \int \mathbb{E}_g\left[v(\mathrm{d}s') \mid s,a\right]\right\}. \tag{4.14}$$

If $\beta_k = 2(\bar{V}H)^2 \log\left(\frac{2\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)}{\delta}\right) + C\bar{V}H$ with $\alpha = 1/(KH\log(KH/\delta))$, C being a constant, and $\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)$ being the covering number of $\mathcal{Q}$, with probability at least $1-\delta$, the pseudo regret of Algorithm 5 is bounded as

$$\mathrm{PReg}(K) \le 1 + d_{\mathcal{Q}}H\bar{V} + 4\sqrt{d_{\mathcal{Q}}\beta_K KH} + H\sqrt{2KH\log(1/\delta)}, \tag{4.15}$$

where $d_{\mathcal{Q}} = \dim_E(\mathcal{Q}, \frac{1}{KH})$ is the Eluder dimension of $\mathcal{Q}$ defined in [338].

Theorem 4.14 bounds the pseudo regret for each episode k. Existing works on function approximation of reinforcement learning show that the regret is sublinear with assumptions on the transition model space [42, 102, 142, 440]. The concrete sublinear regret bounds under different model space assumptions are of independent interest of this paper, but we give the regret bound when the transition model g can be represented by a linear kernel as in [42, 440], i.e. $g(f \mid s,a) = \langle\phi(f\mid,s,a),\theta\rangle$ with dimension of θ as d_θ. Under this assumption, we have $\beta_K = O((\bar{V}H)^2\log(\frac{1}{\delta}\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty))) = O\left((\bar{V}H)^2\log(\frac{1}{\delta}(1/\alpha)^{d_\theta}\right) = \tilde{O}((\bar{V}H)^2(d_\theta + \log(1/\delta)))$ [42], and the Eluder dimension is $d_{\mathcal{Q}} = \tilde{O}(d_\theta)$ [338]. Thus the pseudo regret is $\mathrm{PReg}(K) = \tilde{O}(\sqrt{H^3\bar{V}^2K\log(1/\delta)})$ which is sublinear in terms of K.

With the sublinear pseudo regret, the ACD policy π^k performs as asymptotically well as the optimal ACD policy π° when $K \to \infty$. Combining with the regret of the optimal

ACD policy in Theorem 4.5.1, we can bound the regret of ACRL as

$$\text{Regret}(K) \leq K\mathbb{E}_{y_{1:H}} \left\{ \sum_{h=1}^{H} L_{Q,h} \left[\eta - \frac{1}{\Gamma_{h,h}} (\lambda\epsilon + b + \Delta G_h) \right]^+ \right\} + \tilde{O}(\sqrt{H^3 \bar{V}^2 K \log(1/\delta)}), \tag{4.16}$$

which includes an unavoidable part due to the commitment to satisfy the anytime constraints as illustrated in Theorem 4.5.1 and a sublinear part induced by policy learning. The first regret term indicates the trade-off between the optimization of the average reward and the anytime constraint satisfaction while the second term shows the effect of ACRL in improving the reward under the anytime constraints.

We also experiment with the application of resource management for carbon-aware computing [326] to empirically show the benefits of ACRL. Unlike the existing algorithms that either result in a low reward or violate the anytime Quality-of-Service constraint, our results demonstrate that ACRL can improve the average performance in terms of the operational cost while still offering guaranteed QoS.

4.6 Empirical Results

4.6.1 Problem Formulation

We consider the sustainable workload scheduling problem in datacenters to jointly optimize carbon efficiency and revenue while guaranteeing the quality-of-service (QoS). In this problem, the average carbon efficiency and revenue can be optimized while QoS must always be ensured at each step. The agent needs to decide the computing resource a_h measured by energy (kWh) for each round h. The state x_h is the remaining demand for

each round h and is updated as

$$x_h = f(x_{h-1}, \mu_h, a_h) = [V_x(x_{h-1}) + \mu_h - V_a(a_h)]^+, \tag{4.17}$$

where V_x is a random function of x_{h-1} measuring the randomly decayed remaining demands (e.g., due to workload dropping), μ_h is arrival demand at round h, and V_a is a random function in terms of a_h and x_h and outputs the amount of processed workload. With the random functions V_x and V_a, the remaining workload x_h at round h is drawn from $\mathbb{P}(x_h \mid x_{h-1}, \mu_h, a_h)$. Here, we focus on flexibly deferrable workloads (e.g., model training and batch data processing) [326].

The energy efficiency reward is modeled by a penalty for the carbon footprint at each round. Let C_h be the amount of renewable at round h, the energy efficiency reward is expressed as $\text{efficiency}_h = -([a_h - C_h]^+)^2$. The revenue function is modeled as a general power-law function [349] as $\text{revenue}_h = C_r V_a^{\alpha}(a_h)$ with $\alpha \in (0, 1)$. In datacenters, we also need to consider a switching cost $\gamma_2 \|a_h - a_{h-1}\|$ at each round h to avoid switching on/off servers frequently. Thus, the reward in this problem is formulated as

$$\text{reward}_h = \text{efficiency}_h + \gamma_1 \cdot \text{revenue}_h - \gamma_2 \cdot \|a_h - a_{h-1}\|^2. \tag{4.18}$$

Besides the reward, QoS is also crucial for deferrable workloads in datacenters. In this work, we model QoS as a cost function of the remaining demand as follows:

$$cost_{\text{QoS},h} = x_h^\top Q_1 x_h + Q_2^\top x_h + Q_3, \tag{4.19}$$

where Q_1, Q_2 and Q_3 are constants.

As shown in Eqn. (4.1), given a baseline $\pi^\dagger$ that has been verified to achieve a satisfactory QoS, our goal is to optimize the expected reward and guarantee the QoS for

any time in any sequence, i.e.

$$\begin{aligned} &\min_{\pi\in\Pi} \quad \mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right], \\ &s.t. \quad \sum_{h=1}^{h'} cost_{\text{QoS},h}(\pi) \leq (1+\lambda)\sum_{h=1}^{h'} cost_{\text{QoS},h}(\pi^\dagger) + h'b, \quad \forall h' \in [H], \end{aligned} \tag{4.20}$$

which is consistent with the definition of anytime constraints in Definition 19.

4.6.2 Baselines

In the experiments, we consider different baselines as below.

• QoS Optimization (`OPT-QoS`): This baseline policy prior directly optimizes QoS in (4.19) based on estimated models $\hat{V}_x$ and $\hat{V}_a$. Without taking efficiency or revenue into consideration, `OPT-QoS` essentially always schedules as many computing resources as possible to lower the QoS cost based on the estimated arrival demand.

• Reinforcement Learning (`RL`): This is a model-based reinforcement learning algorithm [42] to optimize the expected reward $\mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right]$ without considering any QoS constraints.

• Constrained Reinforcement Learning (`CRL`): This is a constrained reinforcement learning to optimize the reward with the expected QoS cost constraint as shown below:

$$\min_{\pi\in\Pi} \quad \mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right], \quad s.t. \mathbb{E}\left[\sum_{h=1}^{H} cost_{\text{QoS},h}(\pi) - (1+\lambda)\sum_{h=1}^{H} cost_{\text{QoS},h}(\pi^\dagger)\right] \leq B. \tag{4.21}$$

• Random RL policy with `ACD` (`Random` +`ACD`): This algorithm selects actions by `ACD` in Algorithm 4 with a random RL policy $\tilde{\pi}$ as input of `ACD`.

• Trained RL policy with `ACD` (`RL` +`ACD`): This algorithm selects actions by `ACD` in Algorithm 4 with the RL policy trained to optimize the expected reward without accounting for QoS.

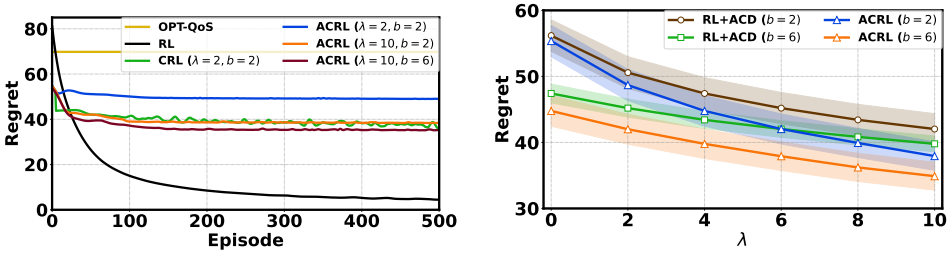
• Anytime-Constrained Reinforcement Learning (`ACRL`): This is the proposed Algorithm 5 which optimizes the expected reward while guaranteeing the anytime QoS cost constraints in (4.1). In each inference, `ACD` in Algorithm 4 is used to select actions.

4.6.3 Experiment Settings

In the experiments, we evaluate the performances with the following experiment settings.

System Parameters. We evaluate the regret and the cost constraints for different choices of parameters. The results are given for different anytime constraint parameters including λ chosen from $[0, 10]$ and b chosen from $\{2, 6\}$. With smaller λ and b, we have more stringent constraints, and vice versa. In the experiments, we choose $\alpha = 0.5$ for the revenue function to simulate a typical effect of the scheduled resource on the revenue. To scale different rewards into the same magnitude, we choose the weight for the revenue as $\gamma_1 = 4$, and the weight for the switching cost as $\gamma_2 = 1$. For the QoS cost function, we choose $Q_1 = Q_2 = Q_3 = 1$, so we have the minimum QoS cost as $\epsilon = Q_3 = 1$. The transition model f are from a function space defined by random functions V_x and V_a. To create the environment for RL, $V_x(x_h)$ is drawn from a uniform distribution with range $[0.9 \cdot x_h, x_h]$, and $V_a(a_h)$ is drawn from a normal distribution with $0.8 \cdot a_h$ as the center.

Data. For experiments, we create an environment based on a renewable dataset and a demand dataset. The renewable dataset is a public dataset from California Independent System Operator [310] which contains the hourly renewable generation in 2019. The renewable sequences from multiple sources (solar, wind, water) are summed together and scaled to be the renewable of $\{C_h\}_{h=1}^{H}$ in the problem formulation. In addition, we use the



(a) Regret per episode

(b) Regret w.r.t. λ

Figure 4.1: Regret of different algorithms.

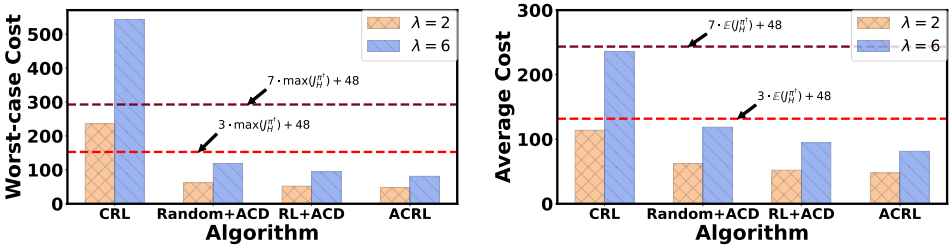
Azure Cloud Dataset [115] as the demand dataset which includes hourly CPU utilization in the same year of 2019. We choose the sequences of the first three months and augment them to 4000 episodes for policy exploration, and we hold out the sequences of the last two months for testing.

Learning Settings. To ensure fair comparisons, we choose the same neural network architecture as the policy network for different methods. The policy neural network has two hidden layers and each hidden layer has 40 neurons. For training, the policy network parameters are initialized by Gaussian distribution. The reinforcement learning has total $K = 4000$ episodes. We update the neural network every 50 episodes with a weight update rate of 10^{-3}. We apply Adam optimizer to update the weights of neural networks.

4.6.4 Results

We show the empirical results for both regret and QoS cost.

Regret Evaluation. The reward regrets as defined in Eqn. (4.2) are given in Figure 4.1. To evaluate the regret, we use the `RL` policy after the exploration for total 4000



(a) Worst-case cost

(b) Average cost

Figure 4.2: QoS costs of different algorithms.

episodes as the optimal RL policy π^*. The results are given for different anytime constraint parameters λ and b

Figure 4.1(a) shows the varying regret of different algorithms for the first 500 episodes. Without including reward as an objective, the policy prior `OPT-QoS` is an algorithm that is not updated over time and always gives the highest regret. Without the QoS cost constraints, `RL` approaches the optimal RL policy that can give the best regret as time goes on. The constrained RL `CRL` and anytime-constrained RL `ACRL` are guaranteed to satisfy the expected constraint in (4.21) and the anytime constraints in (4.1), respectively, so their reward regrets are higher than `RL`. Also, we can find that with larger λ and/or b, `ACRL` can achieve lower regret after about 200 episodes. This is because the larger λ or b gives less stringent anytime constraints and leave more flexibility to reduce regret as shown in Theorem 4.5.1. Moreover, we can observe that `ACRL` with small λ and b (e.g. $\lambda = 2, b = 2$ in the figure) converges faster to the optimal policy under the anytime constraints. The reason is that the constraints with small λ and b provide small policy space to explore, resulting in a smaller Eluder dimension $d_{\mathcal{Q}}$ shown in Theorem 4.5.2.

In Figure 4.1(b), we give the optimal regret of the algorithms after enough exploration. `ACRL` and `RL` +`ACD` are different in terms of the RL policy $\tilde{\pi}$ used in Algorithm 4. The regret of `Random` +`ACD` uses randomly initialized RL policy as $\tilde{\pi}$ and has a regret as large as from 61 to 64.51, exceeding the limit of the regret axis. The optimal regrets of different algorithms decrease the parameter λ given a fixed b, and a smaller b gives smaller optimal regrets given a fixed λ, which is consistent with the findings in Theorem 4.5.1. Importantly, we can find that under the same λ and b, `ACRL` can always improve the regret of `RL` +`ACD`. This is because `RL` explores the original A-CMDP defined in Section 4.3.1 while `ACRL` explores the new MDP defined at the beginning of Section 4.4.2, which represents the true environment created by `ACD`. This highlights the advantage of `ACRL` in terms of reducing the reward regret by learning with awareness of the anytime constraints in the true environment.

Cost Evaluation.

Figure 4.2(a) gives the worst-case cost of different algorithms for all the sequences in the testing dataset. The `RL` algorithm without considering the cost objective achieves the worst-case cost of 5015.62, exceeding the range of the cost axis to a large extent. The dotted horizon lines show the maximum cost bound required by the anytime constraint in Definition 19. Clearly, we can find that given any RL policy (even a randomly initialized RL policy) as input `ACD` can guarantee the anytime constraints even for the worst-case, but `CRL` fails to guarantee the anytime constraints. In the experiments, we verify that all `ACD` algorithms have zero violation of anytime constraints no matter what ML policy is used, but `CRL` has a violation rate of 27.5% for $\lambda = 2$ and a violation rate of 58.6% for $\lambda = 6$.

Figure 4.2(b) shows the average cost of different algorithms for all the sequences in the testing dataset. The `RL` algorithm achieves an average QoS cost of 2070.14, exceeding the range of the cost axis to a large extent. The dotted lines give the maximum average QoS cost bound required by the expected constraint in (4.21). Since `CRL` is designed to optimize the regret subject to the average QoS constraints in (4.21), it has no violation in terms of the average QoS cost. The algorithms that use `ACD` guarantees the stricter anytime constraints than the average constraint, so they can also guarantee the average QoS constraint.

4.7 Concluding Remarks

This chapter considers a novel MDP setting called A-CMDP where the goal is to optimize the average reward while guaranteeing the $(\lambda, b)-$anytime constraints. The anytime constraints require the cost of a learned policy never exceed that of a policy prior $\pi^\dagger$ for any round h in any episode. To guarantee the anytime constraints, we design `ACD`, which projects the output of an ML policy into a safe action set at each round. Then, to optimize the average reward under the anytime constraints, we formulate the decision process of `ACD` as a new MDP and propose an efficient model-based RL algorithm to explore the optimal RL policy in `ACD`. Our performance analysis shows that the regret compared with the optimal-unconstrained policy includes a term quantifying the impact of $(\lambda, b)-$anytime constraints on the average reward optimization and a term showing the effectiveness of policy learning.

Chapter 5

Learning-Assisted Online Optimization With Budget Constraints

5.1 Introduction

Online optimization with budget (or inventory) constraints, also referred to as OOBC, is an important problem modeling a wide range of sequential decision-making applications with limited resources, such as online virtual machine resource allocation [213, 315], one-way trading in economics [140], resource management in wireless networks [304, 231], and data center server provisioning [165]. More specifically, when virtualizing a physical server into a small number of virtual machines (VMs) to satisfy the demand of multiple sequentially-arriving jobs, the agent must make sure that the total VM resource

consumption is no more than what the physical server can provide [395]. Under such resource constraints, the agent decides the VM allocations to optimize the reward of demand satisfaction.

In an OOBC problem, online actions are selected sequentially to maximize the total utility over a short time horizon while the resource consumption over the time horizon is strictly constrained by a fixed amount of budgets (i.e., violating the budget constraint is naturally prohibited due to physical constraints). Consequently, the *short* time horizon (e.g., 24 hourly decisions in a day) and the *strict* budget constraint present substantial algorithmic challenges — the optimal solution relies on complete offline context information, but in the online setting, only the online contexts are revealed and the exact future contexts are unavailable for decision making [266, 265].

A relevant but different problem is online optimization with (long-term) constraints [48, 146, 304, 43]. In the literature, a common approach is to relax the long-term capacity constraints and include them as additional weighted costs into the original optimization objective, i.e., Lagrangian relaxation [126, 447, 48, 304]. The Lagrangian multiplier can be interpreted as the resource *price* [315], and is updated at each time step by a manually-designed algorithm such as Dual Mirror Descent (DMD) [48, 405, 210]. These algorithms require a sufficiently long time horizon for convergence, which hence may not provide satisfactory performance for short-term budget constraints, especially when contexts in an episode are not identically independently distributed (i.i.d.). Additionally, some studies consider constraints on average (i.e., equivalently, long-term constraints) [324, 191] or bound the violation of the constraints [304, 429, 376]. Thus, these algorithms consider different

settings and do not apply to *strict* budget constraints over a short time horizon, and we need to design new algorithms for settings with short time horizon.

The challenges of OOBC with short-term and strict budget constraints can be further highlighted by that competitive online algorithms have only been proposed very recently under settings with linear constraints [266, 265]. Concretely, CR-Pursuit algorithms are proposed to make actions by following a pseudo-optimal algorithm based on the competitive ratio pursuit framework. Nonetheless, to make sure the solution exists for each OOBC episode, the guaranteed competitive ratio (ratio between the algorithm cost and the offline-optimal cost) can be large. Also, they treat each OOBC problem instance as a completely new one and focus on the worst-case competitive ratio without considering the available historical data obtained when solving previous OOBC episodes. Thus, their conservative nature does not result in a satisfactory average performance, which may limit the practicability of these algorithms.

By tapping into the power of historical data, a natural idea for OOBC is to train an machine learning (ML) based optimizer. Indeed, reinforcement learning has been proposed to solve online allocation problems in other contexts [228, 17, 134]. But, the existing ML-based algorithms for online optimization typically learn online actions in an end-to-end manner without exploiting the structure of the online problem being studied, which hence can have an unnecessarily high learning complexity and create additional challenges for generalization to unseen problem instances [96, 272].

Contribution. We study OOBC with short-term and strict budget constraints, and propose a novel ML-assisted unrolling approach based on recurrent architectures, called

LACC (Learning-Assisted Algorithm Unrolling). Instead of using an end-to-end ML model to directly learn online actions, LACC uniquely exploits the LACC problem structure and unrolls the agent's online decision pipeline into decision pipeline with three stages/layers — update the Lagrangian multiplier, optimize decisions subject to constraints, and update remaining resource budgets — and only plugs an ML model into the first stage (i.e., update the Lagrangian multiplier) where the key bottleneck for better performance exists. Thus, compared with the end-to-end model, LACC benefits generalization by exploiting the knowledge of decision pipeline [96]. Moreover, when the action dimension is larger than the number of constraints (i.e., the dimension of Lagrangian multipliers), the complexity advantage of using LACC to learn Lagrangian multipliers can be further enhanced compared to learning the actions using an end-to-end model.

It is challenging to train LACC through backpropagation since the constrained optimization layer is not easily differentiable. Thus, we derive tractable gradients for back-propagation through the optimization layer based on Karush-Kuhn-Tucker (KKT) conditions. In addition, we rigorously analyze the performance of LACC in terms of the expected cost for both the case when the offline distribution information is available and the case when the data is collected online. Finally, to validate LACC, we present numerical results by considering online resource allocation for maximizing the weighted fairness metric. Our results highlight that LACC can significantly outperform the existing baselines and is very close to the optimal oracle in terms of the fairness utility.

5.2 Related Works

Constrained Online Optimization. Some earlier works [125, 146] solve online optimization with (long-term) constraints by estimating a fixed Lagrangian multiplier using offline data. This approach works only for long-term or average constraints. Many other studies design online algorithms by updating the Lagrangian multiplier in an online style [126, 48, 405, 210]. These algorithms guarantee sub-linear regrets under the i.i.d. context setting, and thus can achieve high utility if the number of time steps is sufficiently large. Likewise, the Lyapunov optimization approach addresses the long-term packing constraints by introducing virtual queues (equivalent to the Lagrangian multiplier) [304, 429, 197]. Nonetheless, it also requires a sufficiently large number of time steps for convergence. By contrast, we consider online optimization with short-term strict budget constraints, which, motivated by practical applications, makes OOBC significantly more challenging.

Our work is relevant to the studies on OOBC [266, 265] which design online algorithms to achieve a worst-case performance guarantee. However, to guarantee the worst-case performance and the feasibility of the algorithm, the algorithms are very conservative and their average performances are unsatisfactory. Comparably, we consider a more general setting where the budget constraint can be nonlinear, and utilize available historical data more efficiently to design ML-based `LACC` that unrolls the online decision pipeline and achieves favorable average performance.

Algorithm Unrolling. `LACC` is related to the recent studies on ML-assisted algorithm unrolling and deep implicit layers, which integrate ML into traditional algorithmic frameworks for better generalization and interpretability, lower sampling complexity and/or

smaller ML model size [6, 96, 227, 300, 272]. Algorithm unrolling has been used for sparse coding [183], signal and image processing [300, 255], and solving inverse problems [226] and ordinary differential equations (ODEs) [94]. Also, algorithm unrolling is applied in *learning to optimize* (L2O) [96, 406]. Among these works, [303] predicts the Lagrangian multiplier by a model to efficiently solve offline optimizations which may have a large number of constraints but allow constraint violations. These studies have their own challenges orthogonal to our problem where the key challenge is the lack of complete offline information. Thus, `LACC`, to our knowledge, is the first to leverage ML to unroll an online optimizer for solving the online convex optimization with budget constraints, thus having better generalization than generic RL-based optimizers to directly obtain end solutions [17, 134, 228].

5.3 Problem Formulation

In this section, we formulate the problem of OOBC and fomally show the challenges of OOBC. As in the existing ML-based optimizers for online problems [228, 17, 134], we consider an agent that interacts with a stochastic environment. The time horizon of an episode consists of N time steps. For an episode, two vectors $\boldsymbol{c} = [c_1, \cdots, c_N]^\top$ and $\boldsymbol{B} = [B_1, \cdots, B_M]^\top$, where c_t is a context vector and $B_m \in \mathbb{R}^+$ is the total budget for resource m, are drawn from a certain joint distribution $(\boldsymbol{c}, \boldsymbol{B}) \sim \mathcal{P}$, which we refer to as the environment distribution. Note that c_i and c_j for $i \neq j$ can follow different probability distributions, and so can B_i and B_j for $i \neq j$. The random vector $\boldsymbol{B} = [B_1, \cdots, B_M]^\top$ are revealed at the beginning of an episode, and represents the budgets for M types of resources. On the other hand, $\boldsymbol{c} = [c_1, \cdots, c_N]^\top$ are online contexts sequentially revealed

over N different steps within an episode. That is, at step t, the agent only knows $c_1, \cdots, c_t$, but *not* the future parameters $c_{t+1}, \cdots, c_N$.

At each step $t = 1, \cdots, N$, the agent makes a decision $x_t \in \mathbb{R}^d$, consumes some budgets, and also receives a utility. Given the decision x_t and parameter c_t, the amount of the resource consumption is denoted as a non-negative function $g_m(x_t, c_t) \geq 0$, for $m = 1, \cdots, M$. To be consistent with the notation of loss function, we use a *cost* or *loss* $l(x_t, c_t)$ to denote the *negative* of the utility — the less $l(x_t, c_t)$, the better. As the cost function $l(\cdot, c_t)$ is parameterized by c_t, knowing c_t is also equivalent to knowing the cost function. We assume that the loss function l and the constraint functions $g_m, m = 1, \cdots, M$ are twice continuously differentiable, and either the loss function l or one of the constraint functions $g_m, m = 1, \cdots, M$ is strongly convex in terms of the decision x_t.

For each episode with $(\boldsymbol{c}, \boldsymbol{B}) \sim \mathcal{P}$, the goal of the agent is minimizing its total cost over the N steps subject to M resource capacity constraints, which we formulate as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}=(x_1,\cdots,x_N)} \sum_{t=1}^{N} l(x_t, c_t), \\ &\text{s.t.} \quad \sum_{t=1}^{N} g_m(x_t, c_t) \leq B_m, m = 1, \cdots, M. \end{aligned} \tag{5.1}$$

This is an online optimization problem with inventory constraints (referred to as OOBC) in the sense that the short-term strict inventory constraints are imposed for each episode of N time steps. An episode has its own M capacity constraints which should be strictly satisfied, and the unused budgets cannot roll over to the next episode. For ease of notation, given a policy π which maps available inputs to *feasible* actions, we denote $L(\pi) = \sum_{t=1}^{N} l(x_t, c_t)$ as the total loss for one episode and $\mathrm{E}[L(\pi)]$ as the expected total loss over the distribution of $(\boldsymbol{c}, \boldsymbol{B}) \sim \mathcal{P}$.

The setting of OOBC presents new technical challenges compared with existing works on constrained online optimization. Specifically in OOBC, the time horizon in an episode (episode length) is finite and can be very short. In this case, there are not many steps for algorithms to converge, and bad decisions at early steps have a large impact on the overall performance. Thus, DMD [48, 405, 210] and Lyapunov optimization [304] which are specifically designed for long episodes may not provide good results for OOBC. Besides, unlike some studies that satisfy average constraints [324, 191] or that only approximately satisfy the constraints under bounded violations (i.e., *soft* constraints) [429, 428, 376], OOBC requires all the constraints in Eqn. (5.1) be strictly satisfied. This requirement is necessary for many practical applications with finite available resources (e.g., a data center's power capacity must not be exceeded [143]), but makes the problem more challenging. Last but not least, the contexts in one episode in OOBC are drawn from a general joint distribution (not necessarily i.i.d.). Under non-i.i.d. cases, DMD[48] has performance guarantees only when each episode is long enough. CR-Pursuit[266, 265] has competitive ratios but is too conservative and may not perform well on average. To improve the average performance of OOBC, new algorithms are needed to effectively utilize history data of previous episodes.

5.4 Learning-Assisted Algorithm Unrolling

In this section, we propose a novel learning-assisted online algorithm, called `LACC`, to solve the OOBC problem.

5.4.1 Relaxed Optimization

The design of `LACC` is based on the Lagrangian relaxed optimization method which is introduced here. Since it is difficult to directly solve the constrained optimization in Eqn. (5.1) due to the lack of complete offline information in an online setting, many studies [35, 48, 304] solve the Lagrangian relaxed form written as follows:

$$\min_{\boldsymbol{x}=(x_1,\cdots,x_N)} \sum_{t=1}^{N} l(x_t, c_t) + \lambda^\top \sum_{t=1}^{N} \boldsymbol{g}(x_t, c_t), \tag{5.2}$$

where $\boldsymbol{g}(x_t, c_t) = [g_1(x_t, c_t), \cdots, g_M(x_t, c_t)]^\top$ and $\lambda = [\lambda_1, \cdots, \lambda_M]^\top$ is the non-negative Lagrangian multiplier corresponding to the M constraints $\sum_{t=1}^{N} \boldsymbol{g}(x_t, c_t) \leq \boldsymbol{B}$. The multiplier λ with M dimensions essentially relaxes the M inventory constraints, thus decoupling the decisions over the N time steps within an episode. It is also interpreted as the resource *price* in the resource allocation literature [315, 71, 304]: A greater λ_t means a higher price for the resource consumption, thus pushing the agent to use less resource. Clearly, had we known the optimal Lagrangian multiplier λ^* at the beginning of each episode, the OOBC problem would become very easy. Unfortunately, knowing λ^* also requires the complete offline information $(\boldsymbol{c}, \boldsymbol{B})$, which is not possible in the online case. Nevertheless, if we can appropriately update λ_t in an online manner while strictly satisfying the constraints, we can also efficiently solve the OOBC problem. Formally, by using λ_t that is updated online for each step t, we can instead solve the following relaxed problem:

$$\min_{x_t} l(x_t, c_t) + \lambda_t^\top \boldsymbol{g}(x_t, c_t), \quad \text{s.t.,} \quad \boldsymbol{g}(x_t, c_t) \leq \boldsymbol{b}_t, \tag{5.3}$$

where $\lambda_t = [\lambda_{t,1}, \cdots, \lambda_{t,M}]^\top$, $\boldsymbol{g}(x_t, c_t) = [g_1(x_t, c_t), \cdots, g_M(x_t, c_t)]^\top$, and the remaining budget for step t is $\boldsymbol{b}_t = \boldsymbol{B} - \sum_{s=1}^{t-1} \boldsymbol{g}(x_s, c_s)$.

In fact, designing good update rules for λ_t for each step $t = 1, \cdots, N$ is commonly considered in the literature [48, 10]. For example, [48] update λ_t by DMD, in order to meet *long*-term constraints while achieving a low regret compared to the optimal oracle. Nonetheless, it requires a large number of time steps to converge to a good Lagrangian parameter. Likewise, the Lyapunov optimization technique introduces a virtual queue, whose length essentially takes the role of λ_t and is updated as $\lambda_{t+1} = \max\left\{\lambda_t + \boldsymbol{g}(x_t, c_t) - \frac{1}{N}\boldsymbol{B}, 0\right\}$ or in other similar ways [304]. Nonetheless, the convergence rate of using Lyapunov optimization is slow (even assuming c_t is i.i.d. for $t = 1, \cdots, N$), and there exists a tradeoff between cost minimization and long-term constraint satisfactory, making it unsuitable for the short-term constraints that we focus on.

Alternatively, one may want to exploit the distribution information of $(\boldsymbol{c}, \boldsymbol{B}) \sim \mathcal{P}$ and solve a relaxed problem offline by considering M average constraints (referred to as AVG-LT). That is, we replace the short-term capacity constraints in Eqn. (5.1) with $\mathrm{E}_{\mathcal{P}}\left[\sum_{t=1}^{N} g_m(x_t, c_t)\right] \leq B_m$ for $m = 1, \cdots, M$. By solving this relaxed problem, we can obtain a Lagrangian multiplier $\lambda_{\mathcal{P}}$ that only depends on $\mathcal{P}$ but not the specific $(\boldsymbol{c}, \boldsymbol{B})$. Thus, we can replace λ_t in Eqn. (5.3) with $\lambda_{\mathcal{P}}$. However, since this method uses a constant Lagrangian multiplier for all episodes, we will either be overly conservative and not using the budgets as much as possible, or violating the the constraints.

5.4.2 Algorithm Unrolling

We propose to leverage the powerful capacity of ML to find a solution. One approach is to train an end-to-end model that takes the online input information and

Algorithm 6 Online Inference Procedure of `LACC`

Require: ML model f_θ.

1: **for** t=1 to N **do**

2: Receive c_t, forward propagate f_θ and get Lagrangian multiplier $\lambda_t = f_\theta\left(\boldsymbol{b}_t, c_t, \bar{t}\right)$.

3: Solve the constrained convex optimization in (5.4) and make action x_t.

4: Update the resource budget $\boldsymbol{b}_{t+1} = \boldsymbol{b}_t - \boldsymbol{g}(x_t, c_t)$.

5: **end for**

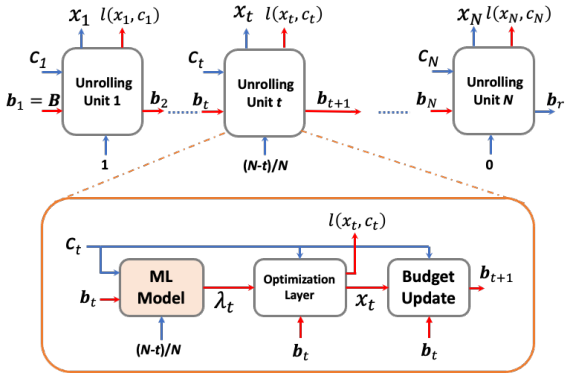


Figure 5.1: Architecture of `LACC`.

directly outputs a decision. But, the end-to-end model should be large enough to capture the possibly complex logic of the optimal policy, and the end-to-end models often have poor interpretability and worse generalization (see the comparison between `LACC` and the generic end-to-end approach in Section 5.7). Therefore, instead of replacing the whole decision pipeline with ML, *we only plug an ML model in the most challenging stage* — online updating of the Lagrangian multiplier λ_t needed to solve the relaxed problem in Eqn. (5.3).

As shown in Algorithm 6 and illustrated in Fig. 5.1, the decision pipeline at step t can be decomposed into three stages as follows.

Updating λ_t. At the beginning of step $t = 1, \cdots, N$, the ML model takes the parameter c_t, the remaining budget $\boldsymbol{b}_t = [b_{t,1}, \cdots, b_{t,m}]^\top$ and the normalized number of remaining steps $\bar{t} = \frac{N-t}{N}$ as the inputs, and outputs the Lagrangian multiplier $\lambda_t = [\lambda_{t,1}, \cdots, \lambda_{t,m}]^\top$. Letting f_θ denote the ML model parameterized by θ, we have $\lambda_t = f_\theta(\boldsymbol{b}_t, c_t, \bar{t})$.

Optimization Layer. In the optimization layer, we solve a relaxed convex problem formulated in Eqn. (5.3). The natural constraints on the remaining resource budgets ensure that the *strict* inventory constraints are always satisfied by `LACC`. We denote the optimization layer as $p(c_t, \lambda_t, \boldsymbol{b}_t)$ and thus have:

$$\begin{aligned} x_t = p(c_t, \lambda_t, \boldsymbol{b}_t) = &\operatorname*{argmin}_x \Big(l(x, c_t) + \lambda_t^\top \boldsymbol{g}(x, c_t) \Big), \\ &\text{s.t., } \boldsymbol{g}(x, c_t) \leq \boldsymbol{b}_t. \end{aligned} \tag{5.4}$$

Updating Resource Budgets. In the last stage, the remaining resource budgets serve as an input for the next recurrence and are updated as $\boldsymbol{b}_{t+1} = \boldsymbol{B} - \sum_{s=1}^{t} \boldsymbol{g}(x_s, c_s) = \boldsymbol{b}_t - \boldsymbol{g}(x_t, c_t)$.

Each episode includes N recurrences, each for one decision step. The cost for step t is calculated after the optimization layer as $l(x_t, c_t)$. Note that in the N-th recurrence which is the final stopping step, the remaining budget $\boldsymbol{b}_r = \boldsymbol{b}_N - \boldsymbol{g}(x_N, c_N)$ will be wasted if not used up. Thus, we directly set $\lambda_N = 0$ for the N-th unrolling unit.

5.5 Training the Unrolling Architecture

In this section, we first consider the offline training where offline data is available. Then we extend to the online training setting where the data is collected online.

5.5.1 Offline Training

Training Objective

For the ease of notation, we denote the online optimizer as $\boldsymbol{h}_\theta(\boldsymbol{B}, \boldsymbol{c})$. For offline training, we are given an *unlabeled* training dataset $S = \{(\boldsymbol{c}_1, \boldsymbol{B}_1), \cdots, (\boldsymbol{c}_n, \boldsymbol{B}_n)\}$, with n samples of $(\boldsymbol{c}, \boldsymbol{B})$. The training dataset can be synthetically generated by sampling from the target distribution for the online input $(\boldsymbol{c}, \boldsymbol{B})$, which is a standard technique in the context of *learning to optimize* [96, 117, 238, 134]. By forward propagation, we can get the empirical training loss as $L(\boldsymbol{h}_\theta, S) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{N} l(\boldsymbol{x}_{i,t}, \boldsymbol{c}_{i,t})$ where $\boldsymbol{x}_{i,t}$ is the output of the online optimizer $\boldsymbol{h}_\theta$ regarding $\boldsymbol{c}_i$ and $\boldsymbol{B}_i$. By minimizing the empirical loss, we get $\hat{\theta} = \arg\min_\theta L(\boldsymbol{h}_\theta, S)$.

Backpropagation

Typically, the minimization of the training loss is performed by gradient descent-based algorithms like SGD or Adam, which need back propagation to get the gradient of the loss with respect to the ML model weight θ. Nonetheless, unlike standard ML training (e.g., neural network training with only linear and activation operations), our unrolled recurrent architecture includes an *implicit* layer — the optimization layer [227]. Additionally, the unrolling architecture has multiple skip connections. Thus, the back-propagation process is dramatically different from that of standard recurrent neural networks. Next, we derive the gradients for back propagation in our unrolling design. Note that the loss $l(x_t, c_t)$ for any $t = 1, \cdots, N$ is directly determined by the output of the optimization layer x_t and the

parameter c_t, and x_t needs back propagation. Thus, by the chain rule, we have

$$\bigtriangledown_\theta l(x_t, c_t) = \bigtriangledown_{x_t} l(x_t, c_t)(\bigtriangledown_{\lambda_t} x_t \bigtriangledown_\theta \lambda_t + \bigtriangledown_{\boldsymbol{b}_t} x_t \bigtriangledown_\theta \boldsymbol{b}_t)\,. \tag{5.5}$$

To get $\bigtriangledown_{\lambda_t} x_t$ and $\bigtriangledown_{\boldsymbol{b}_t} x_t$ in Eqn. (5.5), we need to perform back propagation for the optimization layer $p(c_t, \lambda_t, \boldsymbol{b}_t)$. This is a challenging task and will be addressed in Section 5.5.1. The other gradients in Eqn. (5.5) include $\bigtriangledown_\theta \lambda_t$ and $\bigtriangledown_\theta \boldsymbol{b}_t$. Note that λ_t, which is the ML model output directly determined by its ML model weight θ, and the remaining budget $\boldsymbol{b}_t$ both need back propagation. Thus, the gradient of λ_t with respect to the ML model weight θ is expressed as

$$\bigtriangledown_\theta \lambda_t = \bigtriangledown_\theta f_\theta\left(\boldsymbol{b}_t, c_t, \bar{t}\right) + \bigtriangledown_{\boldsymbol{b}_t} f_\theta\left(\boldsymbol{b}_t, c_t, \bar{t}\right) \bigtriangledown_\theta \boldsymbol{b}_t, \tag{5.6}$$

Now, it remains to derive $\bigtriangledown_\theta \boldsymbol{b}_t$, which is important since b_t is the signal connecting two adjacent recurrences. By the expression of b_t in Line 4 of Algorithm 6, we have

$$\bigtriangledown_\theta \boldsymbol{b}_t = \bigtriangledown_\theta \boldsymbol{b}_{t-1} + \bigtriangledown_{x_{t-1}} \boldsymbol{g}(x_{t-1}, c_{t-1}) \bigtriangledown_{\lambda_{t-1}} x_{t-1} \bigtriangledown_\theta \lambda_{t-1}, \tag{5.7}$$

Combining Eqn. (5.5), (5.6) and (5.7), we get the recurrent expression for back propagation. Then, by adding up the gradients of the losses over N time steps, we get the gradient of the total loss as $\bigtriangledown_\theta L(\boldsymbol{h}_\theta, S) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{N} \bigtriangledown_\theta l\left(x_{i,t}, c_{i,t}\right)$.

Differentiating the Optimization Layer

It is challenging to get the close-form solution and its gradients for many constrained optimization problems. One possible remedy is to use some black-box gradient estimators like zero-order optimization [273, 337]. However, zero-order gradient estimators are not computationally efficient since many samples are needed to estimate a gradient. Another

method is to train a deep neural network to approximate the optimization layer in Eqn. (5.4) and then calculate the gradients based on the neural network. However, we need many samples to pre-train the neural network, and the gradient estimation error can be large. To address these challenges, we analytically differentiate the solution to Eqn. (5.4) in the optimization layer with respect to the inputs λ_t, and $\boldsymbol{b}_t$ by exploiting KKT conditions [227, 71]. The KKT-based differentiation method, given in Proposition 23, is computationally efficient, explainable and accurate (under mild technical conditions).

Proposition 23 (Back-propagation by KKT)**.** *Assume that x_t and μ_t are the primal and dual solutions to Eqn.* (5.4) *, respectively. Let* $\Delta_{11} = \bigtriangledown_{x_t x_t} l\left(x_t, c_t\right) + \sum_{m=1}^{M}\left(\lambda_{m,t} + \mu_{m,t}\right) \bigtriangledown_{x_t x_t} g_m(x_t, c_t)$, $\Delta_{12} = \left[\bigtriangledown_{x_t} \boldsymbol{g}\left(x_t, c_t\right)\right]^{\top}$, $\Delta_{21} = \operatorname{diag}(\mu_t) \bigtriangledown_{x_t} \boldsymbol{g}\left(x_t, c_t\right)$, *and* $\Delta_{22} = \operatorname{diag}\left(\boldsymbol{g}\left(x_t, c_t\right) - \boldsymbol{B}_t\right)$. *If the conditions in Proposition 24 are satisfied, the gradients of the optimization layer w.r.t. λ_t and $\boldsymbol{b}_t$ are*

$$\bigtriangledown_{\lambda_t} x_t = -\left(\Delta_{11}^{-1} + \Delta_{11}^{-1} \Delta_{12} \operatorname{Sc}\left(\Delta, \Delta_{11}\right)^{-1} \Delta_{21} \Delta_{11}^{-1}\right) \Delta_{12},$$

$$\bigtriangledown_{\boldsymbol{b}_t} x_t = -\Delta_{11}^{-1} \Delta_{12} \operatorname{Sc}\left(\Delta, \Delta_{11}\right)^{-1} \operatorname{diag}(\mu_t),$$

where $\operatorname{Sc}\left(\Delta, \Delta_{11}\right) = \Delta_{22} - \Delta_{21} \Delta_{11}^{-1} \Delta_{12}$ *denotes the Shur-complement of* Δ_{11} *in* $\Delta = [[\Delta_{11}, \Delta_{12}]; [\Delta_{21}, \Delta_{22}]]$. □

We find that to get truly accurate gradient computation by Proposition 24, the Shur-complement $\operatorname{Sc}\left(\Delta, \Delta_{11}\right)$ and Δ_{11} should be invertible. Otherwise, we can get approximated gradients by taking pseudo-inverse of $\operatorname{Sc}\left(\Delta, \Delta_{11}\right)$ and Δ_{11}. The sufficient conditions to guarantee perfectly accurate gradient computation are given in Proposition 24.

Proposition 24 (Sufficient Conditions of Accurate Differentiation)**.** *Assume that the problem in Eqn.* (5.4) *satisfies strong duality. The loss l or one of the constraints $g_m, m = 1, \cdots, M$ is strongly convex with respect to x. Denote $\mathcal{A}$ as the index set of constraints that are activated (i.e., equality holds) under the optimal solution. If $\mu_{m,t} \neq 0, \forall m \in \mathcal{A}$, the size of the activation set satisfies $|\mathcal{A}| \leq d$ with d as the action dimension, and the gradients $\bigtriangledown_{x_t} g_m(x_t), m \in \mathcal{A}$ are linearly independent and not zero vectors, the gradients in Proposition 23 are perfectly accurate.*

Remark 25. *To derive the gradient of Eqn.* (5.4) *with respect to an input parameter, we take gradients on both sides of the equations in KKT conditions by the chain rule and get new equations about the gradients. By solving the obtained set of equations and exploiting the block matrix inversion, we can derive the gradients with respect to the inputs in Proposition 23.*

The conditions in Proposition 24 are mild in practice. First, strong duality is easily satisfied for the considered convex optimization in Eqn. (5.4) *given the Slater's condition [71]. Besides, the requirement of strong convexity excludes linear programming (LP). Actually, LP problems with resource constraints are usually solved by other relaxations other than our considered relaxation in Eqn.* (5.2) *[126]. The other conditions are related to activated constraints. According to the condition of complementary slackness [71], the condition that optimal dual variables corresponding to the activated constraints are not zero typically holds. We also require that the number of activated constraints is less than the action dimension, and the gradient vectors of the activated constraint functions under optimal solutions should be independent from each other. Given that at most a small number of constraints are activated in most cases, the two conditions are easily satisfied. Actually, the independence*

condition requires that the activated constraints are not redundant — an activated constraint function is not a linear combination of any other activated constraint functions; otherwise, it can be replaced by other constraints.

5.5.2 Online Training

Algorithm 7 Online Training of `LACC`

1: **Initialization:** The weight $\hat{\theta}_0$ of the unrolling model, step size $\bar{\alpha}$.

2: **while** a new instance i arrives **do**

3: Perform N-step online inference by Algorithm 6.

4: Collect context $\boldsymbol{c}_i$ and budget $\boldsymbol{B}_i$ of instance i and update the model by

$$\hat{\theta}_i = \hat{\theta}_{i-1} - \bar{\alpha} \nabla_\theta L(h_{\hat{\theta}_{i-1}}, \boldsymbol{c}_i),$$

where $\nabla_\theta L(h_{\hat{\theta}_{i-1}}, \boldsymbol{c}_i)$ is obtained by back propagation in Eqn. (5.5).

5: **end while**

In practice, we may have a cold-start setting without many offline samples. An efficient approach for this setting is online stochastic gradient descent (SGD). Concretely, when the i-th instance arrives, we perform online inference by Algorithm 6. After the instance with N steps ends, we collect the context and budget data of this episode and update the ML model weight $\hat{\theta}_i$ by performing one-step gradient descent, i,e, $\hat{\theta}_i = \hat{\theta}_{i-1} - \bar{\alpha} \nabla_\theta L(h_{\hat{\theta}_{i-1}}, \boldsymbol{c}_i)$ where $L(h_{\hat{\theta}_{i-1}}, \boldsymbol{c}_i)$ is the loss of the unrolling model for the ith instance and $\bar{\alpha}$ is the stepsize. Then, with the updated $\hat{\theta}_i$, we perform inference by Algorithm 6 for the instance in the $(i+1)$-th round. We will show by analysis that the average cost decreases with time.

5.6 Performance Analysis

In this section, we bound the expected cost when the trained ML model f_θ is used in LACC.

Definition 26. *The weight in the ML model f_θ (and also the online optimizer $\boldsymbol{h}_\theta$) that minimizes the expected loss $\mathrm{E}\left[L(\boldsymbol{h}_\theta, \boldsymbol{c})\right]$ with respect to the distribution of $(\boldsymbol{c}, \boldsymbol{B}) \sim \mathcal{P}$ is defined as $\theta^* = \arg\min_{\theta\in\Theta} \mathrm{E}\left[L(\boldsymbol{h}_\theta, \boldsymbol{c})\right]$, and the weight that minimizes the empirical loss $L(\boldsymbol{h}_\theta, S)$ is defined as $\hat{\theta}^* = \arg\min_{\theta\in\Theta} L(\boldsymbol{h}_\theta, S)$, where Θ is the weight space.*

In Definition 26, given the weight space Θ, $\boldsymbol{h}_{\theta^*}$ is the best online optimizer based on the unrolling architecture in terms of the expected cost. $\boldsymbol{h}_{\theta^*}$ is not the offline-optimal policy, but it is close to the policy that performs best given available online information when the capacity of the ML model and weight space Θ are large enough. Next, we show the performance gap of LACC compared with h_{θ^*}.

Theorem 5.6.1. *By the optimization layer in Eqn.* (5.4)*, LACC satisfies the inventory constraints for each OOBC instance. Suppose that $\hat{\theta}$ is the ML model weight by offline training on dataset S with n samples, and that we plug it into the online optimizer $\boldsymbol{h}_{\hat{\theta}}$. With probability at least $1-\delta, \delta \in (0,1)$,*

$$\begin{aligned} \mathrm{E}\left[L(\boldsymbol{h}_{\hat{\theta}})\right] - \mathrm{E}\left[L(\boldsymbol{h}_{\theta^*})\right] \leq \mathcal{E}\left(\boldsymbol{h}_{\hat{\theta}}, S\right) + 4R_n(L\circ\mathbb{H}) \\ +2\left(\Gamma_{L,c}\omega_c + \Gamma_{L,b}\omega_b\right)\sqrt{\frac{\ln(2/\delta)}{n}}, \end{aligned} \tag{5.8}$$

where $\mathcal{E}\left(\boldsymbol{h}_{\hat{\theta}}, S\right) = L(\boldsymbol{h}_{\hat{\theta}}, S) - L(\boldsymbol{h}_{\hat{\theta}^*}, S)$ is the training error, $R_n(L\circ\mathbb{H})$ is the Rademacher complexity regarding the loss space $L\circ\mathbb{H} = \{L(\boldsymbol{h}), \boldsymbol{h}\in\mathbb{H}\}$ with $\mathbb{H}$ being the ML model set, $\omega_c = \max_{\boldsymbol{c},\boldsymbol{c}'\in\mathbb{C}} \|\boldsymbol{c}-\boldsymbol{c}'\|$ is the size of the parameter space $\mathbb{C}$, $\omega_b = \max_{\boldsymbol{B},\boldsymbol{B}'\in\mathbb{B}} \|\boldsymbol{B}-\boldsymbol{B}'\|$ is

the size of the capacity constraint space $\mathbb{B}$, $\Gamma_{L,c}$ *and* $\Gamma_{L,b}$ *are the Lipschitz constants of the total loss* $L(\boldsymbol{h}_\theta, \boldsymbol{c}) = \sum_{t=1}^{N} l(x_t, c_t)$ *with respect to* $\boldsymbol{c}$ *and* $\boldsymbol{B}$*, respectively.*

Proposition 27. *If a linear model* $f_\theta(\boldsymbol{v}) = \theta^\top \phi(\boldsymbol{v}), \|\theta\| \leq Z$ *is used as the ML model in* `LACC`*, the Rademacher complexity* $R_n(L \circ \mathbb{H})$ *is bounded ny* $O\left(\frac{ZW}{\sqrt{n}}\right)$*, where* $W = \sup_{\boldsymbol{v}} \sqrt{\phi(\boldsymbol{v})^\top \phi(\boldsymbol{v})}$*. If a neural network, where the depth is* K*, the width is less than* u*, activation functions are* Γ_α*-Lipschitz continuous, and the spectrum norm of the weight matrix in layer* k *with is less than* Z_k*, is used as the ML model, the Rademacher complexity* $R_n(L \circ \mathbb{H})$ *is bounded by* $R_n(L \circ \mathbb{H}) \leq O\left(\frac{K^{3/2} u \Gamma_\alpha (\beta_b + \beta_c) \prod_{k=1}^{K} Z_k}{\sqrt{n}}\right)$*, where* β_b, β_c *are the largest* l_2*-norm of* $\boldsymbol{B}$ *and* $\boldsymbol{c}$*. The notation* O *in this proposition indicates the scaling relying on* M*,* N*, the Lipschitz constants of loss function* l*, constraint function* g*, optimization layer* p *and neural network* f*.*

Remark 28. *Theorem 5.6.1 shows that the performance gap between* `LACC` *and the pseudo-oracle* h_{θ^*} *in terms of expected loss is bounded by the empirical training error, plus a generalization error which relies on the Radmacher complexity, the LipSchitz constant of the online optimizer, and the number of training samples. The Radmacher complexity indicates the richness of the loss function space with respect to the online optimizer space* $\mathbb{H}$ *and the distribution* $\mathcal{P}$ *and is further bounded in Proposition 27. From the bound of Radmacher complexity, we find that for both linear model and neural network, the generalization error increases with episode length* N *and the number of constraints* M*. Besides, the Rademacher complexity relies on the ML model designs. For example, if a linear model is used as the ML model, the generalization error relies on the norm bounds of feature mapping and linear weights, while if a neural network is used as the ML model, the generalization error is related*

to the network length , width, the smoothness of activation functions, and spectral norm bounds of the weights in each layer. The last term of the expected cost bound is scaled by $(\Gamma_{L,c}\omega_c + \Gamma_{L,b}\omega_b)$ *which indicates the sensitivity of the loss when the inputs are changed. This term highly depends on the Lipschitz constants of the total loss regarding the two inputs.*

□

Proposition 29 (Average Cost of Online Training)**.** *Assume that for each round* i, $\nabla_\theta \mathrm{E}[L(h_{\hat{\theta}_i})]$ *is* $\Gamma_{\nabla L,\theta}$*-Lipschitz continuous, and the Polyak-Lojasiewicz inequality is satisfied, i.e.* $\exists \varsigma > 0, \left\|\nabla_\theta \mathrm{E}[L(h_{\hat{\theta}_i})]\right\|^2 \geq 2\varsigma \left(\mathrm{E}[L(h_{\hat{\theta}_i})] - \mathrm{E}[L(h_{\theta^*})]\right)$*. Also, assume that for the distribution of* $\boldsymbol{c}_i$*,,* $\exists \iota_G > \iota > 0$ *such that* $\forall i$, $\left\langle \nabla_\theta \mathrm{E}[L(h_{\hat{\theta}_i})], \mathrm{E}\left[\nabla_\theta L(h_{\hat{\theta}_i}, \boldsymbol{c}_i)\right]\right\rangle \geq \iota \left\|\nabla_\theta \mathrm{E}[L(h_{\hat{\theta}_i})]\right\|_2^2$, $\|\mathrm{E}[\nabla_\theta L(h_{\hat{\theta}_i}, \boldsymbol{c}_i)]\|_2 \leq \iota_G \left\|\nabla_\theta \mathrm{E}[L(h_{\hat{\theta}_i})]\right\|_2$*, and there exist* $\varpi, \varpi_V > 0$ *such that* $\forall i$, $\mathrm{Var}\left[\nabla_\theta L(h_{\hat{\theta}_i}, \boldsymbol{c}_i)\right] \leq \varpi + \varpi_V \left\|\nabla_\theta \mathrm{E}[L(h_{\hat{\theta}_i})]\right\|_2^2$*. Then with the same notations as Theorem 5.6.1, for the online setting where* `LACC` *is trained by SGD with stepsize* $0 < \bar{\alpha} \leq \frac{\iota}{\Gamma_{\nabla L,\theta}(\varpi_V + \iota_G^2)}$*, with probability at least* $1 - \delta, \delta \in (0,1)$*, we have for each round* i

$$\mathrm{E}\Big[L(\boldsymbol{h}_{\hat{\theta}_i}) - L(\boldsymbol{h}_{\theta^*})\Big] \leq \frac{\bar{\alpha}\Gamma_{\nabla L,\theta}\varpi}{2\iota\varsigma} + O\left((1 - \bar{\alpha}\iota\varsigma)^i\right), \tag{5.9}$$

where the expectation is taken over the randomness of context $\boldsymbol{c}$ *and budget* $\boldsymbol{B}$ *and the model weight* $\hat{\theta}_i$ *by SGD.*

Proposition 29 bounds the expected loss gap between the learned weight $\hat{\theta}_i$ by SGD and the optimal weight θ^* in Definition 26. The first non-reducible term is caused by the randomness of the context and budget $\{(\boldsymbol{c}_i, \boldsymbol{B}_i)\}$. The second term decreases with time, and the convergence rate depends on the sequence randomness and the parameter ς in the Polyak-Lojasiewicz inequality.

5.7 Numerical Results

Weighted fairness is a classic performance metric in the resource allocation literature [231], including fair allocation in computer systems [165] and economics [200]. Here, we consider a general *online* setting. A total of N jobs arrive sequentially, and job t has a weight $c_t \geq 0$. The agent allocates resource $x_t \geq 0$ to job t at each step t. We consider the commonly-used weighted fairness $\sum_{t=1}^{N} c_t \log(x_t)$ [231]. We create the training and testing samples based on the Azure cloud workload dataset, which contains the average CPU reading for tasks at each step [348].

We consider several baseline algorithms as follows. The Offline Optimal Oracle **OPT** is the solution to the problem in Eqn. (5.1). We consider two heuristics: One is Equal Resource Allocation (**Equal**) which equally allocates the total resource capacity to N jobs, and another one is Resource Allocation with Average Long-term Constraints (**AVG-LT**) which relaxes the inventory constraints of the weighted fairness problem as $\mathrm{E}_{\mathcal{P}}\left[\sum_{t=1}^{N} x_t\right] \leq B$ and uses the optimal Lagrangian multiplier for this relaxed problem as λ_t for online allocation. We consider two algorithms based on dual mirror descent which are Dual Gradient Descent (**DGD**) and Multiplicative Weight (**MW**) [48]. To reduce the resource waste after the last step, we slightly revise DGD and MW by setting the allocation decision for job N as $\min\left(b_N, x_{\max}\right)$. CR-pursuit (**CR-pursuit**) is the state-of-the-art online algorithm that makes online actions by tracking a pseudo-optimal algorithm with a competitive guarantee [266, 265]. We also compare `LACC` with the end-to-end Reinforcement Learning (**RL**). A neural network with the same size of the ML model as in `LACC` is used in

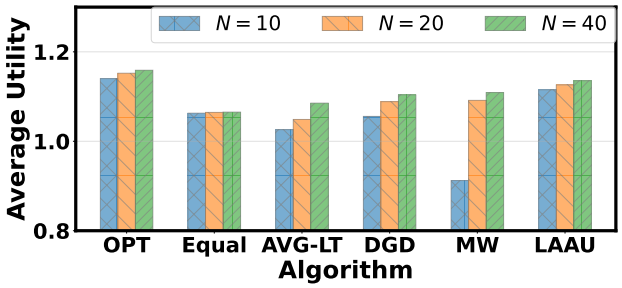


Figure 5.2: Average utility with different episode lengths

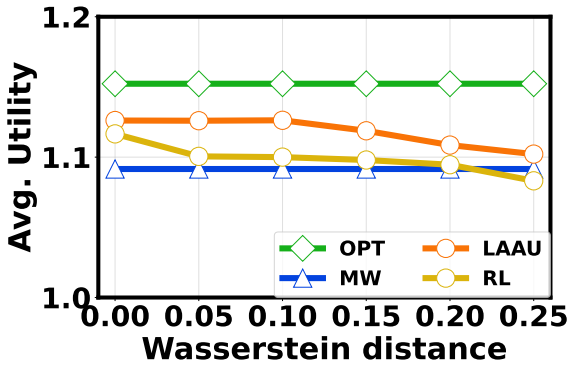


Figure 5.3: Average utility with different Wasserstein distances.

RL to directly predict the solution x_t, given parameter c_t and budget b_t as inputs.

Average Utility. We first show in Fig. 5.2 the average utilities (per time step). We do not add the average utility of CR-pursuit in the figure because its average utility for our evaluation instances is as low as 0.562, exceeding the utility range of the figure. Clearly, OPT achieves the highest utility, but it is infeasible in practice due to the lack of complete offline information. We can observe that the average utilities by expert algorithms including AVG-LT, DGD, MW are even below the average utility by the simple equal allocation (Equal) when the episode length is $N = 10$. This is because all the three algorithms are designed for online optimizations with long-term constraints, and not suitable for the more

challenging short-term counterparts. By contrast, LACC performs well for all cases with large and small episode lengths, and outperforms the other algorithms designed for long-term constraints even when N is as large as 40. This demonstrates the power of LACC in solving challenging online problems with inventory constraints.

OOD Testing. In practice, the training-testing distributional discrepancy is common and decreases ML model performance. We measure the training-testing distributional difference by the Wasserstein distance d_W. We choose the setting with episode length $N = 20$ to perform the OOD evaluation. To create the distributional discrepancy, we add i.i.d. Gaussian noise with different means and variances to the training data and keep the testing data the same as the default setting. The offline optimal and MW do not make use of the training distribution, and hence are not affected. We can see in Fig. 5.3 that, the OOD testing decreases the performance of both LACC and RL, but LACC is less affected by OOD testing than RL and is still higher than that of the baseline MW even under large Wasserstein distance. This is because the unrolling architecture in LACC has an optimization layer and a budget update layer, which are deterministic and have no training parameters, so only the ML model to learn the Lagrangian multiplier is affected by OOD testing. Comparably, RL uses an parameterized end-to-end policy model trained on the offline data, so it has worse performance under OOD testing. This highlights that by the unrolling archietecture, LACC has better generalization performance than end-to-end models.

5.8 Conclusion

In this chapter, we focus on OOBC and propose a novel ML-assisted unrolling approach based on the online decision pipeline, called `LACC`. The key novelty of `LACC` is that it leverages an ML model for updating the Lagrangian multiplier online. We derive the gradients for back-propagation and perform rigorous analysis on the expected cost. Finally, we present numerical results on weighted fairness and highlight `LACC` significantly outperforms the existing baselines in terms of the average performance.

Chapter 6

Theoretical Understanding of Domain Knowledge Informed Learning

6.1 Introduction

The remarkable success of deep neural networks (DNNs), or more generally machine learning, largely relies on the proliferation of data samples with ground-truth labels for supervised learning. Nonetheless, labeled data of high quality can often be very limited and/or extremely expensive to collect in real application domains, including medical sciences, security-related fields, and specialized engineering areas [396].

In parallel with the data-driven learning paradigm, domain *knowledge* (which we simply refer to as knowledge) has been utilized to assist with decision making and system

designs, with a long history of success. As its name would suggest, domain knowledge is naturally domain-specific and can come from various sources in multiple forms, such as subjective experiences (e.g., medical prognosis), external sources, and scientific laws. For example, partial differential equations are used to govern many flow dynamics in physics, and the Shannon channel capacity is the fundamental principle to guide the design of modern communications systems [175, 408].

Importantly, domain knowledge has already been, sometimes implicitly, integrated into every stage of the machine learning pipeline, including training data augmentation, hypothesis set selection, model training and hypothesis finalization. For example, differential equations and logic rules from physical sciences and/or common knowledge provide additional constraints or new functional regularization terms for model training [55, 67, 363, 302, 417].

Despite the numerous successful examples [396, 123], there still lacks a rigorous understanding of the role of domain knowledge in informed learning. In this chapter, we focus on informed DNNs — DNNs with domain knowledge explicitly integrated into the training risk/loss function. Concretely, we consider an over-parameterized DNN with a sufficiently large network width [306], and study how domain knowledge affects the DNN from three complementary aspects: convergence, generalization, and sampling complexity, which is summarized as below.

Convergence (Theorem 6.4.1): We show the convergence of training an informed risk function under milder technical assumptions than the prior works (Section 6.4.1). More specifically, we show that for inputs within a smooth set (Definition 31), the network outputs converge to the optimal solution jointly determined by all the samples in the set.

Generalization (Theorems 6.4.2 and 6.5.1): We show in Theorem 6.4.2 that the population risk relies on the knowledge imperfectness (Definition 38) as well as knowledge-regularized label imperfectness (Definition 39). Specifically, knowledge has two benefits: regularization for noisy labels and supplementing labels. We propose a generalized informed risk function which disentangles the two effects by introducing another hyper-weight β, followed by the population risk bounds in Theorem 6.5.1 and Corollary 41.

Sampling Complexity (Corollay 43): By establishing a quantitative equivalence between domain knowledge and labeled samples, we show that domain knowledge (with a reasonable quality) can effectively reduce the number of labeled samples while achieving the same generalization performance, compared to the no-knowledge case.

6.2 Related Work

Informed Machine Learning. The broad paradigm of informed machine learning [396] includes several existing learning frameworks, such as learning using privileged information (LUPI) [391] where side knowledge is available for labeled samples [391, 301, 352]. Likewise, knowledge distillation [327, 194, 181, 107] transfers prior knowledge from teacher networks to a student network. Some recent studies have also focused on understanding knowledge distillation [13]. In [319], a generalization bound is derived for knowledge distillation based on linear classifiers and deep linear classifiers, providing insights towards the mechanism of knowledge distillation. The subsequent analysis [207, 327] extends to neural networks, showing that the student network may generalize better by exploiting soft labels from the teacher model. Teacher imperfectness is investigated in [119], which

bounds the learning error and proposes enhanced methods to address imperfect teachers. Physics-informed neural networks (PINNs) have been recently proposed to solve partial differential equations (PDEs) [187, 203, 330, 45, 123, 408]. Besides empirical studies, [359] bounds the expected PINN loss, showing that the minimizer of the regularized loss converges to the PDE solution.

More broadly, informed machine learning also includes weakly-supervised learning [445, 336] and few-shot learning [400], where knowledge provides weak supervision. Domain-specific constraints [302] and semantic information [417, 128] can also be viewed as knowledge injected into training. Our work complements these empirical studies and provides a rigorous understanding of knowledge in a unified framework.

Over-Parameterized Neural Networks. Several recent studies [44, 366, 30, 29, 204, 234, 418, 16, 34, 33, 81, 14, 306] show that over-parameterized neural networks have good convergence and generalization performance. In addition to assuming data separability in a strong sense, another crucial assumption often made in the existing studies is that the network widths increase polynomially with the total number of training samples. In informed DNNs, however, we can have many (unlabeled) training samples fed into the knowledge risk, which hence may not satisfy these assumptions. Thus, we analyze knowledge-informed over-parameterized neural networks under relaxed assumptions (Section 6.4).

Regularization. In the broad context of regularization, [404] shows that over-parameterized neural networks with l_2-regularization can achieve a larger margin and thus better generalization, [63] proves that SGD with label noise is equivalent to an implicit regularization term, while [403] shows that the drop-out operation for neural networks has

both explicit and implicit regularization effects. These regularizers are usually imposed on the network weights, whereas the knowledge-based regularizer in informed machine learning also incorporates inputs and directly regularizes the network output.

6.3 Informed Neural Network

Notations: We use the expression $[L]$ to denote the set $\{1, 2, \cdots, L\}$ for a positive integer L. Denote the indicator function as $\mathbb{1}(x) = 1$ if $x > 0$, and $\mathbb{1}(x) = 0$ otherwise. $\mathbb{E}$ is the expectation operator and $\mathbb{P}$ is a probability measure. $\mathbb{R}^d$ is d-dimensional real number space. $\mathcal{N}(x, \sigma^2)$ is the Gaussian distribution with mean x and variance σ^2. Denote $|\mathcal{A}|$ as the size of a set $\mathcal{A}$. For a vector x, $\|x\|$ is l_2-norm and $[x]_j$ is the jth entry. For a matrix $\boldsymbol{X}$, $\|\boldsymbol{X}\|_2$ represents the spectral norm, and $\|\boldsymbol{X}\|$ is the Frobenius norm. $\mathcal{B}(x, \tau) = \{y \mid \|x - y\| \leq \tau\}$ is the neighborhood domain.

6.3.1 Preliminaries of Neural Networks

Consider a supervised learning task to learn a relationship mapping the input $x \in \mathcal{X} \subseteq \mathbb{R}^b$ to its output $y \in \mathcal{Y} \subseteq \mathbb{R}^d$. The pair of input and output (x, y) follows a joint distribution $\mathbb{P}_{XY}$. More concretely, we consider a fully-connected DNN with an input layer, $L \geq 1$ hidden layers, and an output layer. Each hidden layer has m neurons, followed by ReLu activation denoted as $\sigma(\cdot)$. Denote $\boldsymbol{W}_0 \in \mathbb{R}^{b \times m}$ as the weights for the input layer, $\boldsymbol{W}_l \in \mathbb{R}^{m \times m}$ as the weights for the l-th layer for $l \in [L]$, and $\boldsymbol{V} \in \mathbb{R}^{d \times m}$ as the weights for the output layer. We denote the output of the l-th layer as $h_l = \sigma(\boldsymbol{W}_l h_{l-1})$, for $l \in [L]$, where h_0 is the input x. The output of the neural network can be expressed as $h_{\boldsymbol{W}} = \boldsymbol{V} h_L$,

where $\boldsymbol{W} = \{\boldsymbol{W}_0, \boldsymbol{W}_1, \cdots, \boldsymbol{W}_L\}$. Thus, the DNN can be expressed as

$$h_{\boldsymbol{W}}(x) = \boldsymbol{V}\sigma\left(\boldsymbol{W}_L\sigma(\boldsymbol{W}_{L-1}\cdots\sigma\left(\boldsymbol{W}_1\sigma(\boldsymbol{W}_0 x)\right))\right). \tag{6.1}$$

Given a DNN $h_{\boldsymbol{W}}$, the risk for a labeled sample (x, y) is denoted as $r\left(h_{\boldsymbol{W}}(x), y\right)$. The goal of the learning task is to learn a DNN that minimizes the population risk:

$$R(h) = \mathbb{E}\left[r\left(h(x), y\right)\right]. \tag{6.2}$$

6.3.2 Integration of Knowledge

We consider a commonly-used informed learning method, i.e., integrating knowledge into the neural network during the training stage [396]. During training, a labeled dataset $S_z = \{(x_1, z_1), \cdots, (x_{n_z}, z_{n_z})\}$ with n_z samples drawn from $\mathbb{P}_{XZ}$ is provided. We assume $x_i, i \in [n]$ are drawn from the distribution $\mathbb{P}_X$, but the training label $z_i \in \mathcal{Y}$ may not be the same as the true label y_i for the input x_i, because the training label may be of low quality (e.g., corrupted, noisy, and/or quantized)[80, 445]. Denote $h_{\boldsymbol{W},i} = h_{\boldsymbol{W}}(x_i)$ as the output of the neural network with respect to the input x_i. Based on the labeled dataset, the empirical label-based risk can be written as $\hat{R}_{S_z}(\boldsymbol{W}) = \frac{1}{n_z}\sum_{S_z} r\left(h_{\boldsymbol{W},i}, z_i\right)$.

The domain knowledge includes a knowledge-based model $g(x)$ regarding the input x and a knowledge-based risk function $r_{\mathrm{K}}\left(h_{\boldsymbol{W}}(x), g(x)\right)$ that relates the DNN's output $h_{\boldsymbol{W}}(x)$ to $g(x)$.

For the ease of analysis, we assume that both the risk function r and the knowledge-based risk function r_{K} are Lipschitz continuous, upper bounded, and strongly convex with respect to the network output, and the eigenvalues of their Hessian matrix regarding the network output lie in $[\rho, 1]$ for $\rho \in (0, 1]$. Note that the incorporated domain knowledge

may not necessarily be perfect since it can be obtained based on subjective experiences (e.g., medical prognosis) [302, 62], pre-existing machine learning models [194] or theoretical models which itself can deviate from the real physical world [203].

For training, in addition to the labeled dataset S_z, a dataset S_g with n_g unlabeled samples is generated for knowledge-based supervision. Note that S_g can also include inputs in S_z, and n_g can be sufficiently large since unlabeled samples are typically easier to obtain than labeled ones. The training risk of the informed neural network, which we simply refer to as *informed* risk, is

$$\hat{R}_{\mathrm{I}}(\boldsymbol{W})=\frac{1-\lambda}{n_z}\sum_{S_z} r(h_{\boldsymbol{W},i},z_i)+\frac{\lambda}{n_g}\sum_{S_g} r_{\mathrm{K}}(h_{\boldsymbol{W},i},g_i)\,, \tag{6.3}$$

where $\lambda \in [0,1]$ is a hyper-weight, $h_{\boldsymbol{W},i} = h_{\boldsymbol{W}}(x_i)$, and $g_i = g(x_i)$. Note that Eqn. (6.3) can also be re-written as

$$\hat{R}_{\mathrm{I}}(\boldsymbol{W})=\sum_{S_z \bigcup S_g} \left[\mu_i r\left(h_{\boldsymbol{W},i}, z_i\right)+\lambda_i r_{\mathrm{K}}\left(h_{\boldsymbol{W},i}, g_i\right)\right] \tag{6.4}$$

with hyper-parameters chosen as $\mu_i = \frac{1-\lambda}{n_z}\mathbb{1}(x_i \in S_z)$ and $\lambda_i = \frac{\lambda}{n_g}\mathbb{1}(x_i \in S_g)$. Eqn. (6.4) is used for convergence analysis.

To train the informed DNN, we consider a gradient descent approach in Algorithm 8. This training approach has also been commonly considered in the literature [16, 449, 136] for theoretical analysis of standard DNNs without domain knowledge. For the sake of analysis, we also define a hypothesis space $\mathcal{H} = \left\{h_{\boldsymbol{W}} \mid \boldsymbol{W} \in \mathcal{B}\left(\boldsymbol{W}^{(0)}, \tau\right)\right\}$ where $\boldsymbol{W}^{(0)}$ is the initialized weight and τ is the maximum distance between the weights in gradient descent and the initialized weights. We denote $h_l^{(0)}(x), l \in [L]$ as the output of the l-th layer for an input x at initialization.

Algorithm 8 Informed Neural Network Training by Gradient Descent

Initialization: Initialize each entry of weights $\boldsymbol{W}_0^{(0)}$, $\boldsymbol{W}_l^{(0)}, l \in [L]$ independently by $\mathcal{N}\left(0, \frac{2}{m}\right)$ and each entry of $\boldsymbol{V}^{(0)}$ independently by $\mathcal{N}\left(0, \frac{1}{d}\right)$.

for $t = 0, \cdots, T-1$ **do**

Update the weights as $\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} + \eta \bigtriangledown_{\boldsymbol{W}} \hat{R}_{\mathrm{I}}\left(\boldsymbol{W}^{(t)}\right)$.

end for

Output: $\boldsymbol{W}^{(T)}$.

Remark 30. *The considered informed learning is relevant to several other frameworks. For example, it can model weakly-supervised learning [445, 400] with a few (possibly imperfectly) labeled samples as well as other weak supervision signals (i.e., knowledge). Besides, by viewing $\{z_i\}$ as hard labels and the knowledge-based model $g(x)$ as soft labels provided by a teacher model, the informed learning captures knowledge distillation [194, 319, 327]. Thus, our work can complement the existing analysis for the aforementioned learning frameworks from a different and more unified perspective. Additionally, PAC-Bayesian learning optimizes the PAC-Bayesian bound which is a trade-off between the empirical error and a regularization term based on a prior distribution given by knowledge [185, 22, 163]. But, different from PAC-Bayesian learning which considers random hypothesis, we analyze an over-parameterized neural network with a predetermined architecture.*

6.4 Effects of Domain Knowledge

6.4.1 Convergence

Since the domain knowledge is integrated into a neural network during training, it is important to analyze the convergence to understand how the label and knowledge supervision jointly determine the network output. While convergence based on gradient descent for over-parameterized neural networks has been studied extensively [44, 16, 449, 33, 136], the current analysis is *not* suitable to study the convergence of informed over-parameterized neural networks. The reasons are summarized as follows.

• **Inapplicable for Multiple Supervisions.** Typically, assuming one unique label for each distinct training sample and a large enough network width, the prior studies show that the neural network can fit to the labels, i.e., the network output for each training input converges to the corresponding label [432, 33, 449, 313]. But, in our case, one training input can have multiple supervisions from both label and knowledge with possibly different forms of risks. Thus, the network output for an input may not be necessarily determined by a unique label. The convergence of knowledge distillation supervised by both hard and soft labels is studied by [327], but only the quadratic risk and shallow networks are considered.

• **Strong Data Separability Assumption.** Some prior studies require a lower-bounded distance of any two samples [16, 449, 136], but this may not be satisfied for an informed DNN because the input samples for label-based and knowledge-based risks can be very close or even the same. Other studies assume data separability by a neural tangent model [101, 208, 82, 308], but data separability by a neural tangent model is not well defined for training with multiple supervisions in informed DNNs.

To address these challenges, we provide convergence analysis for informed over-parameterized neural networks based on a new data separability assumption of smooth sets. The construction of smooth sets approximates the space $\mathcal{X}$ with discrete pieces, each containing samples that jointly satisfy the smooth properties. The smooth sets are formally defined below, followed by the data separability assumption.

Definition 31 (Smooth sets)**.** *Given $\phi > 0$, construct a $\phi-$net [113] $\mathcal{X}_\phi = \{x'_k, k \in [N], x'_k \in \mathcal{X}\}$ with $N \sim O(1/\phi^b)$ such that $\forall x'_i, x'_j \in \mathcal{X}_\phi$ and $x'_i \neq x'_j$, $\|x'_i - x'_j\| \geq \phi$ holds, and $\forall x_i \in S_z \bigcup S_g$, there exists at least one $x'_k \in \mathcal{X}_\phi$ satisfying $\|x_i - x'_k\| \leq \phi$. Each input $x'_k \in \mathcal{X}_\phi$, referred to as a representative input, determines a smooth set $\mathcal{C}_{\phi,k} = \{x \in \mathcal{X} \mid \|x - x'_k\| \leq \phi, \|x - x'_j\| \geq \phi/2, \forall j \neq k, x'_k, x'_j \in \mathcal{X}_\phi\}$. The index set of training samples within the kth smooth set is $\mathcal{I}_{\phi,k} = \{i \mid x_i \in S_z \bigcup S_g, x_i \in \mathcal{C}_{\phi,k}\}, k \in [N]$.*

Assumption 6 (Data separability by smooth sets)**.** *For each smooth set k with representative sample x'_k, there exists a non-empty subset of neuron indices $\mathcal{G}_{k,\alpha} \in [m]$ with size $|\mathcal{G}_{k,\alpha}| = \alpha m, \alpha \in (0, 1]$ such that at initialization, $\forall i \in \mathcal{I}_{\phi,k}$, $\forall j \in \mathcal{G}_{k,\alpha}$, $\mathbb{1}\left(\left[h_L^{(0)}(x_i)\right]_j \geq 0\right) = \mathbb{1}\left(\left[h_L^{(0)}(x'_k)\right]_j \geq 0\right)$, and $\forall j \notin \mathcal{G}_{k,\alpha}$, the pre-activation of the L-th layer $\left|\left[\boldsymbol{W}_L^{(0)} h_{L-1}^{(0)}(x_i)\right]_j\right| \geq \frac{3\sqrt{2\pi}\phi^{b+1}}{16\sqrt{m}}$.*

Instead of requiring a lower-bounded distance of any two training samples, the data separability assumption requires that, at initialization, for samples in one smooth set, the outputs of the last hidden layer either have the same signs as those of the representative sample, or their absolute values are larger than a very small threshold. Thus, this data separability assumption is set-wise and addresses the cases where two training inputs are very close or the same, and hence is milder than the one in existing studies (e.g., [16]). The

parameter α indicates slackness: with larger α, more neurons have the same signs. Actually, data separation by smooth set with $\phi > 0$ in Assumption 6 always exists: when ϕ is small enough such that only one inputs or several same inputs are included in a smooth set, Assumption 6 is satisfied with $\alpha = 1$. Even in this worst case, our assumption is still milder than the data separability assumption considered in [16, 449] that excludes the existence of two training samples with the same inputs but different supervisions.

With the data-separability assumption by smooth sets, we are ready to show the labels and knowledge jointly determine the network output for training inputs. We introduce the notation *effective label*, as formally defined below.

Definition 32 (Effective label)**.** *For the k-th smooth set, define the effective label as $y_{\text{eff},k} = \arg\min_h \sum_{i \in \mathcal{I}_{\phi,k}} \{\mu_i r(h, z_i) + \lambda_i r_K(h, g_i)\}$ with μ_i, λ_i defined in Eqn.* (6.3) *and h in the space of network output, and the effective optimal risk as $r_{\text{eff},k} = \sum_{i \in \mathcal{I}_{\phi,k}} \{\mu_i r(y_{\text{eff},k}, z_i) + \lambda_i r_K(y_{\text{eff},k}, g_i)\}$.*

Next, we show the convergence analysis. Note that the proof based on the data separability by smooth sets (Assumption 6) invalidates the proofs in previous studies, and we need new lemmas that lead to novel convergence to effective labels in Definition 32. In particular, in Lemma 33, to approximate the outputs in the smooth set k by the output of the representative input x'_k, we need to bound the difference of the outputs with respect to x'_k and an input in the smooth set k. Also, based on Assumption 6, we derive in Lemma 34 the gradient lower bound which relies on the number of smooth sets N instead of the sample size $n_z + n_g$ in the previous analysis. This makes the network width m in our analysis directly rely on the smooth set size ϕ. Moreover, in Lemma 35, we prove based on the definition of smooth sets that the first-order approximation error of the total informed risk depends on

the difference between the risk and effective risk in Definition 32. This is important to prove the convergence to the effective labels.

Lemma 33. *For any $i \in \mathcal{I}_{\phi,k}, k \in [N]$, let $h_{l,k} = h_l(x'_k)$, $h_{l,i} = h_l(x_i)$, and $f_{l,k} = \boldsymbol{W}_l h_{l-1}(x'_k)$, $f_{l,i} = \boldsymbol{W}_l h_{l-1}(x_i)$ and denote $\boldsymbol{D}'_{l,i,k} \in \mathbb{R}^{m\times m}$ as the diagonal matrix with $[\boldsymbol{D}'_{l,i,k}]_{j,j} = \mathbb{1}([f_{l,i}^{(0)}]_j \geq 0) - \mathbb{1}([f_{l,k}^{(0)}]_j \geq 0)$. Assuming $\phi \leq O(L^{-9/2}\log^{-3}(m)\log^{-3/4}(1/\phi))$, we have with probability at least $1-\phi$ over the randomness of $\boldsymbol{W}^{(0)}$,*

(a) At initialization, $\|\boldsymbol{D}'_{l,i,k}\|_0 \leq O(m\phi^{2/3}L\log^{1/2}(1/\phi))$

(b) For $\boldsymbol{W} \in \mathcal{B}(\boldsymbol{W}^{(0)}, \tau)$ with $\tau \leq O(\phi^{3/2})$ we have $\|h_{l,i} - h_{l,k}\| \leq O(L^{5/2}\phi\sqrt{\log(m)\log(1/\phi)})$ and $\|f_{l,i} - f_{l,k}\| \leq O(L^{5/2}\phi\sqrt{\log(m)\log(1/\phi)})$.

Lemma 34 (Gradient Lower Bound)**.** *For any $\boldsymbol{W} : \|\boldsymbol{W} - \boldsymbol{W}^{(0)}\| \leq \tau$ where $\tau = O(N^{-9/2}\phi^{3/2}\rho^{3/2}\bar{\lambda}^{3/2}\alpha^{3/2}L^{-15/2}\log^{-3/2}(m))$ and $\phi \leq \widetilde{O}(L^{-9/2}\log^{-3}(m))$, with Assumption 6 satisfied, we have with probability at least $1 - O(\phi)$ over the randomness of $\boldsymbol{W}^{(0)}$, the gradient of label-based data risk satisfies*

$$\left\|\bigtriangledown_{\boldsymbol{W}}\hat{R}_{\mathrm{I}}\left(\boldsymbol{W}\right)\right\|_F^2 \geq \Omega\left(\frac{\alpha m\phi\rho\bar{\lambda}}{dN^2}\right)\left(\hat{R}_{\mathrm{I}}\left(\boldsymbol{W}\right) - \hat{R}_{\mathrm{eff}} - \widetilde{O}(L^{5/2}\phi\log^{1/2}(m))\right).$$

where $\hat{R}_{\mathrm{eff}} = \sum_{k=1}^N r_{\mathrm{eff},k}$, and $\bar{\lambda}$ is a parameter with lower bound $\Omega(\min(1-\lambda,\lambda)\mathbb{1}(\lambda \in (0,1)) + \mathbb{1}(\lambda \in \{0,1\}))$.

Lemma 35. *For any $\boldsymbol{W}$, $\boldsymbol{W}' \in \mathcal{B}\left(\boldsymbol{W}^{(0)}, \tau\right)$ where $\mathcal{B}\left(\boldsymbol{W}^{(0)}, \tau\right)$ is a ball with center $\boldsymbol{W}^{(0)}$ and radius $\tau \in \left[\Omega(d^{3/2}m^{-3/2}L^{-5/2}\log^{-3/2}(m)), O(L^{-9/2}[\log^{-3}(m)])\right]$ and $\phi \leq \widetilde{O}(L^{-9/2}\log^{-3}(m))$, with probability at least $1 - O(\phi)$ over the randomness of $\boldsymbol{W}^{(0)}$, we have*

$$\hat{R}_{\mathrm{I}}\left(\boldsymbol{W}'\right) \leq \hat{R}_{\mathrm{I}}\left(\boldsymbol{W}\right) + \left\langle \bigtriangledown_{\boldsymbol{W}}\hat{R}_{\mathrm{I}}\left(\boldsymbol{W}\right), \boldsymbol{W}' - \boldsymbol{W}\right\rangle + O(L^2m/d)\left\|\widehat{\boldsymbol{W}}\right\|^2 +$$
$$\left(\sqrt{\left(\hat{R}_{\mathrm{I}}\left(\boldsymbol{W}\right) - \hat{R}_{\mathrm{eff}} - \widetilde{O}(L^{5/2}\phi\log^{1/2}(m))\right)}\right) O\left(N^{1/2}\tau^{1/3}L^{5/2}\sqrt{m\log(m)}d^{-1/2}\right)\left\|\widehat{\boldsymbol{W}}\right\|$$

Theorem 6.4.1. *Assume the network width* $m \geq \Omega\left(\phi^{-11b-4} L^{15} d\rho^{-4} \bar{\lambda}^{-4} \alpha^{-4} \log^3(m)\right)$, *and the step size is set as* $\eta = O(\frac{d}{L^2 m})$. *With Assumptions 6 satisfied, for any* $\epsilon > 0$ *and* $\phi \leq \widetilde{O}\left(\epsilon L^{-9/2} \log^{-3}(m)\right)$, *we have with probability at least* $1 - O(\phi)$, *by gradient descent after* $T = O\left(\frac{L^2}{\phi^{1+2b}\rho\bar{\lambda}\alpha} \log(\epsilon^{-1} \log(\phi^{-1}))\right)$ *steps, the informed risk in Eqn.* (6.4) *is bounded as:* $\hat{R}_{\mathrm{I}}(\boldsymbol{W}^{(T)}) - \hat{R}_{\mathrm{eff}} \leq O(\epsilon)$, *where* $\hat{R}_{\mathrm{eff}} = \sum_{k=1}^{N} r_{\mathrm{eff},k}$, $\bar{\lambda} = \Omega(\min(1-\lambda, \lambda)\mathbb{1}(\lambda \in (0,1)) + \mathbb{1}(\lambda \in \{0,1\}))$. *Also, the DNN outputs satisfy:*

$$\sum_{S_z \bigcup S_g} (\mu_i + \lambda_i) \left\| h_{\boldsymbol{W}^{(T)}}(x_i) - y_{\mathrm{eff},k(x_i)} \right\|^2 \leq O(\epsilon),$$

where $k(x_i)$ *is the index of the smooth set that includes* x_i, $\mu_i = \frac{1-\lambda}{n_z}\mathbb{1}(x_i \in S_z)$ *and* $\lambda_i = \frac{\lambda}{n_g}\mathbb{1}(x_i \in S_g)$.

Remark 36. *The convergence analysis in Theorem 6.4.1 addresses the limitations mentioned at the beginning of this section. First, instead of fitting a unique label for each input, the informed neural network with multiple supervisions converges to effective labels. Second, the data separability assumption is enough for convergence analysis of informed neural networks. Another observation is that with smaller* ϕ *and smaller* α, *Assumption 6 becomes milder, but a larger network width and more training steps are needed to guarantee convergence.*

Additionally, different from previous convergence analysis where the width m *increases directly with the sample size, the network width* m *in our analysis depends on the smooth set size* ϕ *and is non-decreasing with sample size (i.e.,* m *may not always increase with the sample size). To see this, given a construction of smooth sets by size* ϕ *that meets Assumption 6, if we continue to add (either labeled or knowledge-supervised) training samples that lie in the existing smooth sets and satisfy Assumption 6, the width* m *remains the same, and smaller* ϕ *(larger* m*) is needed to guarantee the convergence only when the added samples*

violate Assumption 6 under the current ϕ. The large network width needed for analysis is due to the limitation of over-parameterization techniques, while in practice a much smaller network width is enough. Albeit beyond the scope of our study, addressing the gap between theory and practice is clearly important and still active research in the community [44].

Remark 37. *We can get more insights about the effects of labels and knowledge from the conclusion that the network outputs converge to the corresponding* effective *labels in Definition 32. On the one hand, if knowledge is applied to the samples within the same smooth sets as labeled samples, knowledge-based supervision and label-based supervision jointly determine the network output together: knowledge serves as* a regularization for labels *in this case. On the other hand, if a smooth set only contains knowledge-supervised samples, the network output is determined solely by knowledge: knowledge* supplements labeled samples *(albeit possibly imperfectly) to provide additional supervision.*

6.4.2 Generalization

We now formally analyze how the domain knowledge affects the generalization performance. From our convergence analysis, there are two different effects of knowledge (Remark 37). We characterize the two effects by formally defining knowledge imperfectness and knowledge-regularized label imperfectness. Before this, we list some notations for further analysis. Given a $\phi-$net $\mathcal{X}_\phi$ (Definition 31), $\mathcal{U}_\phi(S_z) = \{k \in [N] \mid \exists x \in S_z, x \in \mathcal{C}_{\phi,k}\}$ is the index collection of smooth sets that contain at least one labeled sample, and $\mathcal{X}_\phi(S_z) = \bigcup_{k\in\mathcal{U}_\phi}(S_z)\mathcal{C}_{\phi,k}$ is the region covered by the smooth sets in $\mathcal{U}_\phi(S_z)$. $S'_g = S_g \bigcap \mathcal{X}_\phi(S_z)$ is the knowledge supervised dataset with samples share the common smooth sets with labeled

samples in S_z while the samples in $S_g'' = S_g \setminus S_g'$ lie in smooth sets without labeled samples. Denote $n_g' = |S_g'|$ and $n_g'' = |S_g''|$.

Definition 38 (Knowledge imperfectness)**.** *Let* $h_{\mathrm{K}}^* = \min_h \frac{1}{n_g''} \sum_{S_g''} [r_{\mathrm{K}}(h(x_i), g(x_i))]$ *be the optimal hypothesis for the knowledge-based risk on the dataset* S_g''*. The imperfectness of domain knowledge* K *applied to the dataset* S_g'' *is defined as* $\widehat{Q}_{\mathrm{K},S_g''} = \frac{1}{n_g''} \sum_{x_i \in S_g''} r(h_{\mathrm{K}}^*(x_i), y_i)$ *where* y_i *is the true label of* x_i*. Correspondingly, let* $\bar{h}_{\mathrm{K}}^* = \min_h \mathbb{E}\left[r_{\mathrm{K}}(h(x), g(x))\right]$ *be the optimal hypothesis for the expected knowledge-based risk, and the expected imperfectness of domain knowledge* K *is defined as* $Q_{\mathrm{K}} = \mathbb{E}\left[r(\bar{h}_{\mathrm{K}}^*(x), y)\right]$.

The (empirical or expected) knowledge imperfectness is defined as the risk under the hypothesis optimally learned by knowledge-based supervision. Thus, it measures the extent to which the domain knowledge is inconsistent with the true labels, measured in terms of the risk over the hypothesis set $\mathcal{H}$. Besides knowledge-based supervision, the network outputs for some smooth sets that contain both samples for knowledge risks and labeled samples are jointly determined by label-based and knowledge-based supervisions. Thus, we define knowledge-regularized label imperfectness below.

Definition 39 (Knowledge-regularized label imperfectness)**.** *The optimal hypothesis for the knowledge-regularized risk is* $h_{\mathrm{R},\beta}^* = \arg\min_h \frac{1-\beta}{n_z} \sum_{S_z} r(h(x_i), z_i) + \frac{\beta}{n_g'} \sum_{S_g'} r_{\mathrm{K}}(h(x_i), g(x_i))$ *with* $\beta \in [0, 1]$*.* $\widehat{Q}_{\mathrm{R},S_z,S_g'}(\beta) = \frac{1}{n_z} \sum_{S_z} r(h_{\mathrm{R},\beta}^*(x_i), y_i)$*, where* y_i *is the true label regarding* x_i*, is the knowledge-regularized label imperfectness. Correspondingly, with* $\bar{h}_{\mathrm{R},\beta}^* = \arg\min_h \mathbb{E}[\frac{1-\beta}{n_z} \sum_{S_z} r(h(x_i), z_i) + \frac{\beta}{n_g'} \sum_{S_g'} r_{\mathrm{K}}(h(x_i), g(x_i))]$ *being the optimal hypothesis for the regularized risk, the expected knowledge regularized label imperfectness is* $Q_{\mathrm{R}}(\beta) = \mathbb{E}\left[r(\bar{h}_{\mathrm{R},\beta}^*(x), y)\right]$.

Like knowledge imperfectness, knowledge-regularized label imperfectness indicates the risk of the hypothesis optimally learned by joint supervision from labels and knowledge. We see that when $\beta = 0$, $\widehat{Q}_{\mathrm{R}}(0)$ (or $Q_{\mathrm{R}}(0)$) is the imperfectness of pure label-based supervision. Thus, the gain due to knowledge is $\Delta\widehat{Q}_{\mathrm{R},\beta} = \widehat{Q}_{\mathrm{R}}(0) - \widehat{Q}_{\mathrm{R}}(\beta)$ (or $\Delta Q_{\mathrm{R},\beta} = Q_{\mathrm{R}}(0) - Q_{\mathrm{R}}(\beta)$ for the expected version). We show in the following theorem how the two types of imperfectness affect the population risk trained on the informed risk in Eqn. (6.3).

Theorem 6.4.2. *With $\boldsymbol{W}^{(T)}$ trained on Eqn. (6.3), $\phi \leq \widetilde{O}\left(\epsilon^2 L^{-9/2}\log^{-3}(m)\right), \phi \leq (\sqrt{\epsilon}/n_z)^{1/b}$, and other assumptions the same as Theorem 6.4.1, with probability at least $1 - O(\phi) - \delta, \delta \in (0,1)$, the population risk satisfies*

$$\begin{aligned} R\left(h_{\boldsymbol{W}^{(T)}}\right) \leq & O(\sqrt{\epsilon}) + (1-\lambda)\widehat{Q}_{\mathrm{R},S_z,S'_g}(\beta_\lambda) \\ & + \lambda\widehat{Q}_{\mathrm{K},S''_g} + O\Big(\Phi + \sqrt{\log(1/\delta)}\Big)\left(\frac{1-\lambda}{\sqrt{n_z}} + \frac{\lambda}{\sqrt{n_g}}\right), \end{aligned}$$

where $\beta_\lambda = \frac{\lambda n'_g}{(1-\lambda)n_g + \lambda n'_g}$, $\widehat{Q}_{\mathrm{R},S_z,S'_g}(\beta_\lambda)$ is the knowledge-regularized label imperfectness in Definition 39 and $\widehat{Q}_{\mathrm{K},S''_g}$ is the knowledge imperfectness in Definition 38 applied to S''_g, and $\Phi = O\left(4^L L^{3/2} m^{1/2}\phi^{-b-1/2} d\rho^{-1/2}\bar{\lambda}^{-1/2}\alpha^{-1/2}\right)$.

Remark 40. *Theorem 6.4.2 shows that by training on the informed risk (6.3), knowledge affects the generation performance in the following two ways.*

- ***Knowledge for regularization.*** *When knowledge is applied to sample inputs inside the same smooth sets as labeled samples, it serves as an explicit regularization for label-based supervision, possibly reducing the label imperfectness from $\widehat{Q}_{\mathrm{R},S_z,S'_g}(0)$ to $\widehat{Q}_{\mathrm{R},S_z,S'_g}(\beta_\lambda)$.*
- ***Knowledge for supplementing labels.*** *The generalization error is in the order of $O\left(\frac{1-\lambda}{\sqrt{n_z}} + \frac{\lambda}{\sqrt{n_g}}\right)$. When no knowledge is used ($\lambda = 0$), the order is as large as $O\left(\frac{1}{\sqrt{n_z}}\right)$. If knowledge is applied ($\lambda > 0$), then the generalization error decreases with the increasing of*

knowledge-supervised sample size n_g. Thus, when knowledge is applied to smooth sets without labeled samples, it serves as an (possibly imperfect) supplement for labels, while introducing knowledge imperfectness $\widehat{Q}_{\mathrm{K},S''_g}$.

The hyper-parameter λ can be used to balance the introduced imperfectness and generalization error from label and knowledge supervision. However, by the risk bound, it is hard to use one hyper-parameter λ to control the two effects of knowledge, which will be further discussed in the next section.

6.5 A Generalized Training Objective

In the informed risk in Eqn. (6.3), only one hyper-weight λ is present, controlling the two different effects of knowledge (Remark 40). To better reap the benefits of knowledge, we consider a generalized informed risk in Eqn.(6.5) by introducing another hyper-weight β, which introduces more flexibility to govern the roles of domain knowledge.

$$\begin{aligned}\hat{R}_{\mathrm{I},G}(\boldsymbol{W}) = &\frac{(1-\lambda)(1-\beta)}{n_z}\sum_{S_z} r\left(h_{\boldsymbol{W},i}, z_i\right) + \\ &\frac{(1-\lambda)\beta}{n'_g}\sum_{S'_g} r_{\mathrm{K}}(h_{\boldsymbol{W},i}, g_i) + \frac{\lambda}{n''_g}\sum_{S''_g} r_{\mathrm{K}}(h_{\boldsymbol{W},i}, g_i)\,,\end{aligned} \tag{6.5}$$

where $\beta, \lambda \in [0,1], h_{\boldsymbol{W},i} = h_{\boldsymbol{W}}(x_i), g_i = g(x_i)$.

In Eqn. (6.5), the two hyper-parameters λ and β can jointly control the knowledge effects (and the introduced imperfectness) when knowledge is applied. The hyperparameter β is used to controls the knowledge regularization strength. By Remark 40, knowledge-supervised samples in S'_g serve as an explicit regularization for label-based supervision while introducing knowledge-regularized label imperfectness $Q_{\mathrm{R}}(\beta)$. Thus, when β is larger, more effects from S'_g are incorporated and the regularization effect from knowledge is stronger.

Also, we use λ to adjust the effect of supplementing labels and the introduction of Q_{K}. By Remark 40, S_g'' serves as an supplement for labels while introducing the knowledge imperfectness Q_{K}. Thus, with larger λ, more effects from S_g'' are incorporated, which means we incorporate more effects of data supplement from knowledge and also knowledge imperfectness Q_{K} but less effect of knowledge regularization and knowledge-regularized label imperfectness Q_{R}. The benefit of the training objective in Eqn. (6.5) will be explained formally in Theorem 6.5.1 and Corollary 41.

Compared with the objective in Eqn. (6.3) with only one hyper-parameter λ, Eqn. (6.5) introduces another hyper-parameter β to independently adjust the degree of the knowledge regularization, making Eqn. (6.5) more general and flexible. To train on Eqn. (6.5), we need to separate dataset for knowledge supervision into two datasets S_g' and S_g'' based on whether an input is close to a labeled input and assign different hyper-weights to them. The knowledge-based dataset separation is determined by ϕ in Definition 31. Specifically, when the network width goes to infinity (ϕ goes to zero), S_g' shares the same inputs as S_z, but S_g' and S_z are supervised by knowledge and labels, respectively. We have $S_g'' = S_g \setminus S_g' = S_g \setminus S_z$ which supplements the labels as shown in Remark 40. Note that when the knowledge is perfect and knowledge-supervised samples are sufficient, we do not need labeled samples, i.e., $S_z = \emptyset$ and we set $\lambda = 1, \beta = 1$. Then, we have $S_g'' = S_g$ and Eqn. (6.5) becomes a purely knowledge-based risk. When no knowledge is applied, we set $\lambda = 0, \beta = 0$, and Eqn. (6.5) becomes a purely lable-based risk. In general cases when labels and knowledge are both used, hyper-parameters λ and β are used to control the effects of knowledge.

6.5.1 Population Risk

Note that Eqn. (6.5) can also be written as the form of Eqn. (6.4) with hyper-parameters chosen as $\mu_i = \frac{(1-\lambda)(1-\beta)}{n_z}\mathbb{1}(x_i \in S_z)$ and $\lambda_i = \frac{(1-\lambda)\beta}{n'_g}\mathbb{1}(x_i \in S'_g) + \frac{\lambda}{n''_g}\mathbb{1}(x_i \in S''_g)$, so Theorem 6.4.1 for convergence still holds. Next, we bound the population risk based on the generalized informed risk.

Theorem 6.5.1. *Assume that $\boldsymbol{W}^{(T)}$ trained on Eqn.* (6.5) *and other assumptions are the same with those of Theorem 6.4.1, setting $\phi : \phi \leq \widetilde{O}\left(\epsilon^2 L^{-9/2}\log^{-3}(m)\right)$ and $\phi \leq (\sqrt{\epsilon}/n_z)^{1/b}$, with probability at least $1 - O(\phi) - \delta, \delta \in (0,1)$, the population risk satisfies*

$$R(h_{\boldsymbol{W}^{(T)}}) \leq O(\sqrt{\epsilon}) + (1-\lambda)\widehat{Q}_{\mathrm{R},S_z,S'_g}(\beta) + \lambda\widehat{Q}_{\mathrm{K},S''_g} + O\left(\Phi + \sqrt{\log(1/\delta)}\right)\left(\frac{1-\lambda}{\sqrt{n_z}} + \frac{\lambda}{\sqrt{n''_g}}\right),$$

where β and λ are trade-off hyper-parameters in Eqn. (6.5)

Additionally, to obtain more insights for sampling complexity, we further bound the population risk in terms of expected imperfectness, at the expense of some tightness.

Corollary 41. *With the same assumptions as in Theorem 6.5.1, with probability at least $1 - O(\phi) - \delta, \delta \in (0,1)$, the population risk satisfies*

$$R(h_{\boldsymbol{W}^{(T)}}) \leq O(\sqrt{\epsilon}) + (1-\lambda)Q_{\mathrm{R}}(\beta) + \lambda Q_{\mathrm{K}} + O\left(\Phi + \log^{1/4}(1/\delta)\right)\sqrt{\frac{1-\lambda}{\sqrt{n_z}} + \frac{\lambda}{\sqrt{n''_g}}},$$

where $Q_{\mathrm{R}}(\beta)$ is the expected knowledge-regularized label imperfectness in Definition 39, Q_{K} is the expected knowledge imperfectness in Definition 38.

Remark 42. *Theorem 6.5.1 and Corollary 41 show that by training on the generalized informed risk in Eqn.* (6.5)*, label and knowledge supervision jointly affect the population*

risk while introducing a combination of knowledge-regularized label imperfectness $Q_{\mathrm{R}}(\beta)$ *and knowledge imperfectness* Q_{K}*. The effect of knowledge regularization is controlled by* β *and the trade-off between the two imperfectness terms and the trade-off between the two generalization errors* $\frac{1-\lambda}{\sqrt{n_z}}$ *and* $\frac{\lambda}{\sqrt{n_g''}}$ *are both controlled by* λ*. Thus, this gives us more flexibility to adjust how much domain knowledge is incorporated when it plays different roles in informed learning as discussed in Remark 40. Also, as shown by the population risk bounds, we can tune the two hyper-parameters separately — we can first tune* β *to minimize* $Q_{\mathrm{R}}(\beta)$*, and then tune* λ *to balance* $Q_{\mathrm{R}}(\beta)$ *and* Q_{K}*, and also balance the generalization errors due to sizes of datasets.*

6.5.2 Sampling Complexity

We discuss the choices of hyper-parameters β and λ in different cases to guarantee a small population risk, and give the sampling complexity in each case.

Corollary 43 (Sampling Complexity)**.** *With the same set of assumptions as in Corollary 41 and setting* $\beta^* = \arg\min_{\beta\in[0,1]} Q_{\mathrm{R}}(\beta)$*, with probability at least* $1 - O(\phi) - \delta, \delta \in (0,1)$*, to guarantee a population risk no larger than* $\sqrt{\epsilon}$*, we have the following cases:*

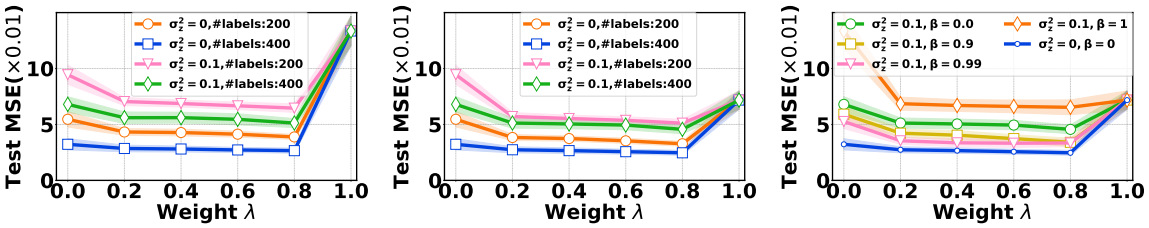
(a) If $Q_{\mathrm{K}} \leq \sqrt{\epsilon}$*, set* $\lambda = 1$*, the sampling complexity for labels is* $n_z = 0$ *and the sampling complexity for knowledge-supervision is* $n_g \sim O(1/(\epsilon^2 - \epsilon^3))$*.*

(b) If $Q_{\mathrm{K}} > \sqrt{\epsilon}$ *and* $\frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}} + \frac{\sqrt{\epsilon}}{Q_{\mathrm{R}}(\beta^*)} \geq 1$*, set* $\lambda = \frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}}$*, the sampling complexity for labels is* $n_z \sim O\left((1/\epsilon - 1/\left(\sqrt{\epsilon}Q_{\mathrm{K}}\right))^2\right)$ *and the sampling complexity for knowledge-supervision is* $n_g \sim O(1/\left((\epsilon - \epsilon^2)Q_{\mathrm{K}}^2\right))$*.*

(c) If $\frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}} + \frac{\sqrt{\epsilon}}{Q_{\mathrm{R}}(\beta^*)} < 1$*, a population risk as low as* $\sqrt{\epsilon}$ *cannot be achieved no matter what* λ *is and how many samples are used.*

Remark 44. *In practice, unlabeled samples are typically cheaper to obtain than* labeled *samples. If $Q_{\mathrm{K}} \leq \sqrt{\epsilon}$, the domain knowledge is good enough for supervision, and thus we can perform purely knowledge-based training without any labeled samples and guarantee a population risk no larger than $\sqrt{\epsilon}$ with $n_g'' \sim O(1/\epsilon^2)$, and hence $n_g \sim O(1/(\epsilon^2 - \epsilon^3))$. When the knowledge imperfectness $Q_{\mathrm{K}} > \sqrt{\epsilon}$, we discuss the following two cases. First, if $\frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}} + \frac{\sqrt{\epsilon}}{Q_{\mathrm{R}}(\beta^*)} \geq 1$, we can choose λ from $\left[1 - \frac{\sqrt{\epsilon}}{Q_{\mathrm{R}}(\beta^*)}, \frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}}\right]$ to control the risk from knowledge and label imperfectness as low as $\sqrt{\epsilon}$. We thus choose the largest $\lambda = \frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}}$ to reduce the label sampling complexity. In this case, knowledge is not good enough, but label imperfectness is not too large. Thus, we can guarantee a population risk no larger than $\sqrt{\epsilon}$ with labeled samples $n_z \sim O\left((1/\epsilon - 1/\left(\sqrt{\epsilon}Q_{\mathrm{K}}\right))^2\right)$ and knowledge supervised samples $n_g \sim O(1/\left((\epsilon - \epsilon^2)Q_{\mathrm{K}}^2\right))$. Finally, if $\frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}} + \frac{\sqrt{\epsilon}}{Q_{\mathrm{R},\beta^*}} < 1$, we cannot guarantee a population risk less than $\sqrt{\epsilon}$ no matter what λ is and how many samples are used since the neither knowledge nor labels are of high enough quality.*

In summary, the extreme cases are: Case (a) where the knowledge supervision alone is nearly perfect, and Case (c) where the knowledge and labels are both of low quality. Usually, we are in Case (b) where knowledge is imperfect but labels (after knowledge regularization) are good enough. In contrast, DNNs without using domain knowledge requires the label imperfectness $Q_{\mathrm{R},0}$ not to exceed $\sqrt{\epsilon}$; otherwise, the population risk cannot be guaranteed to be no greater than $\sqrt{\epsilon}$. The informed DNNs relaxes this requirement by requiring $\frac{\sqrt{\epsilon}}{Q_{\mathrm{K}}} + \frac{\sqrt{\epsilon}}{Q_{\mathrm{R}}(\beta^*)} \geq 1$. In addition, the incorporation of domain knowledge reduces the labeled sampling complexity from $n_z \sim O(\frac{1}{\epsilon^2})$ in the traditional no-knowledge setting to $n_z \sim O\left((1/\epsilon - 1/\left(\sqrt{\epsilon}Q_{\mathrm{K}}\right))^2\right)$. In other words, the incorporation of knowledge is equivalent to $O(\frac{2}{\epsilon^{3/2}Q_{\mathrm{K}}} - \frac{1}{\epsilon Q_{\mathrm{K}}^2})$ labeled



(a) (b) (c)

Figure 6.1: Test MSE under different hyper-parameters.

samples, establishing a quantitative comparison between knowledge supervision and labeled samples.

6.6 Further Discussions

Summary of Analysis. The convergence analysis in Theorem 6.4.1 introduces the concept of smooth sets and explains how the neural network output behaves by training on an informed risk. The generalization analysis in Theorem 6.4.2 explicitly shows the two different effects the domain knowledge has on the population risk (i.e., regularizing labels and supplementing labels). Based on this observation, we propose a generalized informed risk in Eqn. 6.5 to get more flexibility to control the two effects of knowledge, which is validated by Theorem 6.5.1 and its Corollary 41. Finally, the sampling complexity in Corollary 43 shows the effects of joint knowledge and label supervision in a quantitative way.

Understanding Knowledge Distillation From the Perspective of Informed Learning. Knowledge distillation is extremely useful in practice (e.g., for model compression [194]). Here, we show how our analysis complement the existing understanding of knowledge distillation [194, 319, 327, 119, 207] from the perspective of hard label and teacher's knowledge

imperfectness. In our formulation, hard labels are $\{z_i\}$ in the labeled dataset, whose imperfectness (non-softness) is measured by $Q_{\mathrm{R}}(0)$. In Theorems 6.4.2, 6.5.1, and Corollary 41, by viewing the teacher model $g(x)$ as domain knowledge, we show the teacher benefits the student training by providing a regularization gain $\Delta Q_{\mathrm{R},\beta}$, and reducing the sampling complexity of hard labels by Corollary 43. The knowledge-regularized label imperfectness $Q_{\mathrm{R},\beta}$ can be less than pure lable imperfectness $Q_{\mathrm{R}}(0)$ because the soft label can smooth the network output within each smooth set. But, given the teacher (knowledge) imperfectness Q_{K}, there exists a trade-off between hard label and teacher supervision.

Importantly, our results are in line with the observations and also complement the analysis in [207]. Specifically, [207] uses NTK to show that the soft labels provided by a teacher model (knowledge) are easier to learn than hard labels while hard labels can correct imperfect teachers pointwise, exhibiting a trade-off between hard labels and the imperfect teacher. We define the hard label and teacher (knowledge) imperfectness, and show that for a neural network with finite width, hard labels and teacher's knowledge compensate for each other within each smooth set. In consistency with our results, [327] based on NTK also presents a trade-off between labels and the imperfect teacher. The teacher model imperfectness is also observed by [119] which measures the teacher imperfectness by the squared norm of the difference of the soft label and the true Bayesian class probability. Note, however, that our analysis *cannot* adequately explain the benefit of knowledge distillation for the perspective of feature learning due to the inherent limitations of over-parameterization techniques, which are further discussed in [13].

6.7 Numerical Results

6.7.1 Problem Setup

We consider an informed DNN with domain knowledge in the form of constraints to learn a Bohachevsky function. The learning task is to learn a relationship $y(x)$. The learner is provided with a dataset with labeled samples $S_z = \{(x_i, z_i), i \in [n_z]\}$, having possibly noisy labels $z_i = y(x_i) + n_i, n_i \sim \mathcal{N}(0, \sigma_z^2)$, and an unlabeled dataset $S_g = \{(x_i), i \in [n_g]\}$. Additionally, the learner is informed with the constraint knowledge, which includes an upper bound $g_{\text{ub}}(x)$ and an lower bound $g_{\text{lb}}(x)$ on the true label corresponding to input x, i.e. $g_{\text{lb}}(x) \leq y(x) \leq g_{\text{ub}}(x)$. A neural network $h_{\boldsymbol{W}}(x)$ is used for learning and the metric of interest is the mean square error (MSE) of the network output $h_{\boldsymbol{W}}(x)$ with respect to the true label $y(x)$ on a test dataset S_t, which is expressed as $\hat{R}_{S_t}(h_{\boldsymbol{W}}) = \frac{1}{2|S_t|} \sum_{(x_i,y_i)\in S_t} \left(h_{\boldsymbol{W}}(x_i) - y_i\right)^2$. Assume that the relationship to be learned is governed by a multi-dimensional Bohachevsky function $y(x) = x\boldsymbol{A}\boldsymbol{A}^\top x^\top - c\cos\left(a^\top x\right) + c$, where $\boldsymbol{A}$ is a $b \times b$ matrix, a is a b-dimensional vector and c is a constant. The constraint knowledge includes an upper bound model $g_{\text{ub}}(x) = x\boldsymbol{A}\boldsymbol{A}^\top x^\top + ub$ with $ub \geq 2c$, and an lower bound model $g_{\text{lb}}(x) = x\boldsymbol{A}\boldsymbol{A}^\top x^\top + lb.$ with $lb \leq 0$. While it is not strongly convex and hence deviates from the assumptions in our theoretical analysis, we use ReLU as the knowledge-based risk function, i.e., the knowledge-based risk is written as $r_{\text{K}}(h_{\boldsymbol{W}}(x)) = \text{relu}\left(h_{\boldsymbol{W}}(x) - g_{\text{ub}}(x)\right) + \text{relu}\left(g_{\text{lb}}(x) - h_{\boldsymbol{W}}(x)\right)$. If $ub - lb$ is larger, the uncertainty of the label given the knowledge is larger — the knowledge imperfectness is higher. We choose (lb, ub) as $(0, 0.6)$ and $(0, 0.8)$ respectively to show the performances under low and high knowledge imperfectness.

6.8 Conclusion

In this chapter, we consider knowledge informed DNN. We quantitatively demonstrate that domain knowledge can improve the generalization performance and reduce the sampling complexity, while also impacting the point to which the network output converges. Our analysis also reveals that knowledge affects the generalization performance in two ways: regularizing the label supervision, and supplementing the labeled samples. Finally, we discuss how an informed DNN relates to other learning frameworks.

Chapter 7

Conclusions

The dissertation studies algorithms and theory of learning-augmented algorithms exploting domain expert knowledge. We design provable learning-augmented algorithms that can be used for a wide range online decision making problems. Specifically, we propose robust bandit algorithms for bandit decision making with imperfect context. Then, we consider online optimization and control problems with known dynamic model and design learning-augmented algorithms with guaranteed competitiveness. In addition, for MDP without the knowledge dynamic model, we propose learning-augmented algorithms which optimize the average reward under anytime cost constraints. Moreover, we consider online decision making with budget constraints and design a learning assisted unrolling algorithm to solve it. Last by not least, we provide a theoretical analysis for domain knowledge informed learning, demonstrating the benefits of domain knowledge in machine learning.

Bibliography

[1] The cifar-10 dataset (`https://www.cs.toronto.edu/~kriz/cifar.html`).

[2] Downlink and uplink transmission. `https://www.sciencedirect.com/topics/engineering/downlink-and-uplink-transmission`.

[3] The mnist database of handwritten digits (`http://yann.lecun.com/exdb/mnist/`).

[4] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *NeurIPS*, 2011.

[5] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[6] Jonas Adler and Ozan Öktem. Learned primal-dual reconstruction. *IEEE transactions on medical imaging*, 37(6):1322–1332, 2018.

[7] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

[8] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *COLT*, 2012.

[9] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. *ICML*, 2013.

[10] Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1405–1424, 2014.

[11] Irfan Ahmed, Hedi Khammari, Adnan Shahid, Ahmed Musa, Kwang Soon Kim, Eli De Poorter, and Ingrid Moerman. A survey on hybrid beamforming techniques in 5g: Architecture and system model perspectives. *IEEE Communications Surveys & Tutorials*, 20(4):3060–3097, 2018.

[12] Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. *arXiv preprint arXiv:2006.10246*, 2020.

[13] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[14] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*, 2018.

[15] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. *arXiv preprint arXiv:1810.12065*, 2018.

[16] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[17] Mohammad Ali Alomrani, Reza Moravej, and Elias B Khalil. Deep policies for online bipartite matching: A reinforcement learning approach. *arXiv preprint arXiv:2109.10380*, 2021.

[18] Jason Altschuler, Victor-Emmanuel Brunel, and Alan Malek. Best arm identification for contaminated bandits. *Journal of Machine Learning Research*, 20(91):1–39, 2019.

[19] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in Neural Information Processing Systems*, 31:7775–7784, 2018.

[20] Sanae Amani, Christos Thrampoulidis, and Lin Yang. Safe reinforcement learning with linear function approximation. In *International Conference on Machine Learning*, pages 243–253. PMLR, 2021.

[21] Amazon. Amazon aws auto scaling documentation. `https://docs.aws.amazon.com/autoscaling/`, 2021.

[22] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *ICML*, pages 205–214, 2018.

[23] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR, 2017.

[24] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[25] Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*, pages 272–281, 2019.

[26] Lachlan Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *Conference on Learning Theory*, pages 741–763. PMLR, 2013.

[27] Matthew Andrews, Krishnan Kumaran, Kavita Ramanan, Alexander Stolyar, Phil Whiting, and Rajiv Vijayakumar. Providing quality of service over a shared wireless link. *IEEE Communications magazine*, 39(2):150–154, 2001.

[28] 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 *Advances in neural information processing systems*, pages 3981–3989, 2016.

[29] Anonymous. Decentralized learning for overparameterized problems: A multi-agent kernel approximation approach. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. under review.

[30] Anonymous. A global convergence theory for deep reLU implicit networks via over-parameterization. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. under review.

[31] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *ICML*, 2020.

[32] Antonios Antoniadis and Kevin Schewior. A tight lower bound for online convex optimization with switching costs. In *International Workshop on Approximation and Online Algorithms*, pages 164–175. Springer, 2017.

[33] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *ICML*, 2019.

[34] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

[35] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[36] Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 264–273. PMLR, 2018.

[37] Idan Attias, Aryeh Kontorovich, and Yishay Mansour. Improved generalization bounds for robust learning. In *Algorithmic Learning Theory*, pages 162–183, 2019.

[38] J. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. *COLT*, 2009.

[39] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[40] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002.

[41] Peter Auer and Chao-Kai Chiang. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. In *COLT*, 2016.

[42] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

[43] Yossi Azar, Niv Buchbinder, TH Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Naor, et al. Online algorithms for covering and packing problems with convex objectives. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 148–157. IEEE, 2016.

[44] Yasaman Bahri, Quanquan Gu, Amin Karbasi, and Hanie Sedghi. Over-parameterization: Pitfalls and opportunities. In *ICML Workshop*, 2021.

[45] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[46] Maria-Florina Balcan. Rademacher complexity. `http://www.cs.cmu.edu/~ninamf/ML11/lect1117.pdf`, 2011.

[47] Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 919–932, 2021.

[48] Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. Dual mirror descent for online allocation problems. In *International Conference on Machine Learning*, pages 613–628. PMLR, 2020.

[49] Santiago R Balseiro, Haihao Lu, and Vahab Mirrokni. The best of many worlds: Dual mirror descent for online allocation problems. *Operations Research*, 2022.

[50] Robert Bamler, Farnood Salehi, and Stephan Mandt. Augmenting and tuning knowledge graph embeddings. In *Uncertainty in Artificial Intelligence*, pages 508–518. PMLR, 2020.

[51] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs, Kevin Schewior, and Cliff Stein. A 2-competitive algorithm for online convex optimization with switching costs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[52] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *ICML*, 2013.

[53] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

[54] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[55] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[56] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[57] Christian Beck, E Weinan, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

[58] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.

[59] Sagie Benaim and Lior Wolf. One-shot unsupervised cross domain translation. In *Advances in Neural Information Processing Systems*, pages 2104–2114, 2018.

[60] Arka A Bhattacharya, David Culler, Eric Friedman, Ali Ghodsi, Scott Shenker, and Ion Stoica. Hierarchical scheduling for diverse datacenter workloads. In *Proceedings of the 4th annual Symposium on Cloud Computing*, pages 1–15, 2013.

[61] Wei Bi and James Kwok. Efficient multi-label classification with many labels. In *International Conference on Machine Learning*, pages 405–413, 2013.

[62] Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. In *ICLR*, 2020.

[63] Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.

[64] Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.

[65] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[66] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with gaussian processes. In *NIPS*, 2018.

[67] Andrea Borghesi, Federico Baldo, and Michela Milano. Improving deep learning models via constraint-based domain knowledge: a brief survey. *arXiv preprint arXiv:2005.10691*, 2020.

[68] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[69] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, pages 169–207, 2004.

[70] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *SIGACT News*, 47(3):93–129, August 2016.

[71] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[72] Peter J Brockwell, Peter J Brockwell, Richard A Davis, and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2016.

[73] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5:1–122, 2012.

[74] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.

[75] Lucian Buşoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018.

[76] Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*, pages 1283–1294. PMLR, 2020.

[77] Duncan S Callaway, Meredith Fowlie, and Gavin McCormick. Location, location, location: The variable value of renewable energy and demand-side efficiency resources. *Journal of the Association of Environmental and Resource Economists*, 5(1):39–75, 2018.

[78] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[79] Oana-Maria Camburu. Explaining deep neural networks. *arXiv preprint arXiv:2010.01496*, 2020.

[80] Timothy I Cannings, Yingying Fan, and Richard J Samworth. Classification with imperfect training labels. *Biometrika*, 107(2):311–330, 2020.

[81] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *arXiv preprint arXiv:1905.13210*, 2019.

[82] Yuan Cao and Quanquan Gu. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[83] Constantine Caramanis and Sujay Sanghavi. Projection onto a convex set. `https://users.ece.utexas.edu/~cmcaram/EE381V_2012F/Lecture_3_Scribe_Notes.final.pdf`, 2012.

[84] Agustin Castellano, Hancheng Min, Juan Bazerque, and Enrique Mallada. Reinforcement learning with almost sure constraints. In *Learning for Dynamics and Control*, 2022.

[85] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[86] Semih Cayci, Niao He, and R Srikant. Finite-time analysis of entropy-regularized neural natural actor-critic algorithm. *arXiv preprint arXiv:2206.00833*, 2022.

[87] Bingqing Chen, Priya L Donti, Kyri Baker, J Zico Kolter, and Mario Bergés. Enforcing policy feasibility constraints through differentiable projection for energy optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pages 199–210, 2021.

[88] Bokan Chen, Jianhui Wang, Lizhi Wang, Yanyi He, and Zhaoyu Wang. Robust optimization for transmission expansion planning: Minimax cost vs. minimax regret. *IEEE Transactions on Power Systems*, 29(6):3069–3077, 2014.

[89] Lixing Chen and Jie Xu. Budget-constrained edge service provisioning with demand estimation via bandit learning. *IEEE Journal on Selected Areas in Communications*, 37(10):2364–2376, 2019.

[90] Lixing Chen, Jie Xu, Shaolei Ren, and Pan Zhou. Spatio–temporal edge service placement: A bandit learning approach. *IEEE Transactions on Wireless Communications*, 17(12):8388–8401, 2018.

[91] Niangjun Chen, Anish Agarwal, Adam Wierman, Siddharth Barman, and Lachlan LH Andrew. Online convex optimization using predictions. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 191–204, 2015.

[92] Niangjun Chen, Joshua Comden, Zhenhua Liu, Anshul Gandhi, and Adam Wierman. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS*, 2016.

[93] Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Conference On Learning Theory*, pages 1574–1594, 2018.

[94] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *NeurIPS*, 2018.

[95] Shuang Chen, Christina Delimitrou, and José F Martínez. Parties: Qos-aware resource partitioning for multiple interactive services. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 107–120, 2019.

[96] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*, 2021.

[97] Weiqin Chen, Dharmashankar Subramanian, and Santiago Paternain. Policy gradients for probabilistic constrained reinforcement learning, 2022.

[98] Xin Chen, Guannan Qu, Yujie Tang, Steven Low, and Na Li. Reinforcement learning for decision-making and control in power systems: Tutorial, review, and vision. *arXiv*, 2021.

[99] Yaran Chen, Ruiyuan Gao, Fenggang Liu, and Dongbin Zhao. Modulenet: Knowledge-inherited neural architecture search. *arXiv preprint arXiv:2004.05020*, 2020.

[100] Yize Chen, Yushi Tan, and Baosen Zhang. Exploiting vulnerabilities of load forecasting through adversarial attacks. In *e-Energy*, 2019.

[101] Zixiang Chen, Yuan Cao, Difan Zou, and Quanquan Gu. How much over-parameterization is sufficient to learn deep relu networks? *ICLR*, 2021.

[102] Zixiang Chen, Chris Junchi Li, Angela Yuan, Quanquan Gu, and Michael I Jordan. A general framework for sample-efficient function approximation in reinforcement learning. *arXiv preprint arXiv:2209.15634*, 2022.

[103] Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1141–1150. PMLR, 09–15 Jun 2019.

[104] Yuri Chervonyi, Praneet Dutta, Piotr Trochim, Octavian Voicu, Cosmin Paduraru, Crystal Qian, Emre Karagozler, Jared Quincy Davis, Richard Chippendale, Gautam Bajaj, et al. Semi-analytical industrial cooling system model for reinforcement learning. *arXiv preprint arXiv:2207.13131*, 2022.

[105] Mung Chiang, Prashanth Hande, and Tian Lan. *Power control in wireless cellular networks*. Now Publishers Inc, 2008.

[106] Jakub Chłedowski, Adam Polak, Bartosz Szabucki, and Konrad Tomasz Żołna. Robust learning-augmented caching: An experimental study. In *ICML*, 2021.

[107] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4794–4802, 2019.

[108] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. Gram: graph-based attention model for healthcare representation learning. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 787–795, 2017.

[109] S. R. Chowdhury and A. Gopalan. On kernelized multi-armed bandits. *ICML*, 2017.

[110] Nicolas Christianson, Tinashe Handina, and Adam Wierman. Chasing convex bodies and functions with black-box advice. In *COLT*, 2022.

[111] Nicolas Christianson, Junxuan Shen, and Adam Wierman. Optimal robustness-consistency tradeoffs for learning-augmented metrical task systems. In *AISTATS*, 2023.

[112] W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. *NeurIPS*, 2011.

[113] Kenneth L Clarkson. Building triangulations using ε-nets. In *STOC*, 2006.

[114] Joshua Comden, Sijie Yao, Niangjun Chen, Haipeng Xing, and Zhenhua Liu. Online optimization in cloud resource provisioning: Predictions, regrets, and algorithms. *ACM SIGMETRICS*, 2019.

[115] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[116] Wei Cui, Kaiming Shen, and Wei Yu. Spatial deep learning for wireless scheduling. *IEEE Journal on Selected Areas in Communications*, 37(6):1248–1261, 2019.

[117] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, 2017.

[118] V. Dani, T. Hayes, P. Thomas, and S. Kakade. Stochastic linear optimization under bandit feedback. *COLT*, 2008.

[119] Tri Dao, Govinda M Kamath, Vasilis Syrgkanis, and Lester Mackey. Knowledge distillation as semiparametric inference. *arXiv preprint arXiv:2104.09732*, 2021.

[120] Google DeepMind. Safety-first ai for autonomous data centre cooling and industrial control. `https://www.deepmind.com/blog/safety-first-ai-for-autonomous-data-centre-cooling-and-industrial-control`, 2018.

[121] Timo M Deist, Andrew Patti, Zhaoqi Wang, David Krane, Taylor Sorenson, and David Craft. Simulation-assisted machine learning. *Bioinformatics*, 35(20):4072–4080, 2019.

[122] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. *ACM SIGPLAN Notices*, 48(4):77–88, 2013.

[123] Changyu Deng, Xunbi Ji, Colton Rainey, Jianyu Zhang, and Wei Lu. Integrating machine learning with human knowledge. *iScience*, 23(11):101656, 2020.

[124] A. A. Deshmukh, U. Dogan, and C. Scott. Multi-task learning for contextual bandits. *NeurIPS*, 2017.

[125] Nikhil R Devanur and Thomas P Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 71–78, 2009.

[126] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *Journal of the ACM (JACM)*, 66(1):1–41, 2019.

[127] Boya Di, Lingyang Song, and Yonghui Li. Sub-channel assignment, power allocation, and user scheduling for non-orthogonal multiple access networks. *IEEE Transactions on Wireless Communications*, 15(11):7686–7698, 2016.

[128] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

[129] Michelangelo Diligenti, Soumali Roychowdhury, and Marco Gori. Integrating prior knowledge into deep learning. In *ICMLA*, pages 920–923, 2017.

[130] Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably efficient safe exploration via primal-dual policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3304–3312. PMLR, 2021.

[131] Dongsheng Ding, Kaiqing Zhang, Tamer Basar, and Mihailo Jovanovic. Natural policy gradient primal-dual method for constrained markov decision processes. *Advances in Neural Information Processing Systems*, 33:8378–8390, 2020.

[132] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. Measuring the carbon intensity of ai in cloud instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1877–1894, 2022.

[133] Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

[134] Bingqian Du, Chuan Wu, and Zhiyi Huang. Learning resource allocation and pricing for cloud profit maximization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7570–7577, 2019.

[135] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[136] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685, 2019.

[137] Yihan Du, Siwei Wang, and Longbo Huang. A one-size-fits-all solution to conservative bandit problems. In *AAAI*, 2021.

[138] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

[139] Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*, 2020.

[140] Ran El-Yaniv, Amos Fiat, Richard M Karp, and Gordon Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.

[141] Birhanu Eshete. Making machine learning trustworthy. *Science*, 373(6556):743–744, 2021.

[142] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489. PMLR, 2020.

[143] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *ISCA*, 2007.

[144] Jun Fang, Yanning Shen, Fuwei Li, and Hongbin Li. Prior support knowledge-aided sparse bayesian learning with partly erroneous support information. *arXiv preprint arXiv:1410.5055*, 2014.

[145] Yuan Fang, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. Object detection meets knowledge graphs.(2017). In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence: Melbourne, Australia, August 19*, volume 25, pages 1661–1667, 2017.

[146] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *European Symposium on Algorithms*, pages 182–194, 2010.

[147] Ana Lùcia D Franco, Henri Bourlès, Edson R De Pieri, and Herve Guillard. Robust nonlinear control associating robust feedback linearization and h/sub/spl infin//control. *IEEE transactions on automatic control*, 51(7):1200–1207, 2006.

[148] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008.

[149] Spencer Frei, Yuan Cao, and Quanquan Gu. Algorithm-dependent generalization bounds for overparameterized deep residual networks. *arXiv preprint arXiv:1910.02934*, 2019.

[150] Eric Friedman, Ali Ghodsi, and Christos-Alexandros Psomas. Strategyproof allocation of discrete jobs on multiple machines. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 529–546, 2014.

[151] Joel Friedman and Nathan Linial. On convex body chasing. *Discrete & Computational Geometry*, 9(3):293–321, 1993.

[152] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, pages 1607–1616, 2018.

[153] Lingwen Gan, Adam Wierman, Ufuk Topcu, Niangjun Chen, and Steven H Low. Real-time deferrable load control: handling the uncertainties of renewable generation. In *Proceedings of the fourth international conference on Future energy systems*, pages 113–124, 2013.

[154] Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In *Advances in Neural Information Processing Systems*, pages 975–985, 2018.

[155] Ruiqi Gao, Tianle Cai, Haochuan Li, Cho-Jui Hsieh, Liwei Wang, and Jason D Lee. Convergence of adversarial training in overparametrized neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[156] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[157] Evrard Garcelon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Matteo Pirotta. Conservative exploration in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1431–1441. PMLR, 2020.

[158] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[159] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[160] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[161] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[162] S. Gerchinovitz and T. Lattimore. Refined lower bounds for adversarial bandits. *NeurIPS*, 2016.

[163] Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. Pac-bayesian theory meets bayesian inference. *NeurIPS*, 2016.

[164] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

[165] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Nsdi*, volume 11, pages 24–24, 2011.

[166] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. *arXiv preprint arXiv:1904.02868*, 2019.

[167] Amirata Ghorbani and James Zou. Neuron shapley: Discovering the responsible neurons. *arXiv preprint arXiv:2002.09815*, 2020.

[168] Arnob Ghosh, Xingyu Zhou, and Ness Shroff. Provably efficient model-free constrained rl with linear function approximation. *arXiv preprint arXiv:2206.11889*, 2022.

[169] Gautam Goel, Naman Agarwal, Karan Singh, and Elad Hazan. Best of both worlds in online control: Competitive ratio and policy regret. *arXiv preprint arXiv:2211.11219*, 2022.

[170] Gautam Goel and Babak Hassibi. Competitive control, 2021.

[171] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. In *NeurIPS*, volume 32, 2019.

[172] Gautam Goel and Adam Wierman. An online algorithm for smoothed online convex optimization. *SIGMETRICS Perform. Eval. Rev.*, 47(2):6–8, December 2019.

[173] Gautam Goel and Adam Wierman. An online algorithm for smoothed regression and lqr control. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2504–2513, 2019.

[174] Íñigo Goiri, Kien Le, Md E Haque, Ryan Beauchea, Thu D Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2011.

[175] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[176] Andrea J Goldsmith and Soon-Ghee Chua. Variable-rate variable-power mqam for fading channels. *IEEE transactions on communications*, 45(10):1218–1230, 1997.

[177] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[178] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[179] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct. 2017.

[180] Ashutosh Deepak Gore and Abhay Karandikar. Link scheduling algorithms for wireless mesh networks. *IEEE Communications Surveys & Tutorials*, 13(2):258–273, 2010.

[181] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*, 2020.

[182] Brandon M Greenwell, Bradley C Boehmke, and Andrew J McCarthy. A simple and effective model-based variable importance measure. *arXiv preprint arXiv:1805.04755*, 2018.

[183] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

[184] Ziwei Guan, Kaiyi Ji, Donald J Bucci Jr, Timothy Y Hu, Joseph Palombo, Michael Liston, and Yingbin Liang. Robust stochastic bandit algorithms under probabilistic unbounded adversarial attack. In *AAAI*, 2020.

[185] Benjamin Guedj. A primer on pac-bayesian learning. *Proceedings of the 2nd congress of the Société Mathématique de France*, pages 391–414, 2019.

[186] Benjamin Guedj and John Shawe-Taylor. A primer on pac-bayesian learning. *International Conference on Machine Learning*, 2019.

[187] Vincent Le Guen, Yuan Yin, Jérémie Dona, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *arXiv preprint arXiv:2010.04456*, 2020.

[188] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. pages 1321–1330, 2017.

[189] Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose Blanchet, Peter W Glynn, and Yinyu Ye. Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321*, 2020.

[190] Moritz Hardt and Max Simchowitz. Convex optimization and approximation. `https://ee227c.github.io/notes/ee227c-notes.pdf`, 2018.

[191] Jian He, Zheng Xue, Di Wu, Dapeng Oliver Wu, and Yonggang Wen. Cbm: Online strategies on cost-aware buffer management for mobile video streaming. *IEEE Transactions on Multimedia*, 16(1):242–252, 2013.

[192] Jay Heo, Hae Beom Lee, Saehoon Kim, Juho Lee, Kwang Joon Kim, Eunho Yang, and Sung Ju Hwang. Uncertainty-aware attention for reliable interpretation and prediction. In *Advances in Neural Information Processing Systems*, pages 909–918, 2018.

[193] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

[194] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[195] Mingyi Hong and Zhi-Quan Luo. Signal processing and optimal resource allocation for the interference channel. In *Academic Press Library in Signal Processing*, volume 2, pages 409–469. 2014.

[196] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420, 2016.

[197] Longbo Huang, Xin Liu, and Xiaohong Hao. The power of online learning in stochastic network optimization. *SIGMETRICS Perform. Eval. Rev.*, 42(1):153–165, June 2014.

[198] Kelli D Humbird, J Luc Peterson, and Ryan G McClarren. Deep neural network initialization with decision trees. *IEEE transactions on neural networks and learning systems*, 30(5):1286–1295, 2018.

[199] Michael Husken and Christian Goerick. Fast learning for problem classes using knowledge based network initialization. In *IEEE IJCNN*, 2000.

[200] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political economy*, 87(2):293–314, 1979.

[201] Rob J Hyndman, David M Bashtannyk, and Gary K Grunwald. Estimating and visualizing conditional densities. *Journal of Computational and Graphical Statistics*, 5(4):315–336, 1996.

[202] Sungjin Im, Benjamin Moseley, Kamesh Munagala, and Kirk Pruhs. Dynamic weighted fairness with minimal disruptions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(1):1–18, 2020.

[203] The Alan Turing Institute. Physics-informed machine learning, 2020. `https://www.turing.ac.uk/research/theory-and-method-challenge-fortnights/physics-informed-machine-learning`.

[204] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[205] Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. In *AISTATS*, 2015.

[206] Stefanus Jasin and Sunil Kumar. A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Mathematics of Operations Research*, 37(2):313–345, 2012.

[207] Guangda Ji and Zhanxing Zhu. Knowledge distillation in wide neural networks: Risk bound, data efficiency and imperfect teacher. *arXiv preprint arXiv:2010.10090*, 2020.

[208] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*, 2019.

[209] Chenhan Jiang, Hang Xu, Xiaodan Liang, and Liang Lin. Hybrid knowledge routed modules for large-scale object detection. In *Advances in Neural Information Processing Systems*, pages 1552–1563, 2018.

[210] Jiashuo Jiang, Xiaocheng Li, and Jiawei Zhang. Online stochastic optimization with wasserstein based non-stationarity. *arXiv preprint arXiv:2012.06961*, 2020.

[211] Ruiwei Jiang, Jianhui Wang, Muhong Zhang, and Yongpei Guan. Two-stage minimax regret robust unit commitment. *IEEE Transactions on Power Systems*, 28(3):2271–2282, 2013.

[212] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

[213] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Trans. Netw.*, 21(6):1785–1798, December 2013.

[214] Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Xiaojin Zhu. Adversarial attacks on stochastic bandits. In *NIPS*, 2018.

[215] Leonid Vasilevich Kantorovich and SG Rubinshtein. On a space of totally additive functions. *Vestnik of the St. Petersburg University: Mathematics*, 13(7):52–59, 1958.

[216] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[217] Sumeet Katariya, Branislav Kveton, Zheng Wen, and Vamsi K Potluru. Conservative exploration using interleaving. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 954–963. PMLR, 2019.

[218] IS Khalil, JC Doyle, and K Glover. *Robust and optimal control*. Prentice hall, 1996.

[219] Imran Khan. Temporal carbon intensity analysis: renewable versus fossil fuel dominated electricity systems. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, 41(3):309–323, 2019.

[220] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*, 2017.

[221] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pages 5562–5571. PMLR, 2021.

[222] Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust bayesian optimization. In *AISTATS*, 2020.

[223] Johannes Kirschner and Andreas Krause. Stochastic bandits with context distributions. In *NeurIPS*, 2019.

[224] Bahare Kiumarsi, Kyriakos G Vamvoudakis, Hamidreza Modares, and Frank L Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, 29(6):2042–2062, 2017.

[225] Aldebaro Klautau, Pedro Batista, Nuria González-Prelcic, Yuyang Wang, and Robert W Heath. 5g mimo data for machine learning: Application to beam-selection using deep learning. In *ITA*, pages 1–9, 2018.

[226] Erich Kobler, Alexander Effland, Karl Kunisch, and Thomas Pock. Total deep variation: A stable regularizer for inverse problems. *arXiv preprint arXiv:2006.08789*, 2020.

[227] Zico Kolter, David Duvenaud, and Matt Johnson. Deep implicit layers, `http://implicit-layers-tutorial.org/`.

[228] Weiwei Kong, Christopher Liaw, Aranyak Mehta, and D. Sivakumar. A new dog learns old tricks: RL finds classic optimization algorithms. In *ICLR*, 2019.

[229] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations Research & Management Science in the Age of Analytics*, pages 130–166. 2019.

[230] Gakuto Kurata, Bing Xiang, and Bowen Zhou. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, 2016.

[231] Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. An axiomatic theory of fairness in network resource allocation. In *INFOCOM*, 2010.

[232] Hoang Le, Andrew Kang, Yisong Yue, and Peter Carr. Smooth imitation learning for online sequence prediction. In *International Conference on Machine Learning*, pages 680–688. PMLR, 2016.

[233] Hoang M. Le, Andrew Kang, Yisong Yue, and Peter Carr. Smooth imitation learning for online sequence prediction. In *ICML*, 2016.

[234] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

[235] Zachary J Lee, Tongxin Li, and Steven H Low. Acn-data: Analysis and applications of an open ev charging dataset. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, pages 139–149, 2019.

[236] H. Lei, A. Tewari, and S. Murphy. An actor-critic contextual bandit algorithm for personalized interventions using mobile devices. *NeurIPS*, 2014.

[237] Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. Distance-sensitive offline reinforcement learning. *arXiv preprint arXiv:2205.11027*, 2022.

[238] Ke Li and Jitendra Malik. Learning to optimize. In *ICLR*, 2017.

[239] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

[240] Mo Li, Timothy M Smith, Yi Yang, and Elizabeth J Wilson. Marginal emission factors considering renewables: A case study of the us midcontinent independent system operator (miso) system. *Environmental science & technology*, 51(19):11215–11223, 2017.

[241] Pengfei Li, Jianyi Yang, and Shaolei Ren. Expert-calibrated learning for online optimization with switching costs. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(2), Jun 2022.

[242] Pengfei Li, Jianyi Yang, and Shaolei Ren. Expert-calibrated learning for online optimization with switching costs. In *SIGMETRICS*, 2022.

[243] Pengfei Li, Jianyi Yang, and Shaolei Ren. Learning for edge-weighted online bipartite matching with robustness guarantees. *ICML*, 2023.

[244] Pengfei Li, Jianyi Yang, and Shaolei Ren. Robustified learning for online optimization with memory costs. *INDOCOM*, 2023.

[245] Tongxin Li. *Learning-Augmented Control and Decision-Making: Theory and Applications in Smart Grids*. PhD thesis, California Institute of Technology, 2023.

[246] Tongxin Li, Bo Sun, Yue Chen, Zixin Ye, Steven H Low, and Adam Wierman. Learning-based predictive control via real-time aggregate flexibility. *IEEE Transactions on Smart Grid*, 12(6):4897–4913, 2021.

[247] Tongxin Li, Ruixiao Yang, Guannan Qu, Yiheng Lin, Steven Low, and Adam Wierman. Equipping black-box policies with model-based advice for stable nonlinear control. In *https://arxiv.org/abs/2206.01341*, 2022.

[248] Tongxin Li, Ruixiao Yang, Guannan Qu, Guanya Shi, Chenkai Yu, Adam Wierman, and Steven Low. Robustness and consistency in linear quadratic control with untrusted predictions. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(1), feb 2022.

[249] Yingying Li, Xin Chen, and Na Li. Online optimal control with linear dynamics and predictions: Algorithms and regret analysis. *Advances in Neural Information Processing Systems*, 32, 2019.

[250] Yingying Li, Subhro Das, Jeff Shamma, and Na Li. Safe adaptive learning-based control for constrained linear quadratic regulators with regret guarantees. *arXiv preprint arXiv:2111.00411*, 2021.

[251] Yingying Li and Na Li. Leveraging predictions in smoothed online convex optimization via gradient-based algorithms. In *NeurIPS*, volume 33, 2020.

[252] Yingying Li and Na Li. Leveraging predictions in smoothed online convex optimization via gradient-based algorithms. *arXiv preprint arXiv:2011.12539*, 2020.

[253] Yingying Li, James A Preiss, Na Li, Yiheng Lin, Adam Wierman, and Jeff Shamma. Online switching control with stability and regret guarantees. *arXiv preprint arXiv:2301.08445*, 2023.

[254] Yingying Li, Guannan Qu, and Na Li. Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *IEEE Transactions on Automatic Control*, 2020.

[255] Yuelong Li, Mohammad Tofighi, Vishal Monga, and Yonina C Eldar. An algorithm unrolling approach to deep image deblurring. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7675–7679. IEEE, 2019.

[256] Enming Liang, Minghua Chen, and Steven H. Low. Low complexity homeomorphic projection to ensure neural-network solution feasibility for optimization over (non-)convex set. In *ICML*, 2023.

[257] F. Liang, C. Shen, W. Yu, and F. Wu. Towards optimal power control via ensembling deep neural networks. *IEEE Transactions on Communications*, 68(3):1760–1776, 2020.

[258] Fei Liang, Cong Shen, Wei Yu, and Feng Wu. Towards optimal power control via ensembling deep neural networks. *IEEE Transactions on Communications*, 68(3):1760–1776, 2019.

[259] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolutions. In *Advances in Neural Information Processing Systems*, pages 1853–1863, 2018.

[260] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[261] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew. Online algorithms for geographical load balancing. In *IGCC*, 2012.

[262] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *INFOCOM*, 2011.

[263] Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan LH Andrew. Online algorithms for geographical load balancing. In *2012 international green computing conference (IGCC)*, pages 1–10, 2012.

[264] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391, 2012.

[265] Qiulin Lin, Yanfang Mo, Junyan Su, and Minghua Chen. Competitive online optimization with multiple inventories: A divide-and-conquer approach. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(2):1–28, 2022.

[266] Qiulin Lin, Hanling Yi, John Pang, Minghua Chen, Adam Wierman, Michael Honig, and Yuanzhang Xiao. Competitive online optimization under inventory constraints. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(1):1–28, 2019.

[267] Yiheng Lin, Judy Gan, Guannan Qu, Yash Kanoria, and Adam Wierman. Decentralized online convex optimization in networked systems. In *International Conference on Machine Learning*, pages 13356–13393. PMLR, 2022.

[268] Yiheng Lin, Yang Hu, Guannan Qu, Tongxin Li, and Adam Wierman. Bounded-regret mpc via perturbation analysis: Prediction error, constraints, and nonlinearity. *arXiv preprint arXiv:2210.12312*, 2022.

[269] Yiheng Lin, Yang Hu, Guanya Shi, Haoyuan Sun, Guannan Qu, and Adam Wierman. Perturbation-based regret analysis of predictive control in linear time varying systems. *Advances in Neural Information Processing Systems*, 34:5174–5185, 2021.

[270] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[271] Fang Liu and Ness Shroff. Data poisoning attacks on stochastic bandits. In *ICML*, 2019.

[272] Risheng Liu, Shichao Cheng, Long Ma, Xin Fan, and Zhongxuan Luo. Deep proximal unrolling: Algorithmic framework, convergence analysis and applications. *IEEE Transactions on Image Processing*, 28(10):5013–5026, 2019.

[273] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

[274] Xin Liu, Bin Li, Pengyi Shi, and Lei Ying. An efficient pessimistic-optimistic algorithm for stochastic linear bandits with general constraints. *Advances in Neural Information Processing Systems*, 34:24075–24086, 2021.

[275] Zhenhua Liu, Iris Liu, Steven Low, and Adam Wierman. Pricing data center demand response. In *SIGMETRICS*, 2014.

[276] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation*, 70(10):770–791, 2013.

[277] Alfonso Lobos, Paul Grigas, and Zheng Wen. Joint online learning and decision-making via dual mirror descent. *arXiv preprint arXiv:2104.09750*, 2021.

[278] Jihao Long, Jiequn Han, et al. An l^2 analysis of reinforcement learning in high dimensions with kernel and neural network approximation. *arXiv preprint arXiv:2104.07794*, 2021.

[279] Lu Lu, Xuhui Meng, Zhiping Mao, and George E Karniadakis. Deepxde: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

[280] T. Lu, D. Pál, and M. Pál. Contextual multi-armed bandits. *AISTATS*, 2010.

[281] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[282] Jerry Luo, Cosmin Paduraru, Octavian Voicu, Yuri Chervonyi, Scott Munns, Jerry Li, Crystal Qian, Praneet Dutta, Jared Quincy Davis, Ningjia Wu, et al. Controlling commercial cooling systems using reinforcement learning. *arXiv preprint arXiv:2211.07357*, 2022.

[283] Thodoris Lykouris, Max Simchowitz, Alex Slivkins, and Wen Sun. Corruption-robust exploration in episodic reinforcement learning. In *Conference on Learning Theory*, pages 3242–3245. PMLR, 2021.

[284] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4), July 2021.

[285] Mohamed Maher and Sherif Sakr. Smartml: A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. In *EDBT: 22nd International Conference on Extending Database Technology*, 2019.

[286] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

[287] Jayanta Mandi, Vıctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: Through the lens of learning to rank. In *International Conference on Machine Learning*, pages 14935–14947. PMLR, 2022.

[288] Joseph Marino, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue. Iterative amortized policy optimization. *arXiv preprint arXiv:2010.10670*, 2020.

[289] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016.

[290] Pascal Massart, Élodie Nédélec, et al. Risk bounds for statistical learning. *The Annals of Statistics*, 34(5):2326–2366, 2006.

[291] Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory*, pages 3–17, 2016.

[292] Aranyak Mehta et al. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

[293] Shahar Mendelson and Roman Vershynin. Entropy and the combinatorial dimension. *Inventiones mathematicae*, 152(1):37–55, 2003.

[294] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In *Workshop on Automatic Machine Learning*, pages 58–65, 2016.

[295] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pages 4556–4565, 2019.

[296] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[297] Moein Moeini-Aghtaie, Payman Dehghanian, Mahmud Fotuhi-Firuzabad, and Ali Abbaspour. Multiagent genetic algorithm: an online probabilistic view on economic dispatch of energy hubs constrained by wind availability. *IEEE Transactions on sustainable energy*, 5(2):699–708, 2013.

[298] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[299] Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.

[300] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[301] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Information bottleneck learning using privileged information for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[302] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *IEEE Big Data*, pages 36–45, 2018.

[303] Harikrishna Narasimhan, Andrew Cotter, Yichen Zhou, Serena Wang, and Wenshuo Guo. Approximate heavily-constrained learning with lagrange multiplier models. *Advances in Neural Information Processing Systems*, 33:8693–8703, 2020.

[304] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[305] Gergely Neu and Julia Olkhovskaya. Efficient and robust algorithms for adversarial linear contextual bandits. In *COLT*, 2020.

[306] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *ICLR*, 2018.

[307] Thanh Nguyen, Sunil Gupta, Huong Ha, Santu Rana, and Svetha Venkatesh. Distributionally robust bayesian quadrature optimization. In *AISTATS*, 2020.

[308] Atsushi Nitanda, Geoffrey Chinot, and Taiji Suzuki. Gradient descent can learn less over-parameterized two-layer neural networks on classification problems. *arXiv preprint arXiv:1905.09870*, 2019.

[309] Di Niu, Hong Xu, Baochun Li, and Shuqiao Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *2012 Proceedings IEEE INFOCOM*, pages 460–468. IEEE, 2012.

[310] California Independent System Operator. Calfornia renewable datasets. `https://www.caiso.com/Pages/default.aspx`, 2023.

[311] Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

[312] Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. *Advances in Neural Information Processing Systems*, 27, 2014.

[313] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105, 2020.

[314] Aldo Pacchiano, Mohammad Ghavamzadeh, Peter Bartlett, and Heinrich Jiang. Stochastic bandits with linear constraints. In *International conference on artificial intelligence and statistics*, pages 2827–2835. PMLR, 2021.

[315] D. P. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Trans. Automatic Control*, 52(12):2254–2269, Dec. 2007.

[316] Weici Pan, Guanya Shi, Yiheng Lin, and Adam Wierman. Online optimization with feedback delay and nonlinear switching cost. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(1), feb 2022.

[317] Daniel Paulin. Concentration inequalities for markov chains by marton couplings and spectral methods. *Electronic Journal of Probability*, 20:1–32, 2015.

[318] Julius Pfrommer, Clemens Zimmerling, Jinzhao Liu, Luise Kärger, Frank Henning, and Jürgen Beyerer. Optimisation of manufacturing process parameters using deep neural networks as surrogate models. *Procedia CiRP*, 72:426–431, 2018.

[319] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pages 5142–5151. PMLR, 2019.

[320] Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Semi-supervised learning using an unsupervised atlas. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 565–580, 2014.

[321] Marios M Polycarpou and Petros A Ioannou. A robust adaptive nonlinear control design. In *1993 American control conference*, pages 1365–1369. IEEE, 1993.

[322] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. *Neurips*, 2018.

[323] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[324] Chengrun Qiu, Yang Hu, Yan Chen, and Bing Zeng. Lyapunov optimization for energy harvesting wireless sensor communications. *IEEE Internet of Things Journal*, 5(3):1947–1956, 2018.

[325] Guannan Qu, Yuanyuan Shi, Sahin Lale, Anima Anandkumar, and Adam Wierman. Stable online control of linear time-varying systems. In *Learning for Dynamics and Control*, pages 742–753. PMLR, 2021.

[326] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2022.

[327] Arman Rahbar, Ashkan Panahi, Chiranjib Bhattacharyya, Devdatt Dubhashi, and Morteza Haghir Chehreghani. On the unreasonable effectiveness of knowledge distillation: Analysis in the kernel regime. *arXiv preprint arXiv:2003.13438*, 2020.

[328] Akshara Rai, Rika Antonova, Franziska Meier, and Christopher G Atkeson. Using simulation to improve sample-efficiency of bayesian optimization for bipedal robots. *Journal of Machine Learning Research*, 20:49–1, 2019.

[329] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[330] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[331] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *COLT*, 2013.

[332] Connie Loggia Ramsey and John J Grefenstette. Case-based initialization of genetic algorithms. In *ICGA*, pages 84–91, 1993.

[333] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[334] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, volume 18, pages 1527–1535, 2018.

[335] Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International Conference on Machine Learning*, pages 8116–8126, 2020.

[336] Joshua Robinson, Stefanie Jegelka, and Suvrit Sra. Strength from weakness: Fast learning using weak supervision. In *International Conference on Machine Learning*, pages 8127–8136, 2020.

[337] E Ruffio, D Saury, D Petit, and M Girault. Tutorial 2: Zero-order optimization algorithms. *Eurotherm School METTI*, 2011.

[338] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[339] Daan Rutten, Nico Christianson, Debankur Mukherjee, and Adam Wierman. Online optimization with untrusted predictions. *CoRR*, abs/2202.03519, 2022.

[340] Daan Rutten, Nico Christianson, Debankur Mukherjee, and Adam Wierman. Online optimization with untrusted predictions. *arXiv preprint arXiv:2202.03519*, 2022.

[341] Omid Sadeghi and Maryam Fazel. Online continuous dr-submodular maximization with long-term budget constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 4410–4419. PMLR, 2020.

[342] Mohammad A Salahuddin, Ala Al-Fuqaha, and Mohsen Guizani. Reinforcement learning for resource provisioning in the vehicular cloud. *IEEE Wireless Communications*, 23(4):128–135, 2016.

[343] Shahab Sanayei and Aria Nosratinia. Antenna selection in mimo systems. *IEEE Communications magazine*, 42(10):68–73, 2004.

[344] Vidit Saxena, Joakim Jaldén, Joseph E. Gonzalez, Mats Bengtsson, Hugo Tullberg, and Ion Stoica. Contextual multi-armed bandits for link adaptation in cellular networks. In *Workshop on Network Meets AI & ML (NetAI)*, 2019.

[345] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[346] Luciano Serafini and Artur d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

[347] Sanket Shah, Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. Decision-focused learning without decision-making: Learning locally optimized decision losses. In *Advances in Neural Information Processing Systems*, 2022.

[348] Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 205–218, 2020.

[349] Mohammad Shahrad, Cristian Klein, Liang Zheng, Mung Chiang, Erik Elmroth, and David Wentzlaff. Incentivizing self-capping to increase cloud utilization. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 52–65, 2017.

[350] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[351] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[352] Viktoriia Sharmanska, Novi Quadrianto, and Christoph H. Lampert. Learning to rank using privileged information. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

[353] Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.

[354] Shiwen Shen, Simon X Han, Denise R Aberle, Alex A Bui, and William Hsu. An interpretable deep hierarchical semantic convolutional neural network for lung nodule malignancy classification. *Expert systems with applications*, 128:84–95, 2019.

[355] Guanya Shi. Competitive control via online optimization with memory, delayed feedback, and inexact predictions. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, 2021.

[356] Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online optimization with memory and competitive control. *Advances in Neural Information Processing Systems*, 33:20636–20647, 2020.

[357] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[358] Yuanyuan Shi, Guannan Qu, Steven Low, Anima Anandkumar, and Adam Wierman. Stability constrained reinforcement learning for real-time voltage control. In *2022 American Control Conference (ACC)*, pages 2715–2721. IEEE, 2022.

[359] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence and generalization of physics informed neural networks. *arXiv preprint arXiv:2004.01806*, 2020.

[360] Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. Attentive recurrent comparators. In *International Conference on Machine Learning*, pages 3173–3181, 2017.

[361] Nian Si, Fan Zhang, Zhengyuan Zhou, and Jose Blanchet. Distributional robust batch contextual bandits. *arXiv preprint arXiv:2006.05630*, 2020.

[362] Nian Si, Fan Zhang, Zhengyuan Zhou, and Jose Blanchet. Distributionally robust policy evaluation and learning in offline contextual bandits. In *ICML*, 2020.

[363] Mattia Silvestri, Michele Lombardi, and Michela Milano. Injecting domain knowledge in neural networks: a controlled experiment on a constrained problem. *arXiv preprint arXiv:2002.10742*, 2020.

[364] Sayanan Sivaraman and Mohan Manubhai Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):267–276, 2010.

[365] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.

[366] Chaehwan Song, Ali Ramezani-Kebrya, Thomas Pethick, Armin Eftekhari, and Volkan Cevher. Subquadratic overparameterization for shallow neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[367] Pablo Sprechmann, Alexander M Bronstein, and Guillermo Sapiro. Learning efficient sparse and low rank models. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1821–1833, 2015.

[368] Suhas Sreehari, S Venkat Venkatakrishnan, Brendt Wohlberg, Gregery T Buzzard, Lawrence F Drummy, Jeffrey P Simmons, and Charles A Bouman. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Transactions on Computational Imaging*, 2(4):408–423, 2016.

[369] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. *ICML*, 2010.

[370] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[371] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[372] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[373] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hajiesmaili, Adam Wierman, and Danny HK Tsang. Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging. *ACM on Measurement and Analysis of Computing Systems (POMACS)*, 4(3), 2021.

[374] Chenxi Sun, Tongxin Li, and Xiaoying Tang. Data-driven electric vehicle charging station placement for incentivizing potential demand. In *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 27–32. IEEE, 2021.

[375] 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 Transactions on Signal Processing*, 66(20):5438–5453, 2018.

[376] Qihang Sun, Shaolei Ren, Chuan Wu, and Zongpeng Li. An online incentive mechanism for emergency demand response in geo-distributed colocation data centers. In *eEnergy*, 2016.

[377] Ruoyu Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

[378] Wen Sun, Debadeepta Dey, and Ashish Kapoor. Safety-aware algorithms for adversarial contextual bandit. In *ICML*, 2017.

[379] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018.

[380] V. Syrgkanis, A. Krishnamurthy, and R. Schapire. Efficient algorithms for adversarial contextual learning. *ICML*, 2016.

[381] Kalyan T Talluri, Garrett Van Ryzin, and Garrett Van Ryzin. *The theory and practice of revenue management*, volume 1. Springer, 2004.

[382] W. Tang, S. Bi, and Y. Zhang. Online coordinated charging decision algorithm for electric vehicles without future information. *IEEE Trans. Smart Grid*, 5:2810–2824, May 2014.

[383] Garrett Thomas. Markov decision processes. 2007.

[384] Ryan Tibshirani. Projected gradient descent. `https://www.stat.cmu.edu/~ryantibs/convexopt-F18/scribes/Lecture_23.pdf`, 2018.

[385] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[386] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2018.

[387] Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jaques E Slotine. Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control*, 52:135–169, 2021.

[388] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. *UAI*, 2013.

[389] Jan N Van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *ACM SIGKDD*, pages 2367–2376, 2018.

[390] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16(1):2023–2049, 2015.

[391] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

[392] Sharan Vaswani, Lin Yang, and Csaba Szepesvári. Near-optimal sample complexity bounds for constrained mdps. *Advances in Neural Information Processing Systems*, 35:3110–3122, 2022.

[393] Abhinav Verma, Hoang Le, Yisong Yue, and Swarat Chaudhuri. Imitation-projected programmatic reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[394] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019.

[395] VMware. Distributed power management concepts and use, `http://www.vmware.com/files/pdf/Distributed-Power-Management-vSphere.pdf`.

[396] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394*, 2019.

[397] Martin Wainwright. Lecture of dudley's entropy integral. `https://people.eecs.berkeley.edu/~wainwrig/stat241b/lec20.pdf`.

[398] H. Wang, Q. Wu, and H. Wang. Learning hidden features for contextual bandits. *CIKM*, 2016.

[399] Kai Wang, Sanket Shah, Haipeng Chen, Andrew Perrault, Finale Doshi-Velez, and Milind Tambe. Learning MDPs from features: Predict-then-optimize for sequential decision making by reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[400] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3):1–34, 2020.

[401] Larry Wasserman. Density estimation. `https://www.stat.cmu.edu/~larry/=sml/densityestimation.pdf`.

[402] Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *NeurIPS*, 2020.

[403] Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning*, 2020.

[404] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. In *Advances in Neural Information Processing Systems*, pages 9712–9724, 2019.

[405] Xiaohan Wei, Hao Yu, and Michael J Neely. Online primal-dual mirror descent under stochastic constraints. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–36, 2020.

[406] 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 *International Conference on Machine Learning*, pages 3751–3760, 2017.

[407] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

[408] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. 2020.

[409] William Wong, Praneet Dutta, Octavian Voicu, Yuri Chervonyi, Cosmin Paduraru, and Jerry Luo. Optimizing industrial hvac systems with hierarchical reinforcement learning. *arXiv preprint arXiv:2209.08112*, 2022.

[410] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[411] Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. Conservative bandits. In *ICML*, 2016.

[412] Yihong Wu. Packing, covering, and consequences on minimax risk. `http://www.stat.yale.edu/~yw562/teaching/598/lec14.pdf`, 2016.

[413] Yiyan Wu and William Y Zou. Orthogonal frequency division multiplexing: A multi-carrier modulation scheme. *IEEE Transactions on Consumer Electronics*, 41(3):392–399, 1995.

[414] Jiaqi Xiang, Qingdong Li, Xiwang Dong, and Zhang Ren. Continuous control with deep reinforcement learning for mobile robot navigation. In *2019 Chinese Automation Congress (CAC)*, pages 1501–1506. IEEE, 2019.

[415] Chenjun Xiao, Bo Dai, Jincheng Mei, Oscar A Ramirez, Ramki Gummadi, Chris Harris, and Dale Schuurmans. Understanding and leveraging overparameterization in recursive value estimation. In *International Conference on Learning Representations*, 2021.

[416] Jie Xu, Lixing Chen, and Shaolei Ren. Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking*, 3(3):361–373, 2017.

[417] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning*, pages 5502–5511, 2018.

[418] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[419] Jianyi Yang and Shaolei Ren. Robust bandit learning with imperfect context. *arXiv preprint arXiv:2102.05018*, 2021.

[420] Jianyi Yang and Shaolei Ren. Informed learning by wide neural networks: Convergence, generalization and sampling complexity. In *International Conference on Machine Learning*, pages 25198–25240. PMLR, 2022.

[421] Jianyi Yang and Shaolei Ren. Learning-assisted algorithm unrolling for online optimization with budget constraints. *AAAI*, 2022.

[422] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020.

[423] Yunchang Yang, Tianhao Wu, Han Zhong, Evrard Garcelon, Matteo Pirotta, Alessandro Lazaric, Liwei Wang, and Simon Shaolei Du. A reduction-based framework for conservative bandits and reinforcement learning. In *International Conference on Learning Representations*, 2022.

[424] Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I Jordan. On function approximation in reinforcement learning: optimism in the face of large state spaces. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 13903–13916, 2020.

[425] Gal Yona, Amirata Ghorbani, and James Zou. Who's responsible? jointly quantifying the contribution of the learning algorithm and training data. *arXiv preprint arXiv:1910.04214*, 2019.

[426] Chenkai Yu, Guanya Shi, Soon-Jo Chung, Yisong Yue, and Adam Wierman. The power of predictions in online control. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[427] Chenkai Yu, Guanya Shi, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Competitive control with delayed imperfect information. In *2022 American Control Conference (ACC)*, pages 2604–2610. IEEE, 2022.

[428] Hao Yu and Michael J Neely. Learning-aided optimization for energy-harvesting devices with outdated state information. *IEEE/ACM Transactions on Networking*, 27(4):1501–1514, 2019.

[429] Hao Yu and Michael J. Neely. A low complexity algorithm with $\mathcal{O}(\sqrt{T})$ regret and $\mathcal{O}(1)$ constraint violations for online convex optimization with long term constraints. *Journal of Machine Learning Research*, 21(1):1–24, 2020.

[430] Francesco Zanini, David Atienza, Giovanni De Micheli, and Stephen P Boyd. Online convex optimization-based algorithm for thermal management of mpsocs. In *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, pages 203–208, 2010.

[431] Ali Zeynali, Bo Sun, Mohammad Hajiesmaili, and Adam Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *AAAI*, 2021.

[432] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[433] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[434] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting cnns via decision trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6261–6270, 2019.

[435] Runyu Zhang, Yingying Li, and Na Li. On the regret analysis of online lqr control with predictions. In *2021 American Control Conference (ACC)*, pages 697–703. IEEE, 2021.

[436] Yabin Zhang, Hui Tang, and Kui Jia. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In *European conference on computer vision*, pages 233–248, 2018.

[437] Zijun Zhang, Zongpeng Li, and Chuan Wu. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–26, 2017.

[438] Hengyu Zhao, Yubo Zhang, Pingfan Meng, Hui Shi, Li Erran Li, Tiancheng Lou, and Jishen Zhao. Safety score: A quantitative approach to guiding safety-aware autonomous vehicle computing system design. In *IEEE Intelligent Vehicles Symposium*, 2020.

[439] Tianyu Zhao, Xiang Pan, Minghua Chen, and Steven Low. Ensuring dnn solution feasibility for optimization problems with linear constraints. In *The Eleventh International Conference on Learning Representations*.

[440] Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted mdps with feature mapping. In *International Conference on Machine Learning*, pages 12793–12802. PMLR, 2021.

[441] Xingyu Zhou, Ness Shroff, and Adam Wierman. Asymptotically optimal load balancing in large-scale heterogeneous systems with multiple dispatchers. *ACM SIGMETRICS Performance Evaluation Review*, 48(3):57–58, 2021.

[442] Xingyu Zhou, Jian Tan, and Ness Shroff. Flexible load balancing with multi-dimensional state-space collapse: Throughput and heavy-traffic delay optimality. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):10–11, 2019.

[443] Xinyang Zhou, Masoud Farivar, Zhiyuan Liu, Lijun Chen, and Steven H Low. Reverse and forward engineering of local voltage control in distribution networks. *IEEE Transactions on Automatic Control*, 66(3):1116–1128, 2020.

[444] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*, pages 566–576, 2008.

[445] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[446] Feiyun Zhu, Jun Guo, Ruoyu Li, and Junzhou Huang. Robust actor-critic contextual bandit for mobile health (mhealth) interventions. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 492–501, 2018.

[447] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

[448] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

[449] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. In *NeurIPS*, 2019.