# Robust, Deep, and Reinforcement Learning for Management of Communication and Power Networks

**A DISSERTATION**
**SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL**
**OF THE UNIVERSITY OF MINNESOTA**
**BY**

**Alireza Sadeghi**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**
**FOR THE DEGREE OF**
**DOCTOR OF PHILOSOPHY**

**Prof. Georgios B. Giannakis, Advisor**

**August, 2021**

# Acknowledgments

First and foremost, my deepest gratitude goes to my advisor Prof. Georgios B. Giannakis for providing me with the opportunity to be a part of SPiNCOM research group, as well as ECE/CS graduate program of University of Minnesota. He has helped me in developing clear and scientific thought and expression, and without his support, the completion of this PhD Thesis would not have been possible. His vision and enthusiasm about innovative research and beyond, his broad and deep knowledge, and his unbounded energy have constantly been a true inspiration for me. Because of him, I have been very fortunate to be always surrounded by other wonderful students and colleagues.

Due thanks go to Profs. Mostafa Kaveh, Zhi-Li Zhang, and Mehmet Akçakaya for agreeing to serve on my committee as well as all their valuable comments and feedback on my research and thesis. Thanks also go to other professors in the Departments of Electrical Engineering and Computer Science whose graduate level courses helped me build the necessary background to embark on this journey.

During my PhD studies, I had the opportunity to collaborate with several excellent individuals, and I have greatly benefited from their critical thinking, brilliant ideas, and vision. Particularly, I would like to express my greatest gratitude to my friend and collaborator Dr. Fatemeh Sheikholeslami who was patient enough to train me in the first couple of difficult years at UMN, and Prof. Gang Wang, with whom I was fortunate to collaborate with and learn from. I greatly benefited from his vision, ideas, and insights. I would also like to extend my due credit and warmest thanks to Profs. Antonio G. Marques (King Juan Carlos University) for his valuable contribution and insights to our fruitful collaborations. The material in this thesis has also benefited from collaborating with Qiuling Yang.

I would like to extend my gratitude to current and former members of the SPiNCOM group at UMN: Dr. Siavash Ghavami, Dr. Brian Baingana, Dr. Yanning Shen, Dr. Dimitris Berberidis,

Dr. Jia Chen, Dr. Meng Ma, Prof. Tianyi Chen, Dr. Vassilis Ioannidis, Dr. Georgios V. Karanikolas, Dr. Donghoon Lee, Dr. Athanasios Nikolakopoulos, Dr. Panagiotis Traganitis, Prof. Daniel Romero, Dr. Liang Zhang, and Seth Barrash. I am truly grateful to these people for their continuous help. I would also wish to acknowledge the grants that support financially our research.

I am not forgetting my friends, some of which I have already mentioned above, both the ones here in Minneapolis, and my old-term friends that are far away, in particular: Danial Panahandeh-Shahraki, Siavash Ghavami, Amirhossein Hosseini, Mojtaba Kadkhodaei Elyaderani, Mohsen Mahmoodi, Fazel Zare Bidoky, Abolfazl Zamanpour Kiasari, Hamed Mosavat, Mehrdad Damsaz, Movahed Jamshidi, Javad Ansari, Meysam Mohajer, Qiuling Yang and Ali Ghaffarpour.

Finally, gift of a family is incomparable. They are the source of my strength, motivation, and sustenance. A special feeling of gratitude to my loving mother Fattaneh, whose heart I know is sick of my long distance. Special thanks also goes to my lovely sister Maedeh, who never left my side. I am eternally grateful to my mother, who encouraged me to pursue academic endeavor. Without you, I would not be standing here today.

*Alireza Sadeghi, Minneapolis, April, 2021.*

# Dedication

This dissertation is dedicated to my mother Fattaneh, and sister Maedeh for their unconditional love and support.

## Abstract

Data-driven machine learning advances have effectively handled a wide spectrum of application domains. However, formidable challenges remain, especially for managing and optimizing the next-generation complex cyber-physical systems, autonomously-driving cars, and self-surgical systems, which welcome ground-breaking control, monitoring, and decision making that can guarantee robustness, scalability, and situational awareness. In this context, the present thesis first develops principled methods to robustify learning models against distributional uncertainties and adversarial data. The developed framework is particularly attractive when training and testing data are drawn from mismatched distributions. By leveraging the Wasserstein distance, the novel approaches minimize the worst-case expected loss over a prescribed family of data distributions. Building on this robust framework, the thesis next introduces a robust semi-supervised learning approach over networked data whose interdependencies are captured by graphs. Subsequently, the thesis contributes machine learning tools for next-generation wired and wireless networks, through the design of intelligent caching modules using deep reinforcement learning. These modules are equipped with storage devices, and can thus prefetch popular contents (reusable information) during off-peak traffic hours, and service them to the network edge at peak traffic instances. Finally, the thesis contributes to the management and control of power networks, and specifically distribution grids with high penetration of renewable sources and demand response programs. Reactive power is optimally allocated to both utility-owned control devices (e.g., capacitor banks), as well as smart inverters of distributed generation units with cyber-capabilities. The resultant novel dynamic control algorithms are scalable and adaptive to real-time changes of renewable generation and load consumption. To further enhance the situational awareness in power networks, the thesis further contributes robust power system state estimation solvers.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation and Context

Nowadays, overcoming emerging engineering challenges in cyber-physical systems requires successfully performing various learning tasks. At the same time, although machine learning algorithms have been successful in dealing with standard learning tasks with the sheer volume and high dimensionality of data, they are defenseless against adversarially manipulated input data, and sensitive to dynamically changing environments. Recent advancements in non-linear function approximation, optimal transport theory, and minimax optimization techniques provide a timely opportunity to transform machine learning algorithms to a scalable, reliable, secure, and safe technology to control and manage complex cyber-physical systems. In this context, the present thesis aspires to develop principled methods incorporating *scalability* along with *robustness* in machine learning paradigms, having as ultimate goal to enhance their prediction, control, and tracking performance in unknown, dynamic, and possibly adversarial settings.

By putting forth an analytical and algorithmic framework for learning and inferring from data, with applications in management of cyber-physical systems, this thesis will develop a suit of machine learning based tools to optimally control these systems. our vision is to effect technical advances in function approximation, machine learning, optimal transport theory, and optimization to develop state-of-the-art algorithms for managing cyber-physical systems. The central goal is to theoretically, algorithmically, and numerically developed online, robust, and scalable algorithms. Specifically, the following research thrusts will be pursued:

**(T1)** Robust supervised learning under distributional uncertainties due to adversaries;

**(T2)** Distributionally robust semi-supervised learning and inference over graphs; and,

**(T3)** Deep- and reinforced-learning for network resource management;

**(T4)** Data-driven, reinforced, and robust learning approaches for smart power grid.

## 1.2 Learning Robust against distributional uncertainties and adversarial data

It has been recently recognized that learning function models is vulnerable to adversarially manipulated input data, which discourages their use in safety-critical applications. In addition, learning algorithms often rely on the premise that training and testing data are drawn from the *same* distribution, which may not hold in practice. Major efforts have been devoted to robustifying learning models to uncertainties arising from e.g., distributional mismatch using data pre-processing techniques such as compression, sparsification, and variance minimization [68, 65, 127]. While these can handle structured outliers, they are challenged by adversarial attacks, which can be mitigated by augmenting the training set with adversarially manipulated data [64, 94, 118]. Despite their effectiveness, the latter fall short in performance guarantees, which motivates distributionally robust alternatives that minimize the worst-case expected loss over a prescribed ambiguity set of training distributions [20]. Means of quantifying uncertainty include momentum, likelihood, Kullback-Leibler (KL), and the Wasserstein distance [44, 75, 6]. Unfortunately, all robust approaches so far result in suboptimal solvers. In this context, the distributionally robust optimization framework is developed in this thesis for training a parametric model, both in centralized and federated learning settings. The objective is to endow the trained model with robustness against adversarially manipulated input data, or, distributional uncertainties, such as mismatches between training and testing data distributions, or among datasets stored at different workers. To this aim, the data distribution is assumed unknown, and lies within a Wasserstein ball centered around the empirical data distribution. This robust learning task entails an infinite-dimensional optimization problem, which is challenging. Leveraging a strong duality result, a surrogate is obtained, for which three stochastic primal-dual algorithms are developed: i) stochastic proximal gradient descent with an $\epsilon$-accurate oracle, which invokes

an oracle to solve the convex sub-problems; ii) stochastic proximal gradient descent-ascent, which approximates the solution of the convex sub-problems via a single gradient ascent step; and, iii) a distributionally robust federated learning algorithm, which solves the sub-problems locally at different workers where data are stored. Compared to the empirical risk minimization and federated learning methods, the proposed algorithms offer robustness with little computation overhead. Numerical tests using image datasets showcase the merits of the proposed algorithms under several existing adversarial attacks and distributional uncertainties.

## 1.3 Robust semi-supervised inference over graphs.

Inference tasks over social, brain, communication, biological, transportation, and sensor networks, have well-documented success by capitalizing on inter-dependencies captured by graphs [167, 90]. In practice however, data are only available at a subset of nodes, due to e.g. sampling costs, and computational or privacy constraints. As inference is desired across all network nodes, such SSL tasks over networks can benefit from the underlying graph topology [35, 16, 111]. Recent advances in graph neural networks (GNNs), offer parametric models that leverage the topology-guided structure of network data to form nested architectures that conveniently express processes over graphs [218, 196, 56].

By succinctly encoding local graph structures and features of nodes, state-of-the-art GNNs can scale linearly with the size of graph. Despite their success in practice, most of existing methods are unable to handle graphs with uncertain nodal attributes. Specifically whenever mismatches between training and testing data distribution exists, these models fail in practice. Challenges also arise due to distributional uncertainties associated with data acquired by noisy measurements. For instance, small perturbations to input data could significantly deteriorate the regression performance or result in classification error [223, 80], just to name a couple of undesirable consequences. Hence, it is critical to endow learning and inference of processes over graphs with robustness against distributional uncertainties and adversarial data, especially in safety-critical applications, such as robotics [175] and transportation [217]. In this context, a distributionally robust learning framework is developed, where the objective is to train models that exhibit quantifiable robustness against perturbations. The data distribution is considered unknown, but lies within a Wasserstein ball centered around empirical data distribution. A robust model is obtained by minimizing the worst expected loss over this ball. However, solving

the emerging functional optimization problem is challenging, if not impossible. Advocating a strong duality condition, we develop a principled method that renders the problem tractable and efficiently solvable. Experiments assess the performance of the proposed method.

## 1.4 Deep- and reinforced-Learning for network resource management

Consider the Internet, where millions of users rely on to access millions of terabyte of content such as Netflix or Amazon movies, music, social media on a daily bases. Serving end users with high quality of service in such huge-scale is no an easy task. In reality, to meet the ever-increasing data demand, novel technologies are required. Recognized as a key component is the so-called *caching*, which refers to storing reusable popular contents across geographically distributed storage-enabled network entities. The rationale here is to alleviate unfavorable surges of data traffic by pro-actively storing anticipated highly popular contents at local storage devices during off-peak periods. Such resource pre-allocation is envisioned to provide significant savings in terms of network resources such as energy, bandwidth, and cost, in addition to increased user satisfaction. To fully unleash its potential, a content-agnostic caching entity needs to rely on available observations to learn what and when to cache. A part of my research is to empower next generation networks with "smart" caching units, capable of learning, tracking, and adapting to unknown dynamics or environments such as spatio-temporal dynamics of content popularities, network topologies, and diverse caching policies deployed across network entities. By leveraging contemporary (deep) reinforcement learning tools, novel algorithms will be developed which are capable of progressively improving network performance in online and decentralized settings.

Specifically we start with considering the caching problem in wireless networks, where small basestations (SBs) equipped with caching units have potential to handle the unprecedented demand growth in heterogeneous networks. Through low-rate, backhaul connections with the backbone, SBs can prefetch popular files during off-peak traffic hours, and service them to the edge at peak periods. To intelligently prefetch, each SB must learn what and when to cache, while taking into account SB memory limitations, the massive number of available contents, the unknown popularity profiles, as well as the space-time popularity dynamics of user file requests. In this work, local and global Markov processes model user requests, and a RL (RL) framework is put forth for finding the optimal caching policy when the transition probabilities

involved are unknown. Joint consideration of global and local popularity demands along with cache-refreshing costs allow for a simple, yet practical asynchronous caching approach. The novel RL-based caching relies on a Q-learning algorithm to implement the optimal policy in an online fashion, thus enabling the cache control unit at the SB to learn, track, and possibly adapt to the underlying dynamics. To endow the algorithm with scalability, a linear function approximation of the proposed Q-learning scheme is introduced, offering faster convergence as well as reduced complexity and memory requirements. Numerical tests corroborate the merits of the proposed approach in various realistic settings.

Then we build on this framework and consider a network of caches. In this context, distributing the limited storage capacity across network entities calls for decentralized caching schemes. Many practical caching systems involve a parent caching node connected to multiple leaf nodes to serve user file requests. To model the two-way interactive influence between caching decisions at the parent and leaf nodes, a RL framework is put forth. To handle the large continuous state space, a scalable deep RL approach is pursued. The novel approach relies on a hyper-deep Q-network to learn the Q-function, and thus the optimal caching policy, in an online fashion. Reinforcing the parent node with ability to learnand- adapt to unknown policies of leaf nodes as well as spatiotemporal dynamic evolution of file requests, results in remarkable caching performance, as corroborated through numerical tests.

Finally, we design adaptive caching mechanism wedding tools from optimization and RL. We introduce simple but flexible generic time-varying fetching and caching costs, which are then used to formulate a constrained minimization of the aggregate cost across files and time. Since caching decisions per time slot influence the content availability in future slots, the novel formulation for optimal fetch-cache decisions falls into the class of dynamic programming. Under this generic formulation, first by considering stationary distributions for the costs as well as file popularities, an efficient RL-based solver known as value iteration algorithm can be used to solve the emerging optimization problem. Then, it is shown that practical limitations on cache capacity can be handled using a particular instance of this generic dynamic pricing formulation. Under this setting, to provide a light-weight online solver for the corresponding optimization, the well-known RL algorithm, $Q$-learning, is employed to find optimal fetch-cache decisions. Numerical tests corroborating the merits of the proposed approach.

## 1.5 Data-driven, reinforced, and robust learning approaches for smart power grid

Given solar generation and load consumption predictions, voltage and reactive power control aims at optimizing reactive power injections to minimize a certain loss (e.g., power, voltage deviations), while respecting physical and operating constraints. Proper redistribution of reactive power sources can result in local correction of the power factor, increase system capacity, and improve power quality. Traditionally, reactive compensation is provided by utility-owned equipment such as tap-changing under load transformers, voltage regulators, and manually-controlled capacitor banks [159], whose slow responses and limited lifespan render them ineffective in dealing with the variability introduced by distributed energy resources. Advances in smart power inverters offer new opportunities, which can provide fast and continuously-valued reactive power injection or consumption. Methods for compensating reactive power using the inverters of PV and storage systems have been advocated in [53, 213, 85, 221, 183, 84, 105, 119]. Unfortunately, *joint* control of both traditional utility-owned devices as well as contemporary smart inverters is challenging and has not been explored thus far, primarily because they operate in different timescales (e.g., hourly versus every few seconds), and involve discrete and continuous actions. In addition, several fundamental challenges remain. How should one split the reactive power compensation duty equitably between the smart inverters and traditionally utility devices? Should the control law be centralized (potentially vulnerable), distributed (more robust), or hybrid? Whether centralized or decentralized, what variables should be used as inputs to the control algorithms? e.g., what should be the states, actions, and rewards of a RL algorithm? The main challenge arise due to the fact that the discrete on-off commitment of capacitor units is often configured on an hourly or daily basis, yet smart inverters can be controlled within milliseconds, thus challenging joint control of these two types of assets. In this context, a novel two-timescale voltage regulation scheme is developed for distribution grids by judiciously coupling data-driven with physics-based optimization. On a faster timescale, say every second, the optimal setpoints of smart inverters are obtained by minimizing instantaneous bus voltage deviations from their nominal values, based on either the exact alternating current power flow model or a linear approximant of it; whereas, on the slower timescale (e.g., every hour), shunt capacitors are configured to minimize the long-term discounted voltage deviations using a deep RL algorithm. Extensive numerical tests on a real-world $47$-bus distribution network as well as the IEEE $123$-bus test feeder using

real data corroborate the effectiveness of the novel scheme. Finally, we finish this thesis by considering fas and robust state estimation (SE) to maintain a comprehensive view of the system in real time. Conventional PSSE solvers typically entail minimizing a nonlinear and nonconvex least-squares cost using e.g., the Gauss- Newton method. Those iterative solvers however, are sensitive to initialization, and may converge to local minima. To overcome these hurdles, this thesis adapts and leverages recent advances on image denoising to introduce a PSSE approach with a regularizer capturing a deep neural network (DNN) prior. For the resultant regularized PSSE objective, a "Gauss- Newton-type" alternating minimization solver is developed. To accommodate real-time monitoring, a novel end-to-end DNN is constructed subsequently by unrolling the proposed alternating minimization solver. The deep PSSE architecture can further account for the power network topology through a graph neural network (GNN) based prior. To further endow the physics-based DNN with robustness against bad data, an adversarial DNN training method is put forth. Numerical tests using real load data on the IEEE 118-bus benchmark system showcase the improved estimation and robustness performance of the proposed scheme compared with several state-of-the-art alternatives.

## 1.6 Thesis outline

The reminder of this thesis is organized as follows. Chapter 2 puts forth distributionally robust supervised and federated learning methods. Chapter 3 builds on the developed distributionally robust supervised learning framework to arrive at a distributionally robust semi-supervised learning over graphs. Chapter 4 deals with deep and RL approaches to manage limited network resources. Finally, the objective of Chapter 5 is to design efficient learning approaches for smart grid management and control. Finally Chapter 6 presents a concluding discussion of the proposed approaches, along with future research directions.

## 1.7 Notational Conventions

Unless otherwise noted, the following notation will be used throughout the subsequent chapters. Lower- (upper-) case boldface letters denote vectors (matrices). Calligraphic letters are reserved for sets, e.g., $\mathcal{S}$. For vectors, $\|\cdot\|_2$ or $\|\cdot\|$ represents the Euclidean norm, while $\|\cdot\|_0$ denotes the $\ell_0$ pseudo-norm counting the number of nonzero entries. The $n \times n$ identity matrix is denoted by

$\mathbf{I}_n$, and all-one vector by $\mathbf{1}$, and all-zero vector $\mathbf{0}$. The size of the matricies (vectors) is omitted if it is obvious from the context; otherwise it is indicated by a subscript. Operator $(\cdot)^\top$ stands for matrix transpose, $|\cdot|$ the cardinality of a set, or the absolute value of a number.

# Chapter 2

# Learning Robust against Distributional Uncertainties and Adversarial Data

## 2.1 Introduction

Machine learning models and tasks hinge on the premise that the training data are trustworthy, reliable, and representative of the testing data. In practice however, data are usually generated and stored at geographically distributed devices (a.k.a., workers) each equipped with limited computing capability, and adhering to privacy, confidentiality, and possibly cost constraints [100]. Furthermore, the data quality is not guaranteed due to adversarially generated examples and distribution drifts across workers or from the training to testing phases [102]. Visually imperceptible perturbations to a dermatoscopic image of a benign mole can render the first-ever artificial intelligence (AI) diagnostic system approved by the U.S. Food and Drug Administration in 2018, to classify it as cancerous with 100% confidence [55]. A stranger wearing pixelated sunglasses can fool even the most advanced facial recognition software in a home security system to mistake it for the homeowner [163]. Hackers indeed manipulated readings of field devices and control centers of the Ukrainian supervisory control and data acquisition system to cause the first ever cyberattack-caused power outage in 2015 [32, 194]. Examples of such failures in widely used AI-enabled safety- and security-critical systems today could put national infrastructure and even lives at risk.

Recent research efforts have focused on devising defense strategies against adversarial attacks. These strategies fall under two groups: attack detection, and attack recovery. The former

identifies whether a given input is adversarially perturbed [67, 110], while the latter trains a model to gain robustness against such adversarial inputs [68, 161], which is also the theme of the present contribution. To robustify learning models against adversarial data, a multitude of data pre-processing schemes have been devised [127, 164], to identify anomalies not adhering to postulated or nominal data. Adversarial training on the other hand, adds imperceptible well-crafted noise to clean input data to gain robustness [64]; see also e.g., [118, 129, 137], and [34] for a recent survey. In these contributions, optimization tasks are formulated to craft adversarial perturbations. Despite their empirical success, solving the resultant optimization problems is challenging. Furthermore, analytical properties of these approaches have not been well understood, which hinders explainability of the obtained models. In addition, one needs to judiciously tune hyper parameters of the attack model, which tends to be cumbersome in practice.

On the other hand, data are typically generated and/or stored at geographically distributed sites, each having subsets of data with different distributions. While keeping data localized to e.g., respect privacy, as well as reduce communication- and computation-overhead, the federated learning (FL) paradigm targets a global model, whereby multiple devices are coordinated by a central parameter server [100]. Existing FL approaches have mainly focused on the communicating versus computing tradeoff by aggregating model updates from the learners; see e.g., [123, 101, 190, 165] and references therein. From the few works dealing with robust FL, [103] learns from dependent data through e.g., sparsification, while [91] entails an ensemble of untrusted sources. These methods are rather heuristic, and rely on aggregation to gain robustness. This context, motivates well a principled approach that accounts for the uncertainties associated with the underlying data distributions.

## 2.2 Our Contribution

Tapping on a distributionally robust optimization perspective, this Chapter develops robust learning procedures that respect privacy and ensure robustness to distributional uncertainties and adversarial attacks. Independent, identically distributed (i.i.d.) samples can be drawn from the known data distribution. Building on [169], the adversarial input perturbations are constrained to lie in a Wasserstein ball, and the sought robust model minimizes the worst-case expected loss over this ball. As the resulting formulation leads to a challenging infinite-dimensional optimization

problem, we leverage strong duality to arrive at a tractable and equivalent unconstrained minimization problem, requiring solely the empirical data distribution. To solve the latter, a stochastic proximal gradient descent (SPGD) algorithm is developed based on an $\epsilon$-accurate oracle, along with its lightweight stochastic proximal gradient descent-ascent (SPGDA) iteration. The first algorithm relies on the oracle to solve the emerging convex sub-problems to $\epsilon$-accuracy, while the second simply approximates its solution via a single gradient ascent step. To accommodate communication constraints and private or possibly untrusted datasets distributed across multiple workers, we further develop a distributionally robust federated learning (DRFL) algorithm. In a nutshell, the main contributions of this Chapter are as follows. [1]

- A regularized distributionally robust learning framework to endow machine learning models with robustness against adversarial input perturbations;
- Two efficient proximal-type distributionally robust optimization algorithms with finite-sample convergence guarantees; and,
- A distributionally robust federated learning implementation to account for untrusted and possibly anonymized data from distributed sources.

## 2.3 Outline and notation

Bold lowercase letters denote column vectors, while calligraphic uppercase fonts are reserved for sets; $\mathbb{E}[\cdot]$ represents expectation; $\nabla$ denotes the gradient operator; $(\cdot)^\top$ denotes transposition, and $\|\boldsymbol{x}\|$ is the 2-norm of the vector $\boldsymbol{x}$.

The rest of this Chapter is structured as follows. Problem formulation and its robust surrogate are the subjects of Section 2.4. The proposed SPGD with $\epsilon$-accurate oracle and SPGDA algorithms with their convergence analyses are presented in Sections 2.5 and 2.6, respectively. The DRFL implementation is discussed in Section 2.7. Numerical tests are given in Section 2.8 with conclusions drawn in Section 2.9. Technical proofs are deferred to the Appendix.

[1] The results of this Chapter have been submitted in [158]

## 2.4 Problem Statement

Consider the standard regularized statistical learning task

$$\min_{\boldsymbol{\theta}\in\Theta} \ \mathbb{E}_{\boldsymbol{z}\sim P_0}\big[\ell(\boldsymbol{\theta};\boldsymbol{z})\big] + r(\boldsymbol{\theta}) \tag{2.1}$$

where $\ell(\boldsymbol{\theta};\boldsymbol{z})$ denotes the loss of a model parameterized by the unknown parameter vector $\boldsymbol{\theta}$ on a datum $\boldsymbol{z}=(\boldsymbol{x},y)\sim P_0$, with feature $\boldsymbol{x}$ and label $y$, drawn from some nominal distribution $P_0$. Here, $\Theta$ denotes the feasible set for model parameters. To prevent over fitting or incorporate prior information, regularization term $r(\boldsymbol{\theta})$ is oftentimes added to the expected loss. Popular regularizers include $r(\boldsymbol{\theta}):=\beta\|\boldsymbol{\theta}\|_1^2$ or $\beta\|\boldsymbol{\theta}\|_2^2$, where $\beta\geq 0$ is a hyper-parameter controlling the importance of the regularization term relative to the expected loss.

In practice, the nominal distribution $P_0$ is typically unknown. Instead, we are given some data samples $\{\boldsymbol{z}_n\}_{n=1}^N\sim\widehat{P}_0^{(N)}$ (a.k.a. training data), which are drawn i.i.d. from $P_0$. Upon replacing $P_0$ with the so-called empirical distribution $\widehat{P}_0^{(N)}$ in (2.1), we arrive at the empirical loss minimization

$$\min_{\boldsymbol{\theta}\in\Theta} \ \bar{\mathbb{E}}_{\boldsymbol{z}\sim\widehat{P}_0^{(N)}}\big[\ell(\boldsymbol{\theta};\boldsymbol{z})\big] + r(\boldsymbol{\theta}) \tag{2.2}$$

where $\bar{\mathbb{E}}_{\boldsymbol{z}\sim\widehat{P}_0^{(N)}}[\ell(\boldsymbol{\theta};\boldsymbol{z})]=N^{-1}\sum_{n=1}^N\ell(\boldsymbol{\theta};\boldsymbol{z}_n)$. Indeed, a variety of machine learning tasks can be cast as (2.2), including e.g., ridge and Lasso regression, logistic regression, and reinforcement learning. The resultant models obtained by solving (2.2) however, have been shown vulnerable to adversarially corrupted data in $\widehat{P}_0^{(N)}$. Furthermore, the testing data distribution often deviates from the available $\widehat{P}_0^{(N)}$. For this reason, targeting an adversarially robust model against a set of distributions corresponding to perturbations of the underlying data distribution, has led to the formulation [169]

$$\min_{\boldsymbol{\theta}\in\Theta} \ \sup_{P\in\mathcal{P}} \ \mathbb{E}_{\boldsymbol{z}\sim P}[\ell(\boldsymbol{\theta};\boldsymbol{z})] + r(\boldsymbol{\theta}) \tag{2.3}$$

where $\mathcal{P}$ represents a set of distributions centered around the data generating distribution $\widehat{P}_0^{(N)}$. Compared with (2.1), the worst-case formulation (2.3), yields models ensuring reasonable performance across a continuum of distributions characterized by $\mathcal{P}$. In practice, different types of ambiguity sets $\mathcal{P}$ can be considered, and they lead to different robustness guarantees and computational requirements. Popular choices of $\mathcal{P}$ include momentum [44, 191], KL divergence

[75], statistical test [6], and Wasserstein distance-based ambiguity sets [6, 169]; see e.g., [20] for a recent overview. Among all choices, it has been shown that the Wasserstein ambiguity set $\mathcal{P}$ results in a tractable realization of (2.3), thanks to the strong duality result of [6] and [169], which also motivates this work.

To formalize this, consider two probability measures $P$ and $Q$ supported on set $\mathcal{Z}$, and let $\Pi(P, Q)$ be the set of all joint measures supported on $\mathcal{Z}^2$, with marginals $P$ and $Q$. Let $c : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$ measure the cost of transporting a unit of mass from $\boldsymbol{z}$ in $P$ to another element $\boldsymbol{z}'$ in $Q$. The celebrated optimal transport problem is given by [179, page 111]

$$W_c(P, Q) := \inf_{\pi \in \Pi} \mathbb{E}_\pi \big[c(\boldsymbol{z}, \boldsymbol{z}')\big]. \tag{2.4}$$

*Remark* 1. If $c(\cdot, \cdot)$ satisfies the axioms of distance, then $W_c$ defines a distance on the space of probability measures. For instance, if $P$ and $Q$ are defined over a Polish space equipped with metric $d$, then choosing $c(\boldsymbol{z}, \boldsymbol{z}') = d^p(\boldsymbol{z}, \boldsymbol{z}')$ for some $p \in [1, \infty)$ asserts that $W_c^{1/p}(P, Q)$ is the well-known Wasserstein distance of order $p$ between probability measures $P$ and $Q$ [179, Definition 6.1].

For a given empirical distribution $\widehat{P}_0^{(N)}$, define the uncertainty set $\mathcal{P} := \{P | W_c(P, \widehat{P}_0^{(N)}) \leq \rho\}$ to include all probability distributions having at most $\rho$-distance from $P_0^{(N)}$. Incorporating this ambiguity set into (2.3), yields the following reformulation

$$\min_{\boldsymbol{\theta} \in \Theta} \sup_{P} \mathbb{E}_{\boldsymbol{z} \sim P}[\ell(\boldsymbol{\theta}; \boldsymbol{z})] + r(\boldsymbol{\theta}) \tag{2.5a}$$

$$\text{s.t.} \quad W_c(P, \widehat{P}_0^{(N)}) \leq \rho. \tag{2.5b}$$

Observe that the inner supremum in (2.5a) runs over all joint probability measures $\pi$ on $\mathcal{Z}^2$ implicitly characterized by (2.5b). Intuitively, directly solving this optimization over the infinite-dimensional space of distribution functions is challenging, if not impossible. Fortunately, for a broad range of losses as well as transport costs, it has been shown that the inner maximization satisfies a strong duality condition [20]; that is, the optimal objective of this inner maximization and its Lagrangian dual optimal objective, are equal. In addition, the dual problem involves optimization over a one-dimensional dual variable. These two observations make it possible to solve (2.3) in the dual domain. To formally obtain a tractable surrogate to (5.37), we make the following assumptions.

**Assumption 1.** *The transportation cost function $c : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$, is a lower semi-continuous function satisfying $c(\boldsymbol{z}, \boldsymbol{z}) = 0$ for $\boldsymbol{z} \in \mathcal{Z}$*[2].

**Assumption 2.** *The loss function $\ell : \Theta \times \mathcal{Z} \to [0, \infty)$, is upper semi-continuous, and integrable.*

The following proposition provides a tractable surrogate for (5.37), whose proof can be found in [20, Theorem 1].

**Proposition 1.** *Let $\ell : \Theta \times \mathcal{Z} \to [0, \infty)$, and $c : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$ satisfy Assumptions* 1 *and* 2*, respectively. Then, for any given $\widehat{P}_0^{(N)}$, and $\rho > 0$, it holds that*

$$\sup_{P \in \mathcal{P}} \mathbb{E}_{\boldsymbol{z} \sim P}[\ell(\boldsymbol{\theta}; \boldsymbol{z})] \quad = \inf_{\gamma \geq 0} \big\{ \bar{\mathbb{E}}_{\boldsymbol{z} \sim \widehat{P}_0^{(N)}} \big[ \sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \{ \ell(\boldsymbol{\theta}; \boldsymbol{\zeta}) - \gamma(c(\boldsymbol{z}, \boldsymbol{\zeta}) - \rho) \} \big] \big\} \tag{2.6}$$

*where $\mathcal{P} := \big\{ P | W_c(P, \widehat{P}_0^{(N)}) \leq \rho \big\}$.*

*Remark* 2. Thanks to strong duality, the right-hand side in (5.38) simply is a univariate dual reformulation of the primal problem represented in the left-hand side. In sharp contrast with the primal formulation, the expectation in the dual domain is taken only over the empirical $\widehat{P}_0^{(N)}$ rather than any $P \in \mathcal{P}$. In addition, since this reformulation circumvents the need for finding the optimal $\pi \in \Pi$ to form $\mathcal{P}$, and characterizing the primal objective $\forall P \in \mathcal{P}$, it is practically more convenient.

Upon relying on Proposition 3, the following distributionally robust surrogate is obtained

$$\min_{\boldsymbol{\theta} \in \Theta} \inf_{\gamma \geq 0} \big\{ \bar{\mathbb{E}}_{\boldsymbol{z} \sim \widehat{P}_0^{(N)}} \big[ \sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \{ \ell(\boldsymbol{\theta}; \boldsymbol{\zeta}) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta})) \} + r(\boldsymbol{\theta}) \big] \big\}. \tag{2.7}$$

*Remark* 3. The robust surrogate in (5.39) boils down to minimax (saddle-point) optimization which has been widely studied in e.g., [104]. However, (5.39) requires the supremum to be solved separately for each sample $\boldsymbol{z}$, and the problem cannot be handled through existing methods.

A relaxed (hence suboptimal) version of (5.39) with a fixed $\gamma$ value has recently been studied in [169]. Unfortunately, one has to select an appropriate $\gamma$ value using cross validation over a grid search that is also application dependent. Heuristically choosing a $\gamma$ does not guarantee optimality in solving the distributionally robust surrogate (5.39). Clearly, the effect of heuristically selecting $\gamma$ is more pronounced when training deep neural networks. Instead, we advocate algorithms that optimize $\gamma$ and $\boldsymbol{\theta}$ simultaneously.

[2] A simple example satisfying these constraints is the Euclidean distance $c(\boldsymbol{z}, \boldsymbol{z}') = \|\boldsymbol{z} - \boldsymbol{z}'\|$.

Our approach to addressing this, relies on the structure of (5.39) to *iteratively* update parameters $\bar{\boldsymbol{\theta}} := [\boldsymbol{\theta}^\top \ \gamma]^\top$ and $\boldsymbol{\zeta}$. To end up with a differentiable function of $\bar{\boldsymbol{\theta}}$ after maximizing over $\boldsymbol{\zeta}$, Danskin's theorem requires the sup-problem to have a unique solution [17]. For this reason, we design the inner maximization to involve a strongly concave objective function through the selection of a strongly convex transportation cost, such as $c(\boldsymbol{z}, \boldsymbol{z'}) := \|\boldsymbol{z} - \boldsymbol{z'}\|_p^2$ for $p \geq 1$. For the maximization over $\boldsymbol{\zeta}$ to rely on a strongly concave objective, we let $\gamma \in \Gamma := \{\gamma | \gamma > \gamma_0\}$, where $\gamma_0$ is large enough. Since $\gamma$ is the dual variable corresponding to the constraint in (5.37), having $\gamma \in \Gamma$ is tantamount to tuning $\rho$, which in turn *controls* the level of *robustness*. Replacing $\gamma \geq 0$ in (5.39) with $\gamma \in \Gamma$, our *robust learning model* is obtained as the solution of

$$\min_{\boldsymbol{\theta} \in \Theta} \ \inf_{\gamma \in \Gamma} \ \bar{\mathbb{E}}_{\boldsymbol{z} \sim \widehat{P}_0^{(T)}} \big[ \sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z}) \big] + r(\bar{\boldsymbol{\theta}}) \tag{2.8}$$

where $\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z}) := \ell(\boldsymbol{\theta}; \boldsymbol{\zeta}) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta}))$. Intuitively, input $\boldsymbol{z}$ in (5.40) is pre-processed by maximizing $\psi$ accounting for the adversarial perturbation. To iteratively solve our objective in (5.40), the ensuing sections provide efficient solvers under some mild conditions. Those include cases, every inner maximization (supremum) can be solved to $\epsilon$-optimality by an oracle.

Before developing our algorithms, we make several standard assumptions; see also [169], [104].

**Assumption 3.** *Function $c(\boldsymbol{z}, \cdot)$ is $L_c$-Lipschitz and $\mu$-strongly convex for any given $\boldsymbol{z} \in \mathcal{Z}$, with respect to the norm $\|\cdot\|$.*

**Assumption 4.** *The loss function $\ell(\boldsymbol{\theta}; \boldsymbol{z})$ obeys the following Lipschitz smoothness conditions*

$$\|\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{z}) - \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta'}; \boldsymbol{z})\|_* \leq L_{\boldsymbol{\theta\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta'}\| \tag{2.9a}$$

$$\|\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{z}) - \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{z'})\|_* \leq L_{\boldsymbol{\theta z}} \|\boldsymbol{z} - \boldsymbol{z'}\| \tag{2.9b}$$

$$\|\nabla_{\boldsymbol{z}} \ell(\boldsymbol{\theta}; \boldsymbol{z}) - \nabla_{\boldsymbol{z}} \ell(\boldsymbol{\theta}; \boldsymbol{z'})\|_* \leq L_{\boldsymbol{zz}} \|\boldsymbol{z} - \boldsymbol{z'}\| \tag{2.9c}$$

$$\|\nabla_{\boldsymbol{z}} \ell(\boldsymbol{\theta}; \boldsymbol{z}) - \nabla_{\boldsymbol{z}} \ell(\boldsymbol{\theta'}; \boldsymbol{z})\|_* \leq L_{\boldsymbol{z\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta'}\| \tag{2.9d}$$

*and it is continuously differentiable with respect to $\boldsymbol{\theta}$.*

Assumption (4) guarantees that the supremum in (5.39) results in a smooth function of $\bar{\boldsymbol{\theta}}$; thus, one can execute gradient descent to update $\boldsymbol{\theta}$ upon solving the supremum. This will further help to provide convergence analysis of our proposed algorithms. To elaborate more on this, the

**Algorithm 1** SPGD with $\epsilon$-accurate oracle

**Input**: Initial guess $\bar{\boldsymbol{\theta}}^0$, step size sequence $\{\alpha_t > 0\}_{t=0}^T$, $\epsilon$-accurate oracle
$t = 1, \ldots, T$ Draw i.i.d samples $\{\boldsymbol{z}_n\}_{n=1}^N$
Find $\epsilon$-optimizer $\boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}_n)$ via the oracle
Update:
$\bar{\boldsymbol{\theta}}^{t+1} = \mathrm{prox}_{\alpha_t r}\Big[\bar{\boldsymbol{\theta}}^t - \frac{\alpha_t}{N}\sum_{n=1}^N \nabla_{\bar{\boldsymbol{\theta}}}\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}_n); \boldsymbol{z}_n)\big|_{\bar{\boldsymbol{\theta}}=\bar{\boldsymbol{\theta}}^t}\Big]$

following lemma characterizes the smoothness and gradient Lipschitz properties obtained upon solving the maximization problem in (5.40).

**Lemma 1.** *For each $\boldsymbol{z} \in \mathcal{Z}$, define $\bar{\psi}(\bar{\boldsymbol{\theta}}; \boldsymbol{z}) = \sup_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z})$ with $\boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}; \boldsymbol{z}) = \arg\max_{\boldsymbol{\zeta}\in\mathcal{Z}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z})$. Then $\bar{\psi}(\cdot)$ is differentiable, and its gradient is $\nabla_{\bar{\boldsymbol{\theta}}}\bar{\psi}(\bar{\boldsymbol{\theta}}; \boldsymbol{z}) = \nabla_{\bar{\boldsymbol{\theta}}}\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}; \boldsymbol{z}); \boldsymbol{z})$. Moreover, the following conditions hold*

$$\big\|\boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}_1; \boldsymbol{z}) - \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}_2; \boldsymbol{z})\big\| \leq \frac{L_{\boldsymbol{z}\boldsymbol{\theta}}}{\lambda}\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\| + \frac{L_c}{\lambda}\|\gamma_2 - \gamma_1\| \tag{2.10a}$$

$$\big\|\nabla_{\bar{\boldsymbol{\theta}}}\bar{\psi}(\bar{\boldsymbol{\theta}}_1; \boldsymbol{z}) - \nabla_{\bar{\boldsymbol{\theta}}}\bar{\psi}(\bar{\boldsymbol{\theta}}_2; \boldsymbol{z})\big\| \leq \frac{L_{\boldsymbol{\theta}z}L_c + L_c^2}{\lambda}\|\gamma_2 - \gamma_1\| \tag{2.10b}$$

$$+ \Big(L_{\boldsymbol{\theta}\boldsymbol{\theta}} + \frac{L_{\boldsymbol{\theta}z}L_{\boldsymbol{z}\boldsymbol{\theta}} + L_cL_{\boldsymbol{z}\boldsymbol{\theta}}}{\lambda}\Big)\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\| \tag{2.10c}$$

*where $\gamma^{1,2} \in \Gamma$, and $\psi(\bar{\boldsymbol{\theta}}, \cdot; \boldsymbol{z})$ is $\lambda$-strongly concave.*

Proof: See Appendix A.0.1 for the proof.

Lemma 1 paves the way for iteratively solving the surrogate optimization (5.40), intuitively because it guarantees a differentiable and smooth objective upon solving the inner supremum to its optimum.

*Remark* 4. Equation (2.10a) is appealing in practice. Indeed, if $\bar{\boldsymbol{\theta}}^t = [\boldsymbol{\theta}^t, \gamma^t]$ is updated with a small enough step size, the corresponding $\boldsymbol{\zeta}_*(\boldsymbol{\theta}^{t+1}; \boldsymbol{z})$ is close enough to $\boldsymbol{\zeta}_*(\boldsymbol{\theta}^t; \boldsymbol{z})$. Building on this observation, instead of using an oracle to find the optimum $\boldsymbol{\zeta}_*(\boldsymbol{\theta}^{t+1}; \boldsymbol{z})$, an $\epsilon$-accurate solution $\boldsymbol{\zeta}_\epsilon(\boldsymbol{\theta}^{t+1}; \boldsymbol{z})$ suffices to obtain comparable performance. This also circumvents the need to find the optimum for the inner maximization per iteration, which could be computationally demanding.

## 2.5 Stochastic Proximal Gradient Descent with $\epsilon$-accurate Oracle

A standard solver of regularized optimization problems is the proximal gradient algorithm. In this section, we develop a variant of it to tackle the robust surrogate (5.40). For convenience, let us define

$$f(\boldsymbol{\theta}, \gamma) := \mathbb{E}\left[ \sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \{\ell(\boldsymbol{\theta}; \boldsymbol{\zeta}) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta}))\} \right] \tag{2.11}$$

and rewrite our objective as

$$\min_{\boldsymbol{\theta} \in \Theta} \inf_{\gamma \in \Gamma} \; F(\boldsymbol{\theta}, \gamma) := f(\boldsymbol{\theta}, \gamma) + r(\boldsymbol{\theta}) \tag{2.12}$$

where $f(\boldsymbol{\theta}, \gamma)$ is the smooth function in (2.11), and $r(\cdot)$ is a non-smooth and convex regularizer, such as the $\ell_1$-norm. With a slight abuse of notation, upon introducing $\bar{\boldsymbol{\theta}} := [\boldsymbol{\theta}\ \gamma]$, we define $f(\bar{\boldsymbol{\theta}}) := f(\boldsymbol{\theta}, \gamma)$ and $F(\bar{\boldsymbol{\theta}}) := F(\boldsymbol{\theta}, \gamma)$.

The proximal gradient algorithm the updates $\bar{\boldsymbol{\theta}}^t$ as

$$\bar{\boldsymbol{\theta}}^{t+1} = \arg\min_{\boldsymbol{\theta}} \; \alpha_t r(\boldsymbol{\theta}) + \alpha_t \big\langle \boldsymbol{\theta} - \bar{\boldsymbol{\theta}}^t, \boldsymbol{g}(\bar{\boldsymbol{\theta}}^t) \big\rangle + \frac{1}{2} \big\| \boldsymbol{\theta} - \bar{\boldsymbol{\theta}}^t \big\|^2$$

where $\boldsymbol{g}(\bar{\boldsymbol{\theta}}^t) := \nabla f(\bar{\boldsymbol{\theta}})|_{\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}^t}$, and $\alpha_t > 0$ is some step size. The last update is expressed in the compact form

$$\bar{\boldsymbol{\theta}}^{t+1} = \mathrm{prox}_{\alpha_t r}\big[\bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}(\bar{\boldsymbol{\theta}}^t)\big] \tag{2.13}$$

where the proximal gradient operator is given by

$$\mathrm{prox}_{\alpha r}[\boldsymbol{v}] := \arg\min_{\boldsymbol{\theta}} \; \alpha r(\boldsymbol{\theta}) + \frac{1}{2} \|\boldsymbol{\theta} - \boldsymbol{v}\|^2. \tag{2.14}$$

The working assumption is that this optimization problem can be solved efficiently using off-the-shelf solvers.

Starting from the guess $\bar{\boldsymbol{\theta}}^0$, the proposed SPGD with $\epsilon$-accurate oracle executes two steps per iteration $t = 1, 2, \ldots$. First, it relies on an $\epsilon$-accurate maximum oracle to solve the inner problem $\sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \{\ell(\boldsymbol{\theta}^t; \boldsymbol{\zeta}) - \gamma^t c(\boldsymbol{z}, \boldsymbol{\zeta})\}$ for randomly drawn samples $\{\boldsymbol{z}_n\}_{n=1}^N$ to yield $\epsilon$-optimal $\boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t, \boldsymbol{z}_n)$ with the corresponding objective values $\psi(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t, \boldsymbol{z}_n); \boldsymbol{z}_n)$. Next, $\bar{\boldsymbol{\theta}}^t$ is updated

using a stochastic proximal gradient step as

$$\bar{\boldsymbol{\theta}}^{t+1} = \mathrm{p}rox_{\alpha_t r}\Big[\bar{\boldsymbol{\theta}}^t - \frac{\alpha_t}{N}\sum_{n=1}^{N}\nabla_{\bar{\boldsymbol{\theta}}}\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}_n); \boldsymbol{z}_n)\Big].$$

For implementation, the proposed SPGD algorithm with $\epsilon$-accurate oracle is summarized in Alg. 1. Convergence performance of this algorithm is analyzed in the ensuing subsection.

### 2.5.1 Convergence of SPGD with $\epsilon$-accurate oracle

In general, the postulated model is nonlinear, and the robust surrogate $F(\bar{\boldsymbol{\theta}})$ is nonconvex. In this section, we characterize the convergence performance of Alg. 1 to a stationary point. However, lack of convexity and smoothness implies that stationary points must be understood in the sense of the Frèchet subgradient. Specifically, the Frèchet subgradient $\partial F(\check{\boldsymbol{\theta}})$ for the composite optimization in (2.12), is the set [144]

$$\partial F(\check{\boldsymbol{\theta}}) := \Big\{\boldsymbol{v} \,\Big|\, \lim_{\bar{\boldsymbol{\theta}}\to\check{\boldsymbol{\theta}}} \inf \frac{F(\bar{\boldsymbol{\theta}}) - F(\check{\boldsymbol{\theta}}) - \boldsymbol{v}^\top(\bar{\boldsymbol{\theta}} - \check{\boldsymbol{\theta}}))}{\|\bar{\boldsymbol{\theta}} - \check{\boldsymbol{\theta}}\|} \geq 0\Big\}.$$

Consequently, the distance between vector $\mathbf{0}$ and the set $\partial F(\check{\boldsymbol{\theta}})$ is a measure characterizing whether a point is stationary or not. To this end, define the distance between a vector $\boldsymbol{v}$ and a set $\mathcal{S}$ as $\mathrm{dist}(\boldsymbol{v}, \mathcal{S}) := \min_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{v} - \boldsymbol{s}\|$, and the notion of $\delta$-stationary points as defined next.

**Definition 1.** *Given a small $\delta > 0$, we call vector $\check{\boldsymbol{\theta}}$ a $\delta$-stationary point if and only if* $\mathrm{dist}(\mathbf{0}, \partial F(\check{\boldsymbol{\theta}})) \leq \delta$.

Since $f(\cdot)$ in (2.11) is smooth, we have that $\partial F(\bar{\boldsymbol{\theta}}) = \nabla f(\bar{\boldsymbol{\theta}}) + \partial r(\bar{\boldsymbol{\theta}})$ [144]. Hence, it suffices to prove that the algorithm converges to a $\delta$-stationary point $\check{\boldsymbol{\theta}}$ satisfying

$$\mathrm{dist}\big(\mathbf{0}, \nabla f(\check{\boldsymbol{\theta}}) + \partial r(\check{\boldsymbol{\theta}})\big) \leq \delta. \tag{2.15}$$

We further adopt the following assumption that is standard in stochastic optimization.

**Assumption 5.** *Function $f$ satisfies the next two conditions.*

1. *Gradient estimates are unbiased and have a bounded variance, i.e., $\mathbb{E}[\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t) - \nabla f(\bar{\boldsymbol{\theta}}^t)] = \mathbf{0}$, and there is a constant $\sigma^2 < \infty$, so that $\mathbb{E}[\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\|_2^2] \leq \sigma^2$.*

2. *Function $f(\bar{\boldsymbol{\theta}})$ is smooth with $L_f$-Lipschitz continuous gradient, i.e.,* $\|\nabla f(\bar{\boldsymbol{\theta}}_1) - \nabla f(\bar{\boldsymbol{\theta}}_2)\| \leq L_f \|\bar{\boldsymbol{\theta}}_1 - \bar{\boldsymbol{\theta}}_2\|$.

We are now ready to claim the convergence guarantees for Alg. 1; see Appendix A.0.2 for the proof.

**Theorem 1.** *Let Alg. 2 run for $T$ iterations with constant step sizes $\alpha, \eta > 0$. Under Assumptions 1–5, Alg. 2 generates a sequence of $\{\bar{\boldsymbol{\theta}}^t\}$ that satisfies*

$$\mathbb{E}\left[dist(\mathbf{0}, \partial F(\bar{\boldsymbol{\theta}}^{t'}))^2\right] \leq \Big(\frac{2}{\alpha} + \beta\Big)\frac{\Delta_F}{T} + \Big(\frac{\beta}{\eta} + 2\Big)\sigma^2 + \frac{(\beta+2)L_{\bar{\boldsymbol{\theta}}_z}^2 \epsilon}{\lambda_0} \tag{2.16}$$

*where $t'$ is uniformly sampled from $\{1, \ldots, T\}$; here, $\Delta_F := F(\bar{\boldsymbol{\theta}}^0) - F(\bar{\boldsymbol{\theta}}^{T+1})$; $L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2 := L_{\boldsymbol{\theta}\boldsymbol{z}}^2 + \lambda_0 L_c$, and $\beta,\ \lambda_0 > 0$ are some constants.*

Theorem 1 asserts that $\{\bar{\boldsymbol{\theta}}^t\}_{t=1}^T$ generated by Alg. 1 converges to a stationary point on average. The upper bound here is characterized by the initial error $\Delta_F$, which decays at the rate of $\mathcal{O}(1/T)$; and, the constant bias terms induced by the gradient estimate variance $\sigma^2$ as well as the oracle accuracy $\epsilon$.

*Remark* 5 (Oracle implementation). The $\epsilon$-accurate oracle can be implemented in practice by several optimization algorithms, with gradient ascent being a desirable one due to its simplicity. Assuming $\gamma_0 \geq L_{\boldsymbol{z}\boldsymbol{z}}/\mu$, gradient ascent with constant step size $\eta$ obtains an $\epsilon$-accurate solution within at most $\mathcal{O}(\log(d_0^2/\epsilon\eta))$ iterations, where $d_0$ is the diameter of set $\mathcal{Z}$.

The computational complexity of Alg. 1 can grow prohibitively when dealing with large-size datasets and complex models. This motivates lightweight, scalable, yet efficient methods. To this end, we introduce next a stochastic proximal gradient descent-ascent (SPGDA) algorithm.

## 2.6 Stochastic Proximal Gradient Descent-Ascent

Leveraging the strong concavity of the inner maximization problem and Lemma 1, a lightweight variant of the SPGD with $\epsilon$-accurate oracle is developed here. Instead of optimizing the inner maximization problem to $\epsilon$-accuracy by an oracle, we approximate its solution after only a *single* gradient ascent step. Specifically, for a batch of data $\{\boldsymbol{z}_m^t\}_{m=1}^M$ per iteration $t$, our SPGDA

**Algorithm 2** SPGDA

**Input**: Initial guess $\bar{\boldsymbol{\theta}}^0$, step size sequence $\{\alpha_t, \eta_t > 0\}_{t=0}^T$, batch size $M$
For $t = 1, \ldots, T$ Draw a batch of i.i.d samples $\{\boldsymbol{z}_m\}_{m=1}^M$
Find $\{\boldsymbol{\zeta}_m^t\}_{m=1}^M$ via gradient ascent: $\boldsymbol{\zeta}_m^t = \boldsymbol{z}_m^t + \eta_t \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}; \boldsymbol{z}_m^t)\big|_{\boldsymbol{\zeta}=\boldsymbol{z}_m^t}, \quad m = 1, \ldots, M$
Update: $\bar{\boldsymbol{\theta}}^{t+1} = \mathrm{prox}_{\alpha_t r}\Big[\bar{\boldsymbol{\theta}}^t - \frac{\alpha_t}{M} \sum_{m=1}^M \nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}_m^t; \boldsymbol{z}_m^t)\big|_{\bar{\boldsymbol{\theta}}=\bar{\boldsymbol{\theta}}^t}\Big]$

algorithm first perturbs each datum via a gradient ascent step

$$\boldsymbol{\zeta}_m^t = \boldsymbol{z}_m^t + \eta_t \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}; \boldsymbol{z}_m^t)\big|_{\boldsymbol{\zeta}=\boldsymbol{z}_m^t}, \ \forall m = 1, \ldots, M \tag{2.17}$$

and then forms

$$\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) := \frac{1}{M} \sum_{m=1}^M \nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}_m^t; \boldsymbol{z}_m^t)\big|_{\bar{\boldsymbol{\theta}}=\bar{\boldsymbol{\theta}}^t}. \tag{2.18}$$

Using (2.18), an extra proximal gradient step is taken to obtain

$$\bar{\boldsymbol{\theta}}^{t+1} = \mathrm{prox}_{\alpha_t r}\big[\bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)\big]. \tag{2.19}$$

The SPGDA steps are summarized in Alg. 2. Besides its simplicity and scalability, SPGDA enjoys convergence to a stationary point as elaborated next.

### 2.6.1 Convergence of SPGDA

To prove convergence of Alg. 2, let us start by defining

$$\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t) := \frac{1}{M} \sum_{m=1}^M \nabla_{\bar{\boldsymbol{\theta}}} \boldsymbol{\psi}^*(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}_m^*; \boldsymbol{z}_m^t). \tag{2.20}$$

Different from (2.18), the gradient here is obtained at the optimum $\boldsymbol{\zeta}_m^*$. To establish convergence, one more assumption is needed.

**Assumption 6.** *Function $f$ satisfies the following conditions.*

*1) Gradient estimates $\nabla_{\bar{\boldsymbol{\theta}}} \boldsymbol{\psi}^*(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}_m^*; \boldsymbol{z}_m)$ at $\boldsymbol{\zeta}_m^*$ are unbiased and have bounded variance. That is, for $m = 1 \cdots M$, we have $\mathbb{E}\left[\nabla_{\bar{\boldsymbol{\theta}}} \psi^*(\boldsymbol{\theta}, \boldsymbol{\zeta}_m^*; \boldsymbol{z}_m) - \nabla_{\bar{\boldsymbol{\theta}}} f(\boldsymbol{\theta})\right] = \mathbf{0}$ and $\mathbb{E}\left[\|\nabla_{\bar{\boldsymbol{\theta}}} \psi^*(\boldsymbol{\theta}, \boldsymbol{\zeta}_m^*; \boldsymbol{z}_m) - \nabla f(\boldsymbol{\theta})\|^2\right] \leq \sigma^2$.*

*2) The expected norm of $\boldsymbol{g}^t(\bar{\boldsymbol{\theta}})$ is bounded, that is, $\mathbb{E}\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}})\|^2 \leq B^2$.*

We now present a theorem on the convergence of Alg. 2; see Appendix A.0.3 for the proof.

**Theorem 2** (Convergence of Alg. 2)**.** *Let $\Delta_F := F(\bar{\boldsymbol{\theta}}^0) - \inf_{\bar{\boldsymbol{\theta}}} F(\bar{\boldsymbol{\theta}})$, and $D$ denote the diameter of the feasible set $\Theta$. Under As. 1–4 and 6, for a constant step size $\alpha > 0$, and a fixed batch size $M > 0$, after $T$ iterations, Alg. 2 satisfies*

$$\mathbb{E}\big[\mathrm{dist}(\mathbf{0}, \partial F(\bar{\boldsymbol{\theta}}^T))^2\big] \leq \frac{\upsilon}{T+1}\Delta_F + \frac{4\sigma^2}{M} + \frac{2L_{\boldsymbol{\theta z}}^2\nu}{M}\big[(1-\alpha\mu)\,D^2 + \alpha^2 B^2\big] \tag{2.21}$$

*where $\upsilon$, $\nu$, and $\mu = \gamma_0 - L_{\boldsymbol{zz}}$ are some positive constants.*

Theorem 2 implies that the sequence $\{\bar{\boldsymbol{\theta}}^t\}_{t=1}^T$ generated by Alg. 2 converges to a stationary point. The upper bound in (2.21) is characterized by a vanishing term induced by initial error $\Delta_F$, and constant bias terms.

## 2.7 Distributionally Robust Federated Learning

In practice, massive datasets are distributed geographically across multiple sites, where scalability, data privacy and integrity, as well as bandwidth scarcity typically discourage uploading them to a central server. This has propelled the so-called federated learning framework, where multiple workers exchange information with a server to learn a centralized model using data locally generated and/or stored across workers [123, 102, 100, 37]. Workers in this learning framework communicate *iteratively* with the server. Albeit appealing for its scalability, one needs to carefully address the bandwidth bottleneck associated with server-worker links. Furthermore, the workers' data may have (slightly) different underlying distributions, which further challenges the learning task. To seek a model robust to distribution drifts across workers, we will adapt our novel SPGDA approach to design a privacy-respecting and robust algorithm.

To that end, consider $K$ workers with each worker $k \in \mathcal{K}$ collecting samples $\{\boldsymbol{z}_n(k)\}_{n=1}^N$. A globally shared model parameterized by $\boldsymbol{\theta}$ is to be updated at the server by aggregating gradients computed locally per worker. For simplicity, we consider workers having the same number of samples $N$. The goal is to learn a single global model from stored data at all workers by

minimizing the following objective function

$$\min_{\boldsymbol{\theta}\in\Theta} \bar{\mathbb{E}}_{\boldsymbol{z}\sim\widehat{P}}[\ell(\boldsymbol{\theta};\boldsymbol{z})] + r(\boldsymbol{\theta}) \tag{2.22}$$

where $\bar{\mathbb{E}}_{\boldsymbol{z}\sim\widehat{P}}[\ell(\boldsymbol{\theta};\boldsymbol{z})] := \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}\ell(\boldsymbol{\theta},\boldsymbol{z}_n(k))$. To endow the learned model with robustness against distributional uncertainties, our novel formulation will solve the following problem in a distributed fashion

$$\begin{aligned}&\min_{\boldsymbol{\theta}\in\Theta}\ \sup_{P\in\mathcal{P}}\ \mathbb{E}_{\boldsymbol{z}\sim P}[\ell(\boldsymbol{\theta};\boldsymbol{z})] + r(\boldsymbol{\theta})\\ &\text{s. to. } \mathcal{P} := \Big\{P\Big|\sum_{k=1}^{K} W_c(P,\widehat{P}^{(N)}(k)) \le \rho\Big\}\end{aligned} \tag{2.23}$$

where $W_c(P,\widehat{P}^{(N)}(k))$ denotes the Wasserstein distance between distribution $P$ and the local $\widehat{P}^{(N)}(k)$, per worker $k$.

Clearly, the constraint $P\in\mathcal{P}$, couples the optimization in (6.3) across all workers. To offer distributed implementations, we resort to Proposition 3, to arrive at the equivalent reformulation

$$\min_{\boldsymbol{\theta}\in\Theta}\ \inf_{\gamma\in\Gamma}\sum_{k=1}^{K}\bar{\mathbb{E}}_{\boldsymbol{z}(k)\sim\widehat{P}^{(N)}(k)}\big[\sup_{\boldsymbol{\zeta}\in\mathcal{Z}}\{\ell(\boldsymbol{\theta};\boldsymbol{\zeta}) + \gamma(\rho - c(\boldsymbol{z}(k),\boldsymbol{\zeta}))\}\big] + r(\boldsymbol{\theta}). \tag{2.24}$$

Next, we present our communication- and computation-efficient DRFL that builds on the SPGDA scheme in Sec. 2.6.

Specifically, our DRFL hinges on the fact that with fixed server parameters $\bar{\boldsymbol{\theta}}^t := [\boldsymbol{\theta}^{t\top},\gamma^t]^\top$ per iteration $t$, the optimization problem becomes *separable* across all workers. Hence, upon receiving $\bar{\boldsymbol{\theta}}^t$ from the server, each worker $k\in\mathcal{K}$: i) samples a minibatch $\mathcal{B}^t(k)$ of data from $\widehat{P}^{(N)}(k)$; ii) forms the *perturbed* loss $\psi_k(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta};\boldsymbol{z}) := \ell(\boldsymbol{\theta}^t;\boldsymbol{\zeta}) + \gamma^t(\rho - c(\boldsymbol{z},\boldsymbol{\zeta}))$ for each $\boldsymbol{z}\in\mathcal{B}^t(k)$; iii) lazily maximizes $\psi_k(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta};\boldsymbol{z})$ over $\boldsymbol{\zeta}$ using a single gradient ascent step to yield $\boldsymbol{\zeta}(\bar{\boldsymbol{\theta}}^t;\boldsymbol{z}) = \boldsymbol{z} + \eta_t\nabla_{\boldsymbol{\zeta}}\psi_k(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta};\boldsymbol{z})|_{\boldsymbol{\zeta}=\boldsymbol{z}}$; and, iv) sends the stochastic gradient $|\mathcal{B}^t(k)|^{-1}\sum_{\boldsymbol{z}\in\mathcal{B}^t(k)}\nabla_{\bar{\boldsymbol{\theta}}}\psi_k(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}(\bar{\boldsymbol{\theta}}^t;\boldsymbol{z});\boldsymbol{z})\big|_{\bar{\boldsymbol{\theta}}=\bar{\boldsymbol{\theta}}^t}$ back to the server. Upon receiving all local gradients, the server updates $\bar{\boldsymbol{\theta}}^t$ using a proximal gradient descent step to find $\bar{\boldsymbol{\theta}}^{t+1}$, that is

$$\bar{\boldsymbol{\theta}}^{t+1} = \mathrm{prox}_{\alpha_t r}\left[\bar{\boldsymbol{\theta}}^t - \frac{\alpha_t}{K}\sum_{k=1}^{K}\frac{1}{|\mathcal{B}^t(k)|}\times\sum_{\boldsymbol{z}\in\mathcal{B}^t(k)}\nabla_{\bar{\boldsymbol{\theta}}}\psi_k(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}(\bar{\boldsymbol{\theta}}^t;\boldsymbol{z});\boldsymbol{z})\big|_{\bar{\boldsymbol{\theta}}=\bar{\boldsymbol{\theta}}^t}\right] \tag{2.25}$$

**Algorithm 3** DRFL

**Input**: Initial guess $\bar{\boldsymbol{\theta}}^1$, a set of workers $\mathcal{K}$ with data samples $\{\boldsymbol{z}_n(k)\}_{n=1}^N$ per worker $k \in \mathcal{K}$, step size sequence $\{\alpha_t, \eta_t > 0\}_{t=1}^T$
**Output**: $\bar{\boldsymbol{\theta}}^{T+1}$
For$t = 1, \ldots, T$
**Each worker**:
Samples a minibatch $\mathcal{B}^t(k)$ of samples
Given $\bar{\boldsymbol{\theta}}^t$ and $\boldsymbol{z} \in \mathcal{B}^t(k)$, forms local perturbed loss

$$\psi_k(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}; \boldsymbol{z}) := \ell(\bar{\boldsymbol{\theta}}^t; \boldsymbol{\zeta}) + \gamma^t(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta}))$$

Lazily maximizes $\psi_k(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}; \boldsymbol{z})$ over $\boldsymbol{\zeta}$ to find

$$\boldsymbol{\zeta}(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}) = \boldsymbol{z} + \eta_t \nabla_{\boldsymbol{\zeta}} \psi_k(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}; \boldsymbol{z})|_{\boldsymbol{\zeta}=\boldsymbol{z}}$$

Computes stochastic gradient

$$\frac{1}{|\mathcal{B}^t(k)|} \sum_{\boldsymbol{z} \in \mathcal{B}^t(k)} \nabla_{\bar{\boldsymbol{\theta}}} \psi_k(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}); \boldsymbol{z})\big|_{\bar{\boldsymbol{\theta}}=\bar{\boldsymbol{\theta}}^t}$$

and uploads to server
**Server**:
Updates $\bar{\boldsymbol{\theta}}^t$ according to (2.25)
Broadcasts $\bar{\boldsymbol{\theta}}^{t+1}$ to workers

which is then broadcast to all workers to begin a new round of local updates. Our DRFL approach is tabulated in Alg. 3.

## 2.8 Numerical Tests

To assess the performance in the presence of distribution drifts and adversarial perturbations, we will rely on empirical classification of standard MNIST and Fashion- (F-)MNIST datasets. Specifically, we compare performance using models trained with empirical risk minimization (ERM), the fast-gradient method (FGSM) [64], its iterated variant (IFGM) [94], and the Wasserstein robust method (WRM) [169]. We further evaluate the testing performance using the projected gradient descent (PGD) attack [118]. We first test the performance of SPGD with $\epsilon$-accurate oracle, and the SPGDA algorithm on standard classification tasks.

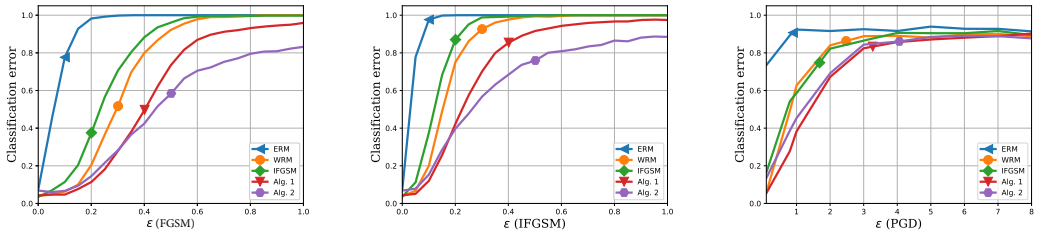



Figure 2.1: Misclassification error rate for different training methods using MNIST dataset; Left: FGSM attack, Middle: IFGSM attack, Right: PGD attack

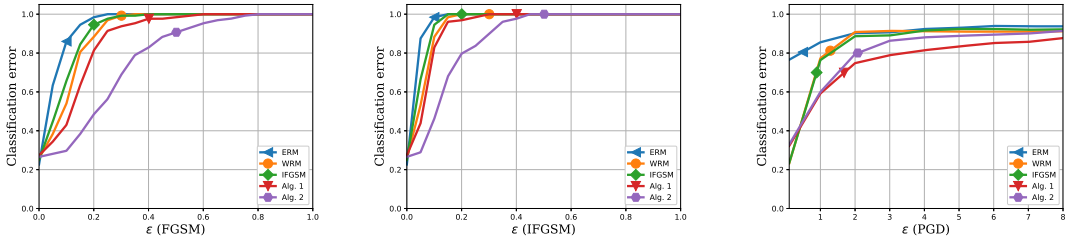



Figure 2.2: Misclassification error rate for different training methods using F-MNIST dataset; Left: FGSM attack, Middle: IFGSM attack, Right: PGD attack

### 2.8.1 SPGD with $\epsilon$-accurate oracle and SPGDA

The FGSM attack performs one step gradient update along the direction of the gradient's sign to find an adversarial sample; that is,

$$\boldsymbol{x}_{\text{adv}} = \text{Clip}_{[-1,1]}\{\boldsymbol{x} + \epsilon_{\text{adv}}\text{sign}(\nabla \ell_{\boldsymbol{x}}(\boldsymbol{\theta}; (\boldsymbol{x}, y)))\} \tag{2.26}$$

where $\epsilon_{\text{avd}}$ controls the maximum $\ell_\infty$ perturbation of adversarial samples. The element-wise $\text{Clip}_{[a,b]}\{\}$ operator forces its input to reside in the prescribed range $[-1, 1]$. By running $T_{\text{adv}}$ iterations of (2.26) iterative (I) FGSM attack samples are generated [64]. Starting with an initialization $\boldsymbol{x}^0_{\text{adv}} = \boldsymbol{x}$, and considering the $\ell_\infty$ norm, the PGD attack iterates [118]

$$\boldsymbol{x}^{t+1}_{\text{adv}} = \Pi_{\mathcal{B}_\epsilon(\boldsymbol{x}^t_{\text{adv}})}\Big\{\boldsymbol{x}^t_{\text{adv}} + \alpha\text{sign}(\nabla \ell_{\boldsymbol{x}}(\boldsymbol{\theta}; (\boldsymbol{x}^t_{\text{adv}}, y)))\Big\} \tag{2.27}$$

for $T_{\rm adv}$ steps, where $\Pi$ denotes projection onto the ball $\mathcal{B}_\epsilon(\boldsymbol{x}_{\rm adv}^t) := \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_{\rm adv}^t\|_\infty \leq \epsilon_{\rm adv}\}$, and $\alpha > 0$ is the stepsize set to 1 in our experiments. We use $T_{\rm adv} = 10$ iterations for all iterative methods both in training and attack samples. The PGD can also be interpreted as an iterative algorithm that solves the optimization problem $\max_{\boldsymbol{x}'} \ell(\boldsymbol{\theta}; (\boldsymbol{x}', y))$ subject to $\|\boldsymbol{x}' - \boldsymbol{x}\|_{\ell_\infty} \leq \alpha$. The Wasserstein attack on the other hand, generates adversarial samples by solving a perturbed training loss with an $\ell_2$-based transportation cost associated with the Wasserstein distance between the training and adversarial data distributions [169].

For the MNIST and F-MNIST datasets, a convolutional neural network (CNN) classifier consisting of $8 \times 8$, $6 \times 6$, and $5 \times 5$ filter layers with rectified linear units (ReLU) and the same padding, is used. Its first, second, and third layers have $64$, $128$, and $128$ channels, respectively, followed by a fully connected layer, and a softmax layer at the output.

CNNs with the same architecture are trained, using different adversarial samples. Specifically, to train a Wasserstein robust CNN model (WRM), $\gamma = 1$ was used to generate Wasserstein adversarial samples, $\epsilon_{\rm adv}$ was set to $0.1$ for the other two methods, and $\rho = 25$ was used to define the uncertainty set for both Algs. 1 and 2. Unless otherwise noted, we set the batch size to $128$, the number of epochs to 30, the learning rates to $\alpha = 0.001$ and $\eta = 0.02$, and used the Adam optimizer [88]. Fig. 2.1(Left) shows the classification error on the MNIST dataset. The error rates were obtained using testing samples generated according to the FGSM method with $\epsilon_{\rm adv}$. Clearly all training methods outperform ERM, and our proposed Algs. 1 and 2 offer improved performance over competing alternatives. The testing accuracy of all methods using samples generated according to an IFGSM attack is presented in Fig. 2.1(Middle). Likewise, Algs. 1 and 2 outperform other methods in this case. Fig. 2.1(Right) depicts the testing accuracy of the considered methods under different levels of a PGD attack. The plots in Fig. 2.1 showcase the improved performance obtained by CNNs trained using Algs. 1 and 2.

The F-MNIST article image dataset is used in our second experiment. Similar to MNIST dataset, each example in F-MNIST is also a $28 \times 28$ gray-scale image, associated with a label from $10$ classes. F-MNIST is a modern replacement for the original MNIST dataset for benchmarking machine learning algorithms. Using CNNs with the same architectures as before, the classification error is depicted for different training methods in Fig. 2.2. Three different attacks, namely FGSM, IFGSM, and PGD are used during testing. The proposed SPGD and SPGDA algorithms outperform the other methods, verifying the superiority of Algs. 1 and 2 in terms of yielding robust models.

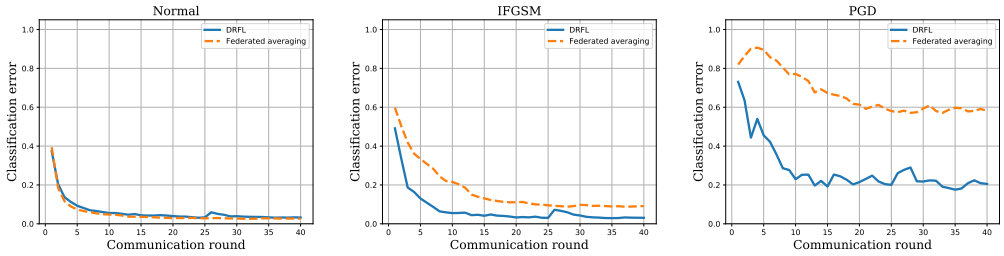
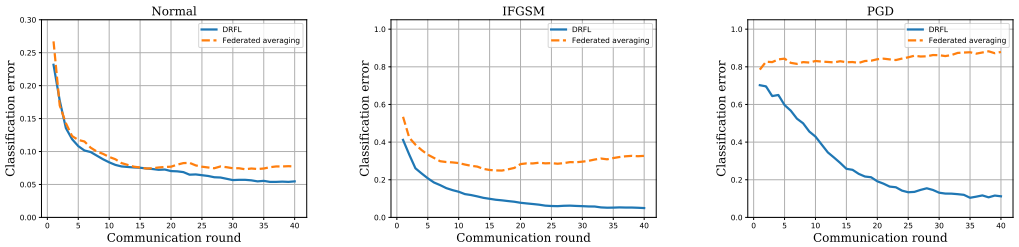



Figure 2.3: Distributionally robust federated learning for image classification using the non-i.i.d. F-MNIST dataset; Left: No attack, Middle: IFGSM attack, Right: PGD attack




(c)
t

Figure 2.4: Federated learning for image classification using the MNIST dataset; Left: No attack, Middle: IFGSM attack, Right: PGD attack

### 2.8.2 Distributionally robust federated learning

To validate the performance of our DRFL algorithm, we considered an FL environment consisting of a server and 10 workers, with local batch size 64, and assigned to every worker an equal-sized subset of training data containing i.i.d. samples from 10 different classes. All workers participated in each communication round. To benchmark the DRFL, we simulated the federated averaging method [123]. The testing accuracy on the MNIST dataset per communication round using clean (normal) images is depicted in Fig. 2.4. Clearly, both DRFL and federated averaging algorithms exhibit reasonable performance when the data is not corrupted. The performance is further tested against IFGSM and PGD attacks with a fixed $\epsilon_{\text{adv}} = 0.1$ during each communication round, and the corresponding misclassification error rates are shown in Figs. 2.4(Middle) and 2.4(Right), respectively. The classification performance using federated averaging does not improve in Fig. 2.4(Middle), whereas the DRFL performance keeps improving across

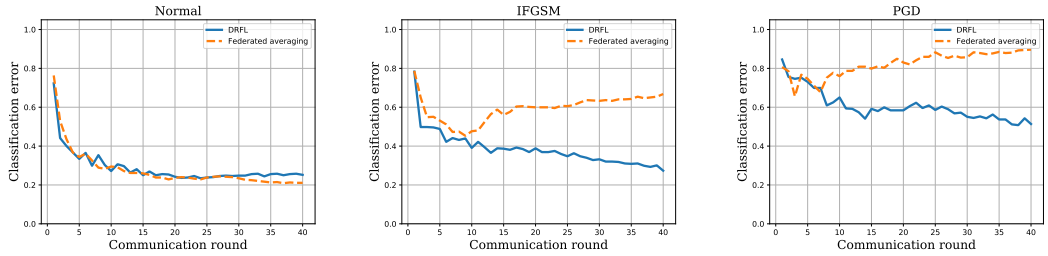



Figure 2.5: Distributionally robust federated learning for image classification using F-MNIST dataset; Left: No attack, Middle: IFGSM attack, Right: PGD attack

communication rounds. This is a direct consequence of accounting for the data uncertainties during the learning process. Moreover, Fig. 2.4(Right) showcases that the federated averaging becomes even worse as the model gets progressively trained under the PGD attack. This indeed motivates our DRFL approach when data are from untrusted entities with possibly adversarial input perturbations. Similarly, Fig. 2.5 depicts the misclassification rate of the proposed DRFL method compared with federated averaging, when using the F-MNIST dataset.

As the distribution of data across devices may influence performance, we further considered a biased local data setting. In particular, each worker $k = 1, \ldots, 10$ has data from only one class, so the distributions at workers are highly perturbed, and data stored across workers are thus non-i.i.d. The testing error rate for normal inputs is reported in Fig. 2.3, while the test error against adversarial attacks is depicted in Figs. 2.3(Middle) and 2.3(Right). This additional set of tests shows that having distributional shifts across workers can indeed enhance testing performance when the samples are adversarially manipulated.

## 2.9 Conclusions

A framework to robustify parametric machine learning models against distributional uncertainties was put forth. The learning task was cast as a distributionally robust optimization problem, for which two scalable stochastic optimization algorithms were developed. The first algorithm relies on an $\epsilon$-accurate maximum-oracle to solve the inner convex subproblem, while the second approximates its solution via a single gradient ascent step. Convergence guarantees for both algorithms to a stationary point were obtained. The upshot of the proposed approach is that it is amenable to federated learning from unreliable datasets across multiple workers. The novel

DRFL algorithm ensures data privacy and integrity, while offering robustness with minimal computational and communication overhead. Numerical tests for classifying standard real images showcased the merits of the proposed algorithms against distributional uncertainties and adversaries. This work also opens up several interesting directions for future research, including distributionally robust deep reinforcement learning.

# Chapter 3

# Distributionally Robust Semi-Supervised Learning Over Graphs

## 3.1 Introduction

Building upon but going well beyond the scope of previous robust learning paradigms, the present Chapter puts forth a novel iterative semi-supervised learning (SSL) over graphs framework. Relations among data in real world applications can often be captured by graphs, for instance the analysis and inference tasks for social, brain, communication, biological, transportation, and sensor networks [167, 90]. In practice however, the data is only available for a subset of nodes, due to for example the cost, and computational or privacy constraints. Most of these applications however, deal with inference of processes across all the network nodes. Such semi-supervised learning (SSL) tasks over networks can be addressed by exploiting the underlying graph topology [35, 16, 111].

Graph neural networks (GNNs) are parametric models that combine graph-filters and topology information with point-wise nonlinearities, to form nested architectures to easily express the functions defined over graphs [218]. By exploiting the underlying irregular structure of network data, the GNNs enjoy lower computational complexity, less parameters for training, and improved generalization capabilities relative to traditional deep neural networks (DNNs),

making them appealing for learning over graphs [218, 196, 56].

Similar to other DNN models, GNNs are also susceptible to adversarial manipulated input data or, distributional uncertainties, such as mismatches between training and testing data distributions. For instance small perturbations to input data would significantly deteriorate the regression performance, or result in classification error [223, 80], just to name a few. Hence, it is critical to develop principled methods that can endow GNNs with robustness, especially in safety-critical applications, such as robotics [175], and transportation [217].

**Contributions.** This Chapter endows SSL over graphs using GNNs with *robustness* against distributional uncertainties and possibly adversarial perturbations. Assuming the data distribution lies inside a Wasserstein ball centered at empirical data distribution, we robustify the model by minimizing the worst expected loss over the considered ball, which is challenging to solve. Invoking recently developed strong duality results, we develop an equivalent unconstrained and tractable learning problem. [1]

## 3.2 Problem formulation

Consider a SSL task over a graph $\mathcal{G} := \{\mathcal{V}, \mathbf{W}\}$ with $N$ nodes, where $\mathcal{V} := \{1, \ldots, N\}$ denotes the vertex set, and $\mathbf{W}$ represents the $N \times N$ weighted adjacency matrix capturing node connectivity. The associated unnormalized graph Laplacian matrix of the undirected graph $\mathcal{G}$ is $\mathbf{L} := \mathbf{D} - \mathbf{A}$, where $\mathbf{D} := \text{diag}\{\mathbf{W}\mathbf{1}_N\}$, with $\mathbf{1}_N$ denoting the $N \times 1$ all-one column vector. Denote by matrix $\mathbf{X}_s \in \mathbb{R}^{N \times F}$ the nodal feature vectors sampled at instances $s = 1, 2, \cdots$, with $n$-th row $\mathbf{x}_{n,s}^\top := [\mathbf{X}_s]_{n:}$ representing a feature vector of length $F$ associated with node $n \in \mathcal{V}$, and $\top$ stands for transposition. In the given graph, the labels $\{y_{n,s}\}_{n \in \mathcal{O}_s}$ are given for *only a small subset* of nodes, where $\mathcal{O}_s$ represents the index set of *observed* nodes sampled at $s$, and $\mathcal{U}_s$ the index set of *unobserved* nodes.

Given $\{\mathbf{X}_s, \mathbf{y}_s\}$, where $\mathbf{y}_s$ is the vector of observed labels, the goal is to find the labels of unobserved nodes $\{y_{n,s}\}_{n \in \mathcal{U}_s}$. To this aim our objective is to learn a functional mapping $f(\mathbf{X}_s; \mathbf{W})$ that can infer the missing labels based on available information. Such a function can

[1] The results of this Chapter have been submitted in [149]

be learned by solving the following optimization problem (see e.g., [89] for more details)

$$\min_{f\in\mathcal{F}} \mathbb{E}\Big[\overbrace{\sum_{n\in\mathcal{O}_s}\|f(\mathbf{x}_n;\mathbf{W})-y_n\|^2}^{\mathcal{L}_0}+\lambda\overbrace{\sum_{n,n'}\mathbf{W}_{nn'}\|f(\mathbf{x}_n;\mathbf{W})-f(\mathbf{x}_{n'};\mathbf{W})\|^2}^{\mathcal{L}_{\rm reg}}\Big], \tag{3.1}$$

where $\mathcal{L}_0$ represents the supervised loss w.r.t. the observed part of the graph, $\mathcal{L}_{\rm reg}$ represents the Laplacian regularization term, $\mathcal{F}$ denotes the feasible set of functions that we can learn, and $\lambda \geq 0$ is a hyper parameter. The regularization term relies on the premise that connected nodes in the graph are likely to share similar labels. The expectation here is taken with respect to (w.r.t) the feature and label data generating distribution.

In this work, we first encode the graph structure using a GNN model denoted by $f(\mathbf{X};\boldsymbol{\theta},\mathbf{W})$, where $\boldsymbol{\theta}$ represents the model parameters. Such a parametric representation enables bypassing explicit graph-based regularization $\mathcal{L}_{\rm reg}$ represented in 3.1. The GNN model of $f(\cdot)$ relies on the weighted adjacency $\mathbf{W}$ and therefore can easily propagate information from observed nodes $\mathcal{O}_s$ to unobserved ones $\mathcal{U}_s$. In a nutshell, objective is to learn a parametric model by solving the following problem

$$\min_{\boldsymbol{\theta}\in\Theta}\ \mathbb{E}_{\{\mathbf{X},\mathbf{y}\}\sim P_0}\ \mathcal{L}_0\big(f(\mathbf{X},\boldsymbol{\theta};\mathbf{W}),\mathbf{y}\big) \tag{3.2}$$

where $\Theta$ is a feasible set, and $P_0$ is the feature and label data generating distribution. Despite restricting the modeling capacity through parameterizing $f(\cdot)$ with GNNs, we may infuse additional prior information into the sought formulation through exploiting the weighted adjacency matrix $\mathbf{W}$, which does not necessarily encode node similarities.

In practice, $P_0$ is typically unknown, instead some data samples $\{\mathbf{X}_s,\mathbf{y}_s\}_{s=1}^S$ are given. Upon replacing the nominal distribution with an empirical one, we arrive at the empirical loss minimization problem, that is $\min_{\boldsymbol{\theta}\in\Theta}\ S^{-1}\sum_{s=1}^S\ \mathcal{L}_0\big(f(\mathbf{X}_s,\boldsymbol{\theta};\mathbf{W}),\mathbf{y}_s\big)$. The model obtained by solving empirical risk minimization does not exhibit any robustness in practice, specifically if there is any mismatch between the training and testing data distributions. To endow robustness, we reformulate this learning problem in a fresh manner as described in ensuing section.

## 3.3 Distributionally robust learning

To endow robustness, we consider the following optimization problem

$$\min_{\boldsymbol{\theta}\in\Theta} \sup_{P\in\mathcal{P}} \; \mathbb{E}_{(\mathbf{X},\mathbf{y})\sim P} \, \mathcal{L}_0\big(f(\mathbf{X},\boldsymbol{\theta};\mathbf{W}),\mathbf{y}\big) \tag{3.3}$$

where $\mathcal{P}$ is a set of distributions centered around the *empirical data distribution* $\widehat{P}_0$. This novel reformulation in 3.3 yields a model that performs reasonably well among a continuum of distributions. Various ambiguity sets $\mathcal{P}$ can be considered in practice, and they lead to different robustness guarantees with different computational requirements. For instance momentum, KL divergence, statistical test, and Wasserstein distance-based sets are popular in practice; see also [20, 168, 19] and references therein. Among possible choices, we utilize the optimal transport theory and the Wasserstein distance to characterize the ambiguity set $\mathcal{P}$. As a result, we can offer a tractable solution for this problem, as delineated next.

To formalize our framework, let us first define the Wasserstein distance between two probability measures. To this aim, consider probability measures $P$ and $\widehat{P}$ supported on some set $\mathcal{X}$, and let $\Pi(P,\widehat{P})$ denote the set of joint measures (a.k.a coupling) defined over $\mathcal{X}\times\mathcal{X}$, with marginals $P$ and $\widehat{P}$, and let $c:\mathcal{X}\times\mathcal{X}\to[0,\infty)$ measure the transportation cost for a unit of mass from $\mathbf{X}\in\mathcal{X}$ in $P$ to $\mathbf{X}'\in\mathcal{X}$ in $\widehat{P}$. The so-called optimal transport problem is concerned with the minimum cost associated with transporting all the mass from $P$ to $\widehat{P}$ through finding the optimal coupling, i.e., $W_c(P,\widehat{P}) := \inf_{\pi\in\Pi} \mathbb{E}_\pi[c(\mathbf{X},\mathbf{X}')]$. If $c(\cdot,\cdot)$ satisfies the axioms of distance, then $W_c$ defines a distance on the space of probability measures. For instance, if $P$ and $\widehat{P}$ are defined over a Polish space equipped with metric $d$, then fixing $c(\mathbf{X},\mathbf{X}') = d^p(\mathbf{X},\mathbf{X}')$ for some $p\in[1,\infty)$ asserts that $W_c^{1/p}(P,\widehat{P})$ is the well-known Wasserstein distance of order $p$ between $P$ and $\widehat{P}$.

Using the Wasserstein distance, let us define the uncertainty set $\mathcal{P} := \{P | W_c(P,\widehat{P}_0) \le \rho\}$ to include all probability distribution functions (pdfs) having at most $\rho$-distance from $\widehat{P}_0$. Incorporating this ambiguity set into 3.3, the following robust surrogate is considered in this work

$$\min_{\boldsymbol{\theta}\in\Theta} \sup_{P\in\mathcal{P}} \quad \mathbb{E}_{(\mathbf{X},\mathbf{y})\sim P} \, \mathcal{L}_0\big(f(\mathbf{X},\boldsymbol{\theta};\mathbf{W}),\mathbf{y}\big), \quad \text{where } \mathcal{P} := \Big\{P \,|W_c(P,\widehat{P}_0) \le \rho\Big\}. \tag{3.4}$$

The inner supremum here goes after pdfs characterized by $\mathcal{P}$. Solving this optimization directly

over the infinite-dimensional space of distribution functions raises practical challenges. Fortunately, under some mild conditions over losses as well as transport costs, the inner maximization satisfies a strong duality condition (see [19] for a detailed discussions), which means the optimal objective of this inner maximization and its Lagrangian dual are equal. Enticingly, the dual reformulation involves optimization over only one-dimensional dual variable. These properties make it practically appealing to solve 5.37 directly in the dual domain. The following proposition highlights the strong duality result, whose proofs can be found in [20].

**Proposition 2.** *Under some mild conditions over the loss $\mathcal{L}_0(\cdot)$ and cost $c(\cdot)$, it holds that*

$$\sup_{P\in\mathcal{P}} \mathbb{E}_P\, \mathcal{L}_0\big(f(\mathbf{X},\boldsymbol{\theta};\mathbf{W}),\mathbf{y}\big) = \inf_{\gamma\geq 0} \frac{1}{S}\sum_{s=1}^{S} \sup_{\boldsymbol{\xi}\in\mathcal{X}} \{\mathcal{L}_0\big(f(\boldsymbol{\xi},\boldsymbol{\theta};\mathbf{W}),\mathbf{y}_s\big) + \gamma\,(\rho - c(\mathbf{X}_s,\boldsymbol{\xi}))\} \tag{3.5}$$

*where* $\mathcal{P} := \Big\{P\,|W_c(P,\widehat{P}_0) \leq \rho\Big\}$.

The right-hand side in 5.38 simply is the univariate dual reformulation of the primal problem represented in the left-hand side. Furthermore, different from the primal formulation, the expectation in the dual domain is replaced with the summation over available training data, rather than any $P \in \mathcal{P}$ that needs to be obtained by solving for the optimal $\pi \in \Pi$ to form $\mathcal{P}$. Because of these two properties, solving the dual problem is practically more appealing. Thus, hinging on Proposition 2, the following distributionally robust surrogate is considered in this work

$$\min_{\boldsymbol{\theta}\in\Theta}\ \inf_{\gamma\geq 0} \frac{1}{S}\sum_{s=1}^{S} \sup_{\boldsymbol{\xi}\in\mathcal{X}} \big\{\mathcal{L}_0\big(f(\boldsymbol{\xi},\boldsymbol{\theta};\mathbf{W}),\mathbf{y}_s\big) + \gamma(\rho - c(\mathbf{X}_s,\boldsymbol{\xi}))\big\} \tag{3.6}$$

This problem requires the supremum to be solved separately for each sample $\mathbf{X}_s$, which cannot be handled through existing methods. Our approach to address this relies on the structure of this problem to iteratively update parameters $\bar{\boldsymbol{\theta}} := [\boldsymbol{\theta}^\top, \gamma]^\top$ and $\boldsymbol{\xi}$. Specifically, we rely on Danskin's theorem to first maximize over $\boldsymbol{\xi}$, which results in a differentiable function of $\bar{\boldsymbol{\theta}}$, and then minimize the objective w.r.t. $\bar{\boldsymbol{\theta}}$ using gradient descent. However, to guarantee convergence to a stationary point and utilize Danskin's theorem, we need to make sure the inner maximization admits a unique solution (singleton). By choosing a strongly convex transportation cost such as $c(\mathbf{X},\boldsymbol{\xi}) := \|\mathbf{X}-\boldsymbol{\xi}\|_F^2$, and by selecting $\gamma \in \Gamma := \{\gamma|\gamma > \gamma_0\}$ with a large enough $\gamma_0$, we arrive at a strongly concave objective function for the maximization over $\boldsymbol{\xi}$. Since $\gamma$ is the dual

variable associated with the constraint in 5.37, having $\gamma \in \Gamma$ is tantamount to tuning $\rho$, which in turn *controls* the level of robustness. Replacing $\gamma \geq 0$ in 5.39 with $\gamma \in \Gamma$, our *robust model* can be obtained as the solution of

$$\min_{\boldsymbol{\theta}\in\Theta} \inf_{\gamma\in\Gamma} \frac{1}{S}\sum_{s=1}^{S} \sup_{\boldsymbol{\xi}\in\mathcal{X}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\xi}; \mathbf{X}_s) \tag{3.7}$$

where $\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\xi}; \mathbf{X}_s) = \mathcal{L}_0(f(\boldsymbol{\xi}, \boldsymbol{\theta}; \mathbf{W}), \mathbf{y}_s) + \gamma(\rho - c(\mathbf{X}_s, \boldsymbol{\xi}))$. Intuitively, input $\mathbf{X}_s$ in 5.40 is pre-processed by maximizing $\psi(\cdot)$ accounting for a perturbation. We iteratively solve 5.40, where after sampling a mini-batch of data, we first pre-process them by maximizing the function $\psi(\cdot)$. Then, we use a simple gradient descent to update $\bar{\boldsymbol{\theta}}$. Notice that the $\boldsymbol{\theta}$ inside function $\psi(\cdot)$, represents the weights of our considered GNN, whose details are provided next.

## 3.4 Graph neural networks

GNNs are parametric models to represent functional relationship for graph structured data. Specifically, the input to a GNN is a data matrix $\mathbf{X}$. Upon multiplying the input $\mathbf{X}$ by $\mathbf{W}$, features will diffuse over the graph, giving a new graph signal $\check{\mathbf{Y}} = \mathbf{W}\mathbf{X}$. To model feature propagation, one can also replace $\mathbf{W}$ with the (normalized) graph Laplacian or random walk Laplacian, since they will also preserve dependencies among nodal attributes.

During the diffusion process, the feature vector of each node is updated by a linear combination of its neighbors. Take the $n$-th node as an example, the shifted $f$-th feature $[\check{\mathbf{Y}}]_{nf}$ is obtained by $[\check{\mathbf{Y}}]_{nf} = \sum_{i=1}^{N}[\mathbf{W}]_{ni}[\mathbf{X}]_{if} = \sum_{i\in\mathcal{N}_n} w_{ni}x_i^f$, where $\mathcal{N}_n$ denotes the set of neighboring nodes for node $n$. The so-called convolution operation in GNNs utilizes topology to combine features, namely

$$[\mathbf{Y}]_{nd} := [\mathcal{H} \star \mathbf{X}; \mathbf{W}]_{nd} := \sum_{k=0}^{K-1} [\mathbf{W}^k\mathbf{X}]_{n:}[\mathbf{H}_k]_{:d} \tag{3.8}$$

where $\mathcal{H} := [\mathbf{H}_0 \ \cdots \ \mathbf{H}_{K-1}]$ with $\mathbf{H}_k \in \mathbb{R}^{F\times D}$ as filter coefficients; $\mathbf{Y} \in \mathbb{R}^{N\times D}$ the intermediate (hidden) matrix with $D$ features per node; and $\mathbf{W}^k\mathbf{X}$ as the linearly combined features of nodes within the $k$-hop neighborhood.

To construct a GNN with $L$ hidden layers, first let us denote by $\mathbf{X}_{l-1}$ the output of the $(l-1)$-th layer, which is also the $l$-th layer input for $l = 1, \ldots, L$, and $\mathbf{X}_0 = \mathbf{X}$ to represent the input matrix. The hidden $\mathbf{Y}_l \in \mathbb{R}^{N\times D_l}$ with $D_l$ features is obtained by applying the

graph convolution operation 5.30 at layer $l$, i.e., $[\mathbf{Y}_l]_{nd} = \sum_{k=0}^{K_l-1}[\mathbf{W}^k\mathbf{X}_{l-1}]_{n:}[\mathbf{H}_{lk}]_{:g}$, where $\mathbf{H}_{lk} \in \mathbb{R}^{F_{l-1}\times F_l}$ is the convolution coefficients for $k = 0, \ldots, K_l - 1$. The output at layer $l$ is constructed by applying a graph convolution followed by a point-wise nonlinear operation $\sigma_l(\cdot)$. The input-output relationship at layer $l$ can be represented succinctly by $\mathbf{X}_l = \sigma_l(\mathbf{Y}_l) = \sigma_l\left(\sum_{k=0}^{K_l-1}\mathbf{W}^k\mathbf{X}_{l-1}\mathbf{H}_{lk}\right)$. Using this mapping, GNNs use a nested architecture to represent nonlinear functional operator $\mathbf{X}_L = f(\mathbf{X}_0; \boldsymbol{\theta}, \mathbf{W})$ that maps the GNN input $\mathbf{X}_0$ to label estimates by taking into account the graph structure through $\mathbf{W}$. Specifically, in a compact representation we have that

$$f(\mathbf{X}_0; \boldsymbol{\theta}, \mathbf{W}) := \sigma_L\left(\sum_{k=0}^{K_L-1}\mathbf{W}^k\left(\ldots\left(\sigma_1\left(\sum_{k=0}^{K_1-1}\mathbf{W}^k\mathbf{X}_0\mathbf{H}_{1k}\right)\ldots\right)\right)\mathbf{H}_{Lk}\right) \tag{3.9}$$

where the parameter set $\boldsymbol{\theta}$ contains all the *trainable* filter weights $\{\mathbf{H}_{lk}, \forall l, k\}$.

## 3.5 Experiments

The performance of our novel distributionally robust GNN-based SSL is tested in a regression task using real load consumption data from the 2012 Global Energy Forecasting Competition (GEFC). Our objective here is to estimate only the amplitudes of voltages across all the nodes in a standard IEEE $118$-bus network. Utilizing this data set, the training and testing data are prepared by solving the so-called AC power flow equations using the MATPOWER toolbox [222].

The measurements $\mathbf{X}$ used include all active and reactive power injections, corrupted by small additive white Gaussian noise. Using MATPOWER we generated $1,000$ pairs of measurements and ground-truth voltages. We used $80\%$ of this data for training and the remaining for testing. Throughout the training, the Adam optimizer with a fixed learning rate $10^{-3}$ was employed to minimize the Hüber loss. Furthermore, the batch size was set to $32$ during all $100$ epochs.

To compare our method we employed $3$ different benchmarks, namely: i) the prox-linear network introduced in [214]; ii) a 6-layer vanilla feed-forward neural network (FNN); and, iii) an 8-layer FNN. Our considered GNN uses $K = 2$ with $D = 8$ hidden units with ReLU activation.

The first set of tests are carried out using normal (not-corrupted) data, where the results are depicted in Fig. 3.1. Here we show the estimated (normalized) voltage amplitudes at different nodes, namely $105$, and $20$ during the given time course. The black curve represents the ground

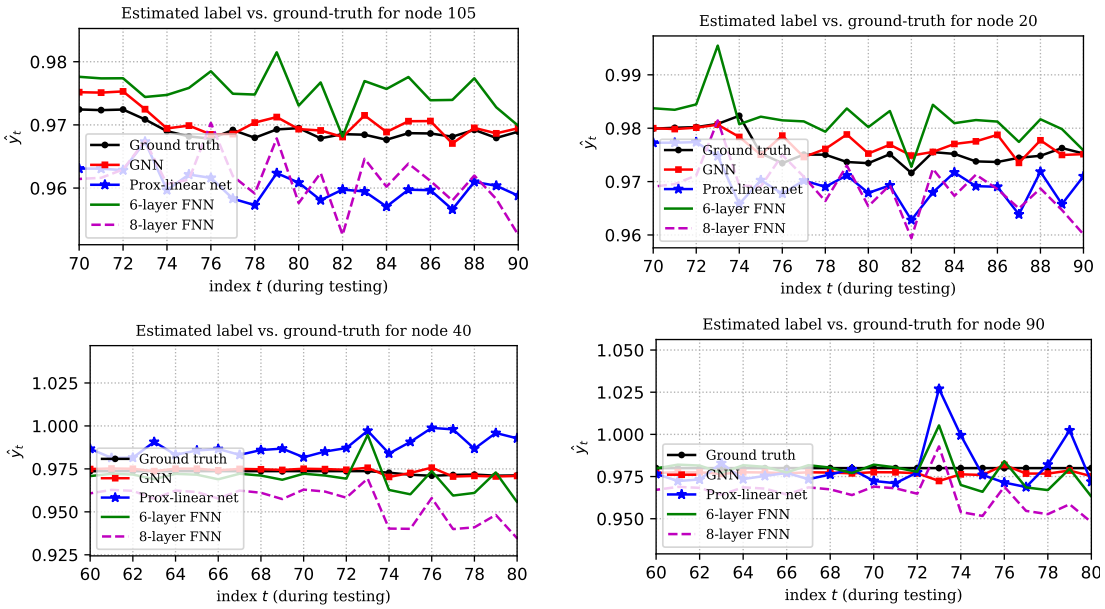



Figure 3.1: Performance during testing for both normal (a) - (b), and perturbed input features (c) - (d).

truth signal to be estimated. Clearly our GNN-based method outperforms alternative methods.

The second set of experiments are carried out over corrupted input signals, and the results are reported in Fig. 3.1. Specifically the training samples were generated according to $P_0$, but during testing samples were perturbed to satisfy the constraint $P \in \mathcal{P}$, that would yield the worst expected loss. Fig. 3.1 depicts the estimated signals across nodes $40$ and $90$. Here we fixed $\rho = 10$ and related hyper-parameters are tuned using grid search. As the plots showcase, the our proposed GNN-based robust method outperforms competing alternatives with corrupted inputs.

## 3.6 conclusions

This Chapter dealt with semi-supervised learning over graphs using GNNs. To account for uncertainties associated with data distributions, or adversarially manipulated input data, a principled robust learning framework was developed. Using the parametric models, we were able to reconstruct the unobserved nodal values. Experiments corroborated the outstanding performance of the novel method when the input data are corrupted.

# Chapter 4

# Deep and Reinforced Learning for Network Resource Management

## 4.1 Introduction

The advent of smart phones, tablets, mobile routers, and a massive number of devices connected through the Internet of Things (IoT) have led to an unprecedented growth in data traffic. Increased number of users trending towards video streams, web browsing, social networking and online gaming, have urged providers to pursue new service technologies that offer acceptable quality of experience (QoE). One such technology entails network densification by deploying small pico- and femto-cells, each serviced by a low-power, low-coverage, small basestation (SB). In this infrastructure, referred to as heterogeneous network (HetNet), SBs are connected to the backbone by a cheap 'backhaul' link. While boosting the network density by substantial reuse of scarce resources, e.g., frequency, the HetNet architecture is restrained by its low-rate, unreliable, and relatively slow backhaul links [5].

During peak traffic periods specially when electricity prices are also high, weak backhaul links can easily become congested–an effect lowering the QoE for end users. One approach to mitigate this limitation is to shift the excess load from peak periods to off-peak periods. Caching realizes this shift by fetching the "anticipated" popular contents, e.g., reusable video streams, during off-peak periods, storing this data in SBs equipped with memory units, and reusing them during peak traffic hours [138, 62, 185]. In order to utilize the caching capacity intelligently, a content-agnostic SB must rely on available observations to learn what and when to cache. To this

end, machine learning tools can provide 5G cellular networks with efficient caching, in which a "smart" caching control unit (CCU) can learn, track, and possibly adapt to the space-time popularities of reusable contents [138, 4].

**Prior work**. Existing efforts in 5G caching have focused on enabling SBs to learn unknown time-invariant content popularity profiles, and cache the most popular ones accordingly. A multi-armed bandit approach is reported in [22], where a reward is received when user requests are served via cache; see also [162] for a distributed, coded, and convexified reformulation. A belief propagation-based approach for distributed and collaborative caching is also investigated in [107]. Beyond [22], [162], and [107] that deal with deterministic caching, [36] and [23] introduce probabilistic alternatives. Caching, routing and video encoding are jointly pursued in [141] with users having different QoE requirements. However, a limiting assumption in [22, 162, 107, 36, 23, 141] pertains to space-time invariant modeling of popularities, which can only serve as a crude approximation for real-world requests. Indeed, temporal dynamics of local requests are prevalent due to user mobility, as well as emergence of new contents, or, aging of older ones. To accommodate dynamics, Ornstein-Uhlenbeck processes and Poisson shot noise models are utilized in [87] and [97], respectively, while context- and trend-aware caching approaches are investigated in [131] and [98].

Another practical consideration for 5G caching is driven by the fact that a relatively small number of users request contents during a caching period. This along with the small size of cells can challenge SBs from estimating accurately the underlying content popularities. To address this issue, a transfer-learning approach is advocated in [12], [18] and [97], to improve the time-invariant popularity profile estimates by leveraging prior information obtained from a surrogate (source) domain, such as social networks.

Finally, recent studies have investigated the role of coding for enhancing performance in cache-enabled networks [115, 117, 140]; see also [78], [79], and [47], where device-to-device "structureless" caching approaches are envisioned.

## 4.2 Our Contribution

The present chapter introduces a novel approach to account for space-time popularity of user requests by casting the caching task in a reinforcement learning (RL) framework. The CCU of the local SB is equipped with storage and processing units for solving the emerging RL

optimization in an online fashion. Adopting a Markov model for the popularity dynamics, a Q-learning caching algorithm is developed to learn the optimal policy even when the underlying transition probabilities are unknown.

Given the geographical and temporal variability of cellular traffic, global popularity profiles may not always be representative of local demands. To capture this, the proposed framework entails estimation of the popularity profiles both at the local as well as at the global scale. Specifically, each SB estimates its local vector of popularity profiles based on limited observations, and transmits it to the network operator, where an estimate of the global profile is obtained by aggregating the local ones. The estimate of the global popularity vector is then sent back to the SBs. The SBs can adjust the cost (reward) to trade-off tracking global trends versus serving local requests. [1]

To obtain a scalable caching scheme, a novel approximation of the proposed Q-learning algorithm is also developed. Furthermore, despite the stationarity assumption on the popularity Markov models, proper selection of stepsizes broadens the scope of the proposed algorithms for tracking demands even in non-stationary settings.

## 4.3 Modeling and problem statement

Consider a local section of a HetNet with a single SB connected to the backbone network through a low-bandwidth, high-delay, unreliable backhaul link. Suppose further that the SB is equipped with $M$ units to store contents (files) that are assumed for simplicity to have unit size; see Fig. 1. Caching will be carried out in a slotted fashion over slots $t = 1, 2, \ldots$, where at the beginning of each slot $t$, the CCU-enabled SB selects "intelligently" $M$ files from the total of $F \gg M$ available ones at the backbone, and prefetches them for possible use in subsequent slots. The slots may not be of equal length, as the starting times may be set a priori, for example at 3 AM, 11 AM, or 4 PM, when the network load is low; or, slot intervals may be dictated to CCU by the network operator on the fly. Generally, a slot starts when the network is at an off-peak period, and its duration coincides with the peak traffic time when pertinent costs of serving users are high.

During slot $t$, each user locally requests a subset of files from the set $\mathcal{F} := \{1, 2, \ldots, F\}$. If a requested file has been stored in the cache, it will be simply served locally, thus incurring

[1] The results of this Chapter have been published in [151, 156, 154, 150, 153, 155, 157, 152]

(almost) zero cost. Conversely, if the requested file is not available in the cache, the SB must fetch it from the cloud through its cheap backhaul link, thus incurring a considerable cost due to possible electricity price surges, processing cost, or the sizable delay resulting in low QoE and user dissatisfaction. The CCU wishes to intelligently select the cache contents so that costly services from the cloud be avoided as often as possible.

Let $\mathbf{a}(t) \in \mathcal{A}$ denote the $F \times 1$ binary *caching action vector* at slot $t$, where $\mathcal{A} := \{\mathbf{a}|\mathbf{a} \in \{0,1\}^F, \mathbf{a}^\top \mathbf{1} = M\}$ is the set of all feasible actions; that is, $[\mathbf{a}(t)]_f = 1$ indicates that file $f$ is cached for the duration of slot $t$, and $[\mathbf{a}(t)]_f = 0$ otherwise.

Depending on the received requests from locally connected users, the CCU computes the $F \times 1$-vector of *local popularity profile* $\mathbf{p}_L(t)$ per slot $t$, whose $f$-th entry indicates the expected local demand for file $f$, defined as

$$\left[\mathbf{p}_{\mathrm{L}}(t)\right]_f := \frac{\text{Number of local requests for } f \text{ at slot } t}{\text{Number of all local requests at slot } t}.$$

Similarly, suppose that the backbone network estimates the $F \times 1$ *global popularity profile* vector $\mathbf{p}_G(t)$, and transmits it to all CCUs.

Having observed the local and global user requests by the end of slot $t$, our overall system state is

$$\mathbf{s}(t) := \left[\mathbf{p}_G^\top(t), \mathbf{p}_L^\top(t), \mathbf{a}^\top(t)\right]^\top. \tag{4.1}$$

Being at slot $t-1$, our *objective* is to leverage historical observations of states, $\{\mathbf{s}(\tau)\}_{\tau=0}^{t-1}$, and pertinent costs in order to learn the optimal action for the next slot, namely $\mathbf{a}^*(t)$. Explicit expression of the incurred costs, and analytical formulation of the objective will be elaborated in the ensuing subsections.

### 4.3.1 Cost functions and caching strategies

Efficiency of a caching strategy will be measured by how well it utilizes the available storage of the local SB to keep the most popular files, versus how often local user requests are met via fetching through the more expensive backhaul link. The overall cost incurred will be modeled as the superposition of three types of costs.

The first type $c_{1,t}$ corresponds to the cost of refreshing the cache contents. In its general form, $c_{1,t}(\cdot)$ is a function of the upcoming action $\mathbf{a}(t)$, and available contents at the cache according to

current caching action $\mathbf{a}(t-1)$, where the subscript $t$ captures the possibility of a time-varying cost for refreshing the cache. A reasonable choice of $c_{1,t}(\cdot)$ is

$$c_{1,t}(\mathbf{a}(t), \mathbf{a}(t-1)) := \lambda_{1,t}\mathbf{a}^{\top}(t)\left[\mathbf{1} - \mathbf{a}(t-1)\right] \tag{4.2a}$$

which upon recalling that the action vectors $\mathbf{a}(t-1)$ and $\mathbf{a}(t)$ have binary $\{0,1\}$ entries, implies that $c_{1,t}$ counts the number of those files to be fetched and cached prior to slot $t$, which were not stored according to action $\mathbf{a}(t-1)$.

The second type of cost is incurred during the operational phase of slot $t$ to satisfy user requests. With $c_{2,t}(\mathbf{s}(t))$ denoting this type of cost, a prudent choice must: i) penalize requests for files already cached much less than requests for files not stored; and, ii) be a non-decreasing function of popularities $[\mathbf{p}_L]_f$. Here for simplicity, we assume that the transmission cost of cached files is relatively negligible, and choose

$$c_{2,t}(\mathbf{s}(t)) := \lambda_{2,t}\left[\mathbf{1} - \mathbf{a}(t)\right]^{\top}\mathbf{p}_L(t) \tag{4.2b}$$

which solely penalizes the non-cached files in descending order of their local popularities.

The third type of cost captures the "mismatch" between caching action $\mathbf{a}(t)$, and the global popularity profile $\mathbf{p}_{\mathrm{G}}(t)$. Indeed, it is reasonable to consider the global popularity of files as an acceptable representative of what the local profiles will look like in the near future; thus, keeping the caching action close to $\mathbf{p}_{\mathrm{G}}(t)$ may reduce future possible costs. Note also that a relatively small number of local requests may only provide a crude estimate of local popularities, while the global popularity profile can serve as side information in tracking the evolution of content popularities over the network. This has prompted the advocation of transfer learning approaches, where content popularities in a surrogate domain are utilized for improving estimates of popularity; see, e.g., [12] and [18]. However, this approach is limited by the degree the surrogate (source) domain, e.g., Facebook or Twitter, is a good representative of the target domain requests. When it is not, techniques will misguide caching decisions, while imposing excess processing overhead to the network operator or to the SB.

To account for this issue, we introduce the third type of cost as

$$c_{3,t}(\mathbf{s}(t)) := \lambda_{3,t}\left[\mathbf{1} - \mathbf{a}(t)\right]^{\top}\mathbf{p}_G(t) \tag{4.2c}$$

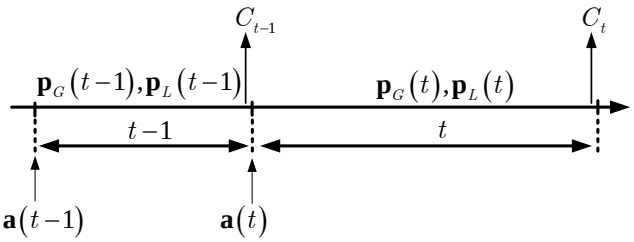



Figure 4.1: A schematic depicting the evolution of key quantities across time slots. Duration of slots can be unequal.

penalizing the files not cached according to the global popularity profile $\mathbf{p}_G(\cdot)$ provided by the network operator, thus promoting adaptation of caching policies close to global demand trends.

All in all, upon taking action $\mathbf{a}(t)$ at slot $t$, the *aggregate cost conditioned* on the popularity vectors revealed, can be expressed as (cf. (4.2a)-(4.2c))

$$\begin{aligned} & C_t\Big(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p}_{\mathrm{G}}(t), \mathbf{p}_{\mathrm{L}}(t)\Big) \\ & \quad := c_{1,t}\left(\mathbf{a}(t), \mathbf{a}(t-1)\right) + c_{2,t}\left(\mathbf{s}(t)\right) + c_{3,t}(\mathbf{s}(t)) \\ & \quad = \lambda_{1,t}\mathbf{a}^\top(t)(\mathbf{1}-\mathbf{a}(t-1)) + \lambda_{2,t}(\mathbf{1}-\mathbf{a}(t))^\top\mathbf{p}_L(t) \\ & \quad + \lambda_{3,t}(\mathbf{1}-\mathbf{a}(t))^\top\mathbf{p}_G(t). \end{aligned} \tag{4.3}$$

Weights $\lambda_{1,t}$, $\lambda_{2,t}$, and $\lambda_{3,t}$ control the relative significance of the corresponding summands, whose tuning influences the optimal caching policy at the CCU. As asserted earlier, the cache-refreshing cost at off-peak periods is considered to be less than fetching the contents during slots, which justifies the choice $\lambda_{1,t} \ll \lambda_{2,t}$. In addition, setting $\lambda_{3,t} \ll \lambda_{2,t}$ is of interest when the local popularity profiles are of acceptable accuracy, or, if tracking local popularities is of higher importance. In particular, setting $\lambda_{3,t} = 0$ corresponds to the special case where the caching cost is decoupled from the global popularity profile evolution. On the other hand, setting $\lambda_{2,t} \ll \lambda_{3,t}$ is desirable in networks where globally popular files are of high importance, for instance when users have high mobility and may change SBs rapidly, or, when a few local requests prevent the SB from estimating accurately the local popularity profiles. Fig. 4.1 depicts the evolution of popularity and action vectors along with the aggregate conditional costs across slots.

**Remark 1.** As with slot sizes, proper selection of $\lambda_{1,t}$, $\lambda_{2,t}$, and $\lambda_{3,t}$ is a design choice. Depending on how centralized or decentralized the network operation is desired to be, these parameters

may be selected autonomously by the CCUs or provided by the network operator in a centralized fashion. However, the overall approach requires the network service provider and the SBs to inter-operate by exchanging relevant information. On the one hand, estimating global popularities requires SBs to transmit their locally obtained $\mathbf{p}_L(t)$ to the network operator at the end of each slot. On the other hand, the network operator informs the CCUs of the global popularity $\mathbf{p}_G(t)$, and possibly weights $\lambda_{1,t}$, $\lambda_{2,t}$, and $\lambda_{3,t}$. By providing the network operator with means of parameter selection, a "master-slave" hierarchy emerges, which enables the network operator (master) to influence SBs (slaves) caching decisions, leading to a centrally controlled adjustment of caching policies. Interestingly, these few bytes of information exchanges occur once per slot and at off-peak instances, thus imposing negligible overhead to the system, while enabling a simple, yet practical and powerful optimal semi-distributed caching process; see Fig. 3.

### 4.3.2 Popularity profile dynamics

As depicted in Fig. 3, we will model user requests (and thus popularities) at both global and local scales using Markov chains. Specifically, global popularity profiles will be assumed generated by an underlying Markov process with $|\mathcal{P}_G|$ states collected in the set $\mathcal{P}_G := \left\{\mathbf{p}_G^1, \ldots, \mathbf{p}_G^{|\mathcal{P}_G|}\right\}$; and likewise for the set of all local popularity profiles $\mathcal{P}_L := \left\{\mathbf{p}_L^1, \ldots, \mathbf{p}_L^{|\mathcal{P}_L|}\right\}$. Although $\mathcal{P}_G$ and $\mathcal{P}_L$ are known, the underlying transition probabilities of the two Markov processes are considered unknown.

Given $\mathcal{P}_G$ and $\mathcal{P}_L$ as well as feasible caching decisions in set $\mathcal{A}$, the overall set of states in the network is

$$\mathcal{S} := \left\{\mathbf{s} | \mathbf{s} = [\mathbf{p}_G^\top, \mathbf{p}_L^\top, \mathbf{a}^\top]^\top, \mathbf{p}_G \in \mathcal{P}_G \, , \mathbf{p}_L \in \mathcal{P}_L, \mathbf{a} \in \mathcal{A}\right\}.$$

The lack of knowledge on transition probabilities of the underlying Markov chains motivates well our ensuing RL-based approach, where the learner seeks the optimal policy by interactively making sequential decisions, and observing the corresponding costs. The caching task is formulated in the following subsection, and an efficient solver is developed to cope with the "curse of dimensionality" typically emerging with RL problems [173].

### 4.3.3 Reinforcement learning formulation

As showing in Fig. 2, the CCU takes caching action $\mathbf{a}(t)$, at the beginning of slot $t$, and by the end of slot $t$, the profiles $\mathbf{p}_G(t)$ and $\mathbf{p}_L(t)$ become available, so that the system state is updated to $\mathbf{s}(t)$, and the conditional cost $C_t\Big(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t)\Big)$ is revealed. Given the random nature of user requests locally and globally, $C_t$ in (4.3) is a random variable with mean

$$\begin{aligned}&\overline{C}_t\left(\mathbf{s}(t-1), \mathbf{a}(t)\right) \\ &:= \mathbb{E}_{\mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t)}\Big[C_t\left(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t)\right)\Big] \\ &= \lambda_1 \mathbf{a}^\top(t)\left[\mathbf{1} - \mathbf{a}(t-1)\right] + \lambda_2 \mathbb{E}\left[\left(\mathbf{1} - \mathbf{a}(t)\right)^\top \mathbf{p}_L(t)\right] \\ &+ \lambda_3 \mathbb{E}\left[\left(\mathbf{1} - \mathbf{a}(t)\right)^\top \mathbf{p}_G(t)\right]\end{aligned} \tag{4.4}$$

where the expectation is taken with respect to (wrt) $\mathbf{p}_L(t)$ and $\mathbf{p}_G(t)$, while the weights are selected as $\lambda_{1,t} = \lambda_1$, $\lambda_{2,t} = \lambda_2$, and $\lambda_{3,t} = \lambda_3$ for simplicity.

Let us now define the policy function $\pi : \mathcal{S} \to \mathcal{A}$, which maps any state $\mathbf{s} \in \mathcal{S}$ to the action set. Under policy $\pi(\cdot)$, for the current state $\mathbf{s}(t)$, caching is carried out via action $\mathbf{a}(t+1) = \pi(\mathbf{s}(t))$ dictating what files to be stored for the $(t+1)$-st slot. Caching performance is measured through the so-termed state value function

$$V_\pi\left(\mathbf{s}(t)\right) := \lim_{T\to\infty} \mathbb{E}\left[\sum_{\tau=t}^{T} \gamma^{\tau-t} \overline{C}\left(\mathbf{s}\left[\tau\right], \pi\left(\mathbf{s}\left[\tau\right]\right)\right)\right] \tag{4.5}$$

which is the total average cost incurred over an infinite time horizon, with future terms discounted by factor $\gamma \in [0, 1)$. Since taking action $\mathbf{a}(t)$ influences the SB state in future slots, future costs are always affected by past and present actions. Discount factor $\gamma$ captures this effect, whose tuning trades off current versus future costs. Moreover, $\gamma$ also accounts for modeling uncertainties, as well as imperfections, or dynamics. For instance, if there is ambiguity about future costs, or if the system changes very fast, setting $\gamma$ to a small value enables one to prioritize current costs, whereas in a stationary setting one may prefer to demote future costs through a larger $\gamma$.

The objective of this paper is to find the optimal policy $\pi^*$ such that the average cost of any

state $\mathbf{s}$ is minimized (cf. (4.5))

$$\pi^* = \arg\min_{\pi\in\Pi} V_\pi(\mathbf{s}), \quad \forall \mathbf{s} \in \mathcal{S} \tag{4.6}$$

where $\Pi$ denotes the set of all feasible policies.

The optimization in (4.6) is a sequential decision making problem. In the ensuing section, we present optimality conditions (known as Bellman equations) for our problem, and introduce a Q-learning approach for solving (4.6).

## 4.4 Optimality conditions

Bellman equations, also known as dynamic programming equations, provide necessary conditions for optimality of a policy in a sequential decision making problem. Being at the $(t-1)$st slot, let $[\boldsymbol{P}^a]_{\mathbf{s}\mathbf{s}'}$ denote the transition probability of going from the current state $\mathbf{s}$ to the next state $\mathbf{s}'$ under action $\mathbf{a}$; that is,

$$[\boldsymbol{P}^a]_{\mathbf{s}\mathbf{s}'} := \Pr\Big\{\mathbf{s}(t) = \mathbf{s}' \Big| \mathbf{s}(t-1) = \mathbf{s}, \pi(\mathbf{s}(t-1)) = \mathbf{a}\Big\}.$$

Bellman equations express the state value function by (4.5) in a recursive fashion as [173, pg. 47]

$$V_\pi(\mathbf{s}) = \overline{C}(\mathbf{s}, \pi(\mathbf{s})) + \gamma \sum_{\mathbf{s}'\in\mathcal{S}} [\boldsymbol{P}^{\pi(\mathbf{s})}]_{\mathbf{s}\mathbf{s}'} V_\pi(\mathbf{s}') \ , \forall \mathbf{s}, \mathbf{s}' \tag{4.7}$$

which amounts to the superposition of $\overline{C}$ plus a discounted version of future state value functions under a given policy $\pi$. Specifically, after dropping the current slot index $t-1$ and indicating with prime quantities of the next slot $t$, $\overline{C}$ in (4.4) can be written as

$$\overline{C}(\mathbf{s}, \pi(\mathbf{s})) = \sum_{\mathbf{s}':=[\mathbf{p}'_{\mathbf{G}}, \mathbf{p}'_{\mathbf{L}}, \mathbf{a}']\in\mathcal{S}} [\boldsymbol{P}^{\pi(\mathbf{s})}]_{\mathbf{s}\mathbf{s}'} C\Big(\mathbf{s}, \pi(\mathbf{s}) \Big| \mathbf{p}'_{\mathbf{G}}, \mathbf{p}'_{\mathbf{L}}\Big)$$

where $C\Big(\mathbf{s}, \pi(\mathbf{s}) \Big| \mathbf{p}'_{\mathbf{G}}, \mathbf{p}'_{\mathbf{L}}\Big)$ is found as in (4.3). It turns out that, with $[\boldsymbol{P}^a]_{\mathbf{s}\mathbf{s}'}$ given $\forall \mathbf{s}, \mathbf{s}'$, one can readily obtain $\{V_\pi(\mathbf{s}), \forall \mathbf{s}\}$ by solving (4.7), and eventually the optimal policy $\pi^*$ in (4.9) using the so-termed policy iteration algorithm [173, pg. 79]. To outline how this algorithm works in our context, define the state-action value function that we will rely on under policy $\pi$ [173, pg.

62]

$$Q_\pi\left(\mathbf{s},\mathbf{a}'\right) := \overline{C}\left(\mathbf{s},\mathbf{a}'\right) + \gamma \sum_{\mathbf{s}'\in\mathcal{S}} [\boldsymbol{P}^{\mathbf{a}'}]_{\mathbf{s}\mathbf{s}'} V_\pi\left(\mathbf{s}'\right). \tag{4.8}$$

Commonly referred to as the "Q-function," $Q_\pi(\mathbf{s},\boldsymbol{\alpha})$ basically captures the expected current cost of taking action $\boldsymbol{\alpha}$ when the system is in state $\mathbf{s}$, followed by the discounted value of the future states, provided that the future actions are taken according to policy $\pi$.

In our setting, the policy iteration algorithm initialized with $\pi_0$, proceeds with the following updates at the $i$th iteration.

- **Policy evaluation**: Determine $V_{\pi_i}(\mathbf{s})$ for all states $\mathbf{s}\in\mathcal{S}$ under the current (fixed) policy $\pi_i$, by solving the system of linear equations in (4.7) $\forall\mathbf{s}$.
- **Policy update**: Update the policy using

$$\pi_{i+1}(\mathbf{s}) := \arg\max_{\boldsymbol{\alpha}} Q_{\pi_i}(\mathbf{s},\boldsymbol{\alpha}), \quad \forall \mathbf{s}\in\mathcal{S}.$$

The policy evaluation step is of complexity $\mathcal{O}(|\mathcal{S}|^3)$, since it requires matrix inversion for solving the linear system of equations in (4.7). Furthermore, given $V_{\pi_i}(\mathbf{s})$ $\forall\mathbf{s}$, the complexity of the policy update step is $\mathcal{O}(|\mathcal{A}||\mathcal{S}|^2)$, since the Q-values must be updated per state-action pair, each subject to $|\mathcal{S}|$ operations; see also (4.8). Thus, the per iteration complexity of the policy iteration algorithm is $\mathcal{O}(|\mathcal{S}|^3 + |\mathcal{A}||\mathcal{S}|^2)$. Iterations proceed until convergence, i.e., $\pi_{i+1}(\mathbf{s}) = \pi_i(\mathbf{s})$, $\forall\mathbf{s}\in\mathcal{S}$.

Clearly, the policy iteration algorithm relies on knowing $[\boldsymbol{P}^{\mathbf{a}}]_{\mathbf{s}\mathbf{s}'}$, which is typically not available in practice. This motivates the use of adaptive dynamic programming (ADP) that learn $[\boldsymbol{P}^{\mathbf{a}}]_{\mathbf{s}\mathbf{s}'}$ for all $\mathbf{s},\mathbf{s}'\in\mathcal{S}$, and $\mathbf{a}\in\mathcal{A}$, as iterations proceed [148, pg. 834]. Unfortunately, ADP algorithms are often very slow and impractical, as they must estimate $|\mathcal{S}|^2\times|\mathcal{A}|$ probabilities. In contrast, the Q-learning algorithm elaborated next finds the optimal $\pi^*$ as well as $V_\pi(\mathbf{s})$, while circumventing the need to estimate $[\boldsymbol{P}^a]_{\mathbf{s}\mathbf{s}'}$, $\forall\mathbf{s},\mathbf{s}'$; see e.g., [173, pg. 140].

### 4.4.1 Optimal caching via Q-learning

Q-learning is an online RL scheme to jointly infer the optimal policy $\pi^*$, and estimate the optimal state-action value function $Q^*(\mathbf{s},\mathbf{a}') := Q_{\pi^*}(\mathbf{s},\mathbf{a}')$ $\forall\mathbf{s},\mathbf{a}'$. Utilizing (4.7) for the optimal policy $\pi^*$, it can be shown that [173, pg. 67]

$$\pi^*(\mathbf{s}) = \arg\min_{\boldsymbol{\alpha}} \; Q^*(\mathbf{s}, \boldsymbol{\alpha}), \quad \forall \mathbf{s} \in \mathcal{S}. \tag{4.9}$$

The Q-function and $V(\cdot)$ under $\pi^*$ are related by

$$V^*(\mathbf{s}) := V_{\pi^*}(\mathbf{s}) = \min_{\boldsymbol{\alpha}} Q^*(\mathbf{s}, \boldsymbol{\alpha}) \tag{4.10}$$

which in turn yields

$$Q^*\left(\mathbf{s}, \mathbf{a}'\right) = \overline{C}\left(\mathbf{s}, \mathbf{a}'\right) + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} [\boldsymbol{P}^{\mathbf{a}}]_{\mathbf{s}\mathbf{s}'} \min_{\boldsymbol{\alpha} \in \mathcal{A}} Q^*\left(\mathbf{s}', \boldsymbol{\alpha}\right). \tag{4.11}$$

Capitalizing on the optimality conditions (4.9)-(4.11), an online Q-learning scheme for caching is listed under Alg. 1. In this algorithm, the agent updates its estimated $\hat{Q}(\mathbf{s}(t-1), \mathbf{a}(t))$ as $C\Big(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p}_{\mathrm{G}}(t), \mathbf{p}_{\mathrm{L}}(t)\Big)$ is observed. That is, given $\mathbf{s}(t-1)$, Q-learning takes action $\mathbf{a}(t)$, and upon observing $\mathbf{s}(t)$, it incurs cost $C\Big(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p}_{\mathrm{G}}(t), \mathbf{p}_{\mathrm{L}}(t)\Big)$. Based on the instantaneous error

$$\begin{aligned}\varepsilon\left(\mathbf{s}(t-1), \mathbf{a}(t)\right) :=& \frac{1}{2}\Big(C\left(\mathbf{s}(t-1), \mathbf{a}(t)\right) + \gamma \min_{\boldsymbol{\alpha}} \widehat{Q}\left(\mathbf{s}(t), \boldsymbol{\alpha}\right) \\ & - \widehat{Q}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)\Big)^2\end{aligned} \tag{4.12}$$

the Q-function is updated using stochastic gradient descent as

$$\begin{aligned}&\hat{Q}_t\left(\mathbf{s}(t-1), \mathbf{a}(t)\right) = (1-\beta_t)\hat{Q}_{t-1}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right) + \\ &\beta_t\Big[C\left(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p}_{\mathrm{G}}(t), \mathbf{p}_{\mathrm{L}}(t)\right) + \gamma \min_{\boldsymbol{\alpha}} \hat{Q}_{t-1}\left(\mathbf{s}(t), \boldsymbol{\alpha}\right)\Big]\end{aligned}$$

while keeping the rest of the entries in $\hat{Q}_t(\cdot, \cdot)$ unchanged.

Regarding convergence of the Q-learning algorithm, a necessary condition ensuring $\hat{Q}_t\left(\cdot, \cdot\right) \to Q^*\left(\cdot, \cdot\right)$, is that all state-action pairs must be continuously updated. Under this and the usual stochastic approximation conditions that will be specified later, $\hat{Q}_t\left(\cdot, \cdot\right)$ converges to $Q^*\left(\cdot, \cdot\right)$ with probability 1; see [24] for a detailed description.

To meet the requirement for continuous updates, Q-learning utilizes a probabilistic exploration-exploitation approach to selecting actions. At slot $t$, exploitation happens with probability $1-\epsilon_t$ through the action $\mathbf{a}(t) = \arg\min_{\boldsymbol{\alpha} \in \mathcal{A}} \hat{Q}_{t-1}(\mathbf{s}(t-1), \boldsymbol{\alpha})$, while the exploration happens with

**Algorithm 4** Caching via Q-learning at CCU

**Initialize** $\mathbf{s}(0)$ randomly and $\hat{Q}_0(\mathbf{s},\mathbf{a})=0\ \forall \mathbf{s},\mathbf{a}$
For $t=1,2,...$
Take action $\mathbf{a}(t)$ chosen probabilistically by

$$\mathbf{a}(t)=\begin{cases}\arg\min_{\mathbf{a}}\hat{Q}_{t-1}\left(\mathbf{s}(t-1),\mathbf{a}\right) & \text{w.p.} \quad 1-\epsilon_t\\ \text{random } \mathbf{a}\in\mathcal{A} & \text{w.p.} \quad \epsilon_t\end{cases}$$

$\mathbf{p}_L(t)$ and $\mathbf{p}_G(t)$ are revealed based on user requests
Set $\mathbf{s}(t):=\left[\mathbf{p}_G^\top(t),\mathbf{p}_L^\top(t),\mathbf{a}(t)^\top\right]^\top$
Incur cost $C\Big(\mathbf{s}(t-1),\mathbf{a}(t)\Big|\mathbf{p}_{\mathrm{G}}(t),\mathbf{p}_{\mathrm{L}}(t)\Big)$ Update

$$\begin{aligned}\hat{Q}_t\left(\mathbf{s}(t-1),\mathbf{a}(t)\right)=(1-\beta_t)\hat{Q}_{t-1}\left(\mathbf{s}(t-1),\mathbf{a}(t)\right)\\ +\beta_t\Big[C\left(\mathbf{s}(t-1),\mathbf{a}(t)\Big|\mathbf{p}_{\mathrm{G}}(t),\mathbf{p}_{\mathrm{L}}(t)\right)\\ +\gamma\min_{\boldsymbol{\alpha}}\hat{Q}_{t-1}\left(\mathbf{s}(t),\boldsymbol{\alpha}\right)\Big]\end{aligned}\tag{4.13}$$

probability $\epsilon_t$ through a random action $\mathbf{a}\in\mathcal{A}$. Parameter $\epsilon_t$ trades off exploration for exploitation, and its proper selection guarantees a necessary condition for convergence. During initial iterations or when the CCU observes considerable shifts in content popularities, setting $\epsilon_t$ high promotes exploration in order to learn the underlying dynamics. On the other hand, in stationary settings and once "enough" observations are made, small values of $\epsilon_t$ promote exploiting the learned $\hat{Q}_{t-1}(\cdot,\cdot)$ by taking the estimated optimal action $\arg\min_{\boldsymbol{\alpha}}\hat{Q}_{t-1}\left(\mathbf{s}(t),\boldsymbol{\alpha}\right)$.

Regarding stochastic approximation conditions, the stepsize sequence $\{\beta_t\}_{t=1}^{\infty}$ must obey $\sum_{t=1}^{\infty}\beta_t=\infty$ and $\sum_{t=1}^{\infty}\beta_t^2<\infty$ [24], both of which are satisfied by e.g., $\beta_t=1\,/\,t$. However, with a selection of constant stepsize $\beta_t=\beta$, the mean-square error (MSE) of $\hat{Q}_{t+1}(\cdot,\cdot)$ is bounded as (cf. [24])

$$\mathbb{E}\left[\left\|\hat{Q}_{t+1}-Q^*\right\|_F^2\Big|\hat{Q}_0\right]\leq\varphi_1\left(\beta\right)+\varphi_2(\hat{Q}_0)\exp\left(-2\beta t\right)\tag{4.14}$$

where $\varphi_1\left(\beta\right)$ is a positive and increasing function of $\beta$; while the second term denotes the initialization error, which decays exponentially as the iterations proceed.

Although selection of a constant stepsize prevents the algorithm from exact convergence to $Q^*$ in stationary settings, it enables CCU adaptation to the underlying non-stationary Markov

processes in dynamic scenaria. Furthermore, the optimal policy in practice can be obtained from the Q-function values before convergence is achieved [173, pg. 79].

However, the main practical limitation of the Q-learning algorithm is its slow convergence, which is a consequence of independent updates of the Q-function values. Indeed, Q-function values are related, and leveraging these relationships can lead to multiple updates per observation as well as faster convergence. In the ensuing section, the structure of the problem at hand will be exploited to develop a linear function approximation of the Q-function, which in turn will endow our algorithm not only with fast convergence, but also with scalability.

## 4.5 Scalable caching

Despite simplicity of the updates as well as optimality guarantees of the Q-learning algorithm, its applicability over real networks faces practical challenges. Specifically, the Q-table is of size $|\mathcal{P}_G||\mathcal{P}_L||\mathcal{A}|^2$, where $|\mathcal{A}| = \binom{F}{M}$ encompasses all possible selections of $M$ from $F$ files. Thus, the Q-table size grows prohibitively with $F$, rendering convergence of the table entries, as well as the policy iterates unacceptably slow. Furthermore, action selection in $\min_{\boldsymbol{\alpha}\in\mathcal{A}} Q(\mathbf{s},\mathbf{a})$ entails an expensive exhaustive search over the feasible action set $\mathcal{A}$.

Linear function approximation is a popular scheme for rendering Q-learning applicable to real-world settings [60, 120, 148]. A linear approximation for $Q(\mathbf{s},\mathbf{a})$ in our setup is inspired by the additive form of the instantaneous costs in (4.3). Specifically, we propose to approximate $Q(\mathbf{s},\mathbf{a}')$ as

$$Q(\mathbf{s},\mathbf{a}') \simeq Q_G(\mathbf{s},\mathbf{a}') + Q_L(\mathbf{s},\mathbf{a}') + Q_R(\mathbf{s},\mathbf{a}') \tag{4.15}$$

where $Q_G$, $Q_L$, and $Q_R$ correspond to global and local popularity mismatch, and cache-refreshing costs, respectively.

Recall that the state vector $\mathbf{s}$ consists of three subvectors, namely $\mathbf{s} := [\mathbf{p}_G^\top, \mathbf{p}_L^\top, \mathbf{a}^\top]^\top$. Corresponding to the global popularity subvector, our first term of the approximation in (4.15) is

$$Q_G(\mathbf{s},\mathbf{a}') := \sum_{i=1}^{|\mathcal{P}_G|}\sum_{f=1}^{F} \theta_{i,f}^G \mathbf{1}_{\{\mathbf{p_G}=\mathbf{p_G^i}\}} \mathbf{1}_{\{[\mathbf{a}']_\mathbf{f}=\mathbf{0}\}} \tag{4.16}$$

where the sums are over all possible global popularity profiles as well as files, and the indicator function $\mathbf{1}_{\{\cdot\}}$ takes value 1 if its argument holds, and 0 otherwise; while $\theta_{i,f}^G$ captures the average

"overall" cost if the system is in global state $\mathbf{p}_G^i$, and the CCU decides not to cache the $f$th content. By defining the $|\mathcal{P}_G| \times |\mathcal{F}|$ matrix with $(i,f)$-th entry $\left[\mathbf{\Theta}^G\right]_{i,f} := \theta_{i,f}^G$, one can rewrite (4.16) as

$$Q_G(\mathbf{s}, \mathbf{a}') = \boldsymbol{\delta}_G^\top(\mathbf{p_G})\mathbf{\Theta}^G(\mathbf{1} - \mathbf{a}') \tag{4.17}$$

where

$$\boldsymbol{\delta}_G(\mathbf{p}_G) := \left[\delta(\mathbf{p}_G - \mathbf{p}_G^1), \ldots, \delta(\mathbf{p}_G - \mathbf{p}_G^{|\mathcal{P}_G|})\right]^\top.$$

Similarly, we advocated the second summand in the approximation (4.15) to be

$$\begin{aligned} Q_L(\mathbf{s}, \mathbf{a}') &:= \sum_{i=1}^{|\mathcal{P}_L|}\sum_{f=1}^{F} \theta_{i,f}^L \mathbf{1}_{\{\mathbf{p_L}=\mathbf{p_L^i}\}}\mathbf{1}_{\{[\mathbf{a}']_\mathbf{f}=\mathbf{0}\}} \\ &= \boldsymbol{\delta}_L^\top(\mathbf{p}_L)\mathbf{\Theta}^L(\mathbf{1} - \mathbf{a}') \end{aligned} \tag{4.18}$$

where $\left[\mathbf{\Theta}^L\right]_{i,f} := \theta_{i,f}^L$, and

$$\boldsymbol{\delta}_L(\mathbf{p}_L) := \left[\delta(\mathbf{p}_L - \mathbf{p}_L^1), \ldots, \delta(\mathbf{p}_L - \mathbf{p}_L^{|\mathcal{P}_L|})\right]^\top$$

with $\theta_{i,f}^L$ modeling the average overall cost for not caching file $f$ when the local popularity is in state $\mathbf{p}_L^i$.

Finally, our third summand in (4.15) corresponds to the cache-refreshing cost

$$\begin{aligned} Q_R(\mathbf{s}, \mathbf{a}') :&= \sum_{f=1}^{F} \theta^R \mathbf{1}_{\{[\mathbf{a}']_\mathbf{f}=\mathbf{1}\}}\mathbf{1}_{\{[\mathbf{a}]_\mathbf{f}=\mathbf{0}\}} \\ &= \theta^R \mathbf{a}'^\top (1 - \mathbf{a}) \\ &= \theta^R \left[\mathbf{a}'^\top (1 - \mathbf{a}) + \mathbf{a}^\top \mathbf{1} - \mathbf{a}'^\top \mathbf{1}\right] \\ &= \theta^R \mathbf{a}^\top (\mathbf{1} - \mathbf{a}') \end{aligned} \tag{4.19}$$

where $\theta^R$ models average cache-refreshing cost per content. The constraint $\mathbf{a}^\top\mathbf{1} = \mathbf{a}'^\top\mathbf{1} = M$, is utilized to factor out the term $\mathbf{1} - \mathbf{a}'$, which will become useful later.

Upon defining the set of parameters $\Lambda := \{\mathbf{\Theta}^G, \mathbf{\Theta}^L, \theta^a\}$, the Q-function is readily approximated (cf. (4.15))

$$\widehat{Q}_\Lambda(\mathbf{s}, \mathbf{a}') := \Big(\boldsymbol{\delta}_{\mathbf{G}}^\top(\mathbf{p_G})\mathbf{\Theta^G} + \boldsymbol{\delta}_{\mathbf{L}}^\top(\mathbf{p_L})\mathbf{\Theta^L} + \theta^{\mathbf{R}}\mathbf{a}^\top\Big)(\mathbf{1} - \mathbf{a}'). \tag{4.20}$$

Thus, the original task of learning $|\mathcal{P}_G||\mathcal{P}_L||\mathcal{A}|^2$ parameters in Alg. 1 is now reduced to learning $\Lambda$ containing $(|\mathcal{P}_G| + |\mathcal{P}_L|)\,|\mathcal{F}| + 1$ parameters.

### 4.5.1 Learning $\Lambda$

Given the current parameter estimates $\{\widehat{\mathbf{\Theta}}_{t-1}^G, \widehat{\mathbf{\Theta}}_{t-1}^L, \hat{\theta}_{t-1}^R\}$ at the end of slot $t$, the instantaneous error is given by

$$\begin{aligned}\widehat{\varepsilon}(\mathbf{s}(t-1), \mathbf{a}(t)) := \frac{1}{2}\Big(&C\left(\mathbf{s}(t-1), \mathbf{a}(t)\right) \\ &+ \gamma \min_{\mathbf{a}'} \widehat{Q}_{\Lambda_{t-1}}\left(\mathbf{s}(t), \mathbf{a}'\right) - \widehat{Q}_{\Lambda_{t-1}}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)\Big)^2.\end{aligned} \tag{4.21}$$

Based on this error form, the parameter update rules are obtained using stochastic gradient descent iterations as

$$\begin{aligned}\hat{\mathbf{\Theta}}_t^G &= \hat{\mathbf{\Theta}}_{t-1}^G - \alpha_G \nabla_{\mathbf{\Theta}^G} \widehat{\varepsilon}(\mathbf{s}(t-1), \mathbf{a}(t)) \\ &= \hat{\mathbf{\Theta}}_{t-1}^G + \alpha_G \sqrt{\widehat{\varepsilon}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)}\ \nabla_{\mathbf{\Theta}^G} \widehat{Q}_{\Lambda_{t-1}}(\mathbf{s}(t-1), \mathbf{a}(t)) \\ &= \hat{\mathbf{\Theta}}_{t-1}^G + \alpha_G \sqrt{\widehat{\varepsilon}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)}\ \boldsymbol{\delta}_G(\mathbf{p_G(t-1)})(\mathbf{1} - \mathbf{a(t)})^\top\end{aligned} \tag{4.22}$$

$$\begin{aligned}\hat{\mathbf{\Theta}}_t^L &= \hat{\mathbf{\Theta}}_{t-1}^L - \alpha_L \nabla_{\mathbf{\Theta}^L} \widehat{\varepsilon}(\mathbf{s}(t-1), \mathbf{a}(t)) \\ &= \hat{\mathbf{\Theta}}_{t-1}^L + \alpha_L \sqrt{\widehat{\varepsilon}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)}\ \nabla_{\mathbf{\Theta}^L} \widehat{Q}_{\Lambda_{t-1}}(\mathbf{s}(t-1), \mathbf{a}(t)) \\ &= \hat{\mathbf{\Theta}}_{t-1}^L + \alpha_L \sqrt{\widehat{\varepsilon}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)}\ \boldsymbol{\delta}_L(\mathbf{p_L(t-1)})(\mathbf{1} - \mathbf{a(t)})^\top\end{aligned} \tag{4.23}$$

and

$$\begin{aligned}\hat{\theta}_t^R &= \hat{\theta}_{t-1}^R - \alpha_R \nabla_{\theta^R} \widehat{\varepsilon}(\mathbf{s}[t-1], \mathbf{a}[t]) \\ &= \hat{\theta}_{t-1}^R + \alpha_R \sqrt{\widehat{\varepsilon}\left(\mathbf{s}[t-1], \mathbf{a}[t]\right)}\ \nabla_{\theta^R} \widehat{Q}_{\Lambda_{t-1}}(\mathbf{s}\left[t-1\right], \mathbf{a}\left[t\right]) \\ &= \hat{\theta}_{t-1}^R + \alpha_R \sqrt{\widehat{\varepsilon}\left(\mathbf{s}[t-1], \mathbf{a}[t]\right)}\ \mathbf{a}^\top[t-1](\mathbf{1} - \mathbf{a}[t]).\end{aligned} \tag{4.24}$$

**Algorithm 5** Scalable Q-learning

**Initialize** $\mathbf{s}(0)$ randomly, $\widehat{\mathbf{\Theta}}_0^G = \mathbf{0}$, $\widehat{\mathbf{\Theta}}_0^L = \mathbf{0}$, $\hat{\theta}_0^R = 0$, and thus $\widehat{\boldsymbol{\psi}}(\mathbf{s}) = \mathbf{0}$
For $t = 1, 2, ...$
Take action $\mathbf{a}(t)$ chosen probabilistically by

$$\mathbf{a}(t) = \begin{cases} M \text{ best files via } \widehat{\boldsymbol{\psi}}\left(\mathbf{s}(t-1)\right) & \text{w.p. } 1-\epsilon_t \\ \text{random } \mathbf{a} \in \mathcal{A} & \text{w.p. } \epsilon_t \end{cases}$$

$\mathbf{p}_G(t)$ and $\mathbf{p}_L(t)$ are revealed based on user requests
Set $\mathbf{s}(t) := \left[\mathbf{p}_G^\top(t), \mathbf{p}_L^\top(t), \mathbf{a}(t)^\top\right]^\top$
Incur cost $C\Big(\mathbf{s}(t-1), \mathbf{a}(t)\Big|\mathbf{p_G}(t), \mathbf{p_L}(t)\Big)$
Find $\widehat{\varepsilon}\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)$ Update $\widehat{\mathbf{\Theta}}_t^G$, $\widehat{\mathbf{\Theta}}_t^L$ and $\hat{\theta}_t^R$ based on (4.22)-(4.24)

The pseudocode for this scalable approximation of the Q-learning scheme is tabulated in Alg. 2.

The upshot of this scalable scheme is three-fold.

- The large state-action space in the Q-learning algorithm is handled by reducing the number of parameters from $|\mathcal{P}_G||\mathcal{P}_L||\mathcal{A}|^2$ to $\left(|\mathcal{P}_G| + |\mathcal{P}_L|\right)|\mathcal{F}| + 1$.
- In contrast to single-entry updates in the exact Q-learning Alg. 1, $F - M$ entries in $\widehat{\mathbf{\Theta}}^G$ and $\widehat{\mathbf{\Theta}}^L$ as well as $\theta^R$, are updated per observation using (4.22)-(4.24), which leads to a much faster convergence.
- The exhaustive search in $\min_{\mathbf{a}\in\mathcal{A}} Q\left(\mathbf{s}, \mathbf{a}\right)$ required in exploitation; and also in the error evaluation (4.21), is circumvented. Specifically, it holds that (cf. (4.20))

$$\min_{\mathbf{a}'\in\mathcal{A}} Q(\mathbf{s}, \mathbf{a}') \approx \min_{\mathbf{a}'\in\mathcal{A}} \boldsymbol{\psi}^\top(\mathbf{s})\left(\mathbf{1} - \mathbf{a}'\right) = \max_{\mathbf{a}'\in\mathcal{A}} \boldsymbol{\psi}^\top(\mathbf{s})\ \mathbf{a}' \tag{4.25}$$

  where

$$\boldsymbol{\psi}(\mathbf{s}) := \boldsymbol{\delta}_G^\top(\mathbf{p_G})\mathbf{\Theta}^G + \boldsymbol{\delta}_L^\top(\mathbf{p_L})\mathbf{\Theta}^L + \theta^R\mathbf{a}^\top.$$

  The solution of (4.25) is readily given by $[\mathbf{a}]_{\nu_i} = 1$ for $i = 1, \ldots, M$, and $[\mathbf{a}]_{\nu_i} = 0$ for $i > M$, where $[\boldsymbol{\psi}(\mathbf{s})]_{\nu_F} \leq \cdots \leq [\boldsymbol{\psi}(\mathbf{s})]_{\nu_1}$ are sorted entries of $\boldsymbol{\psi}(\mathbf{s})$.

**Remark 2.** In the model of Sec. II-B, the state-space cardinality of the popularity vectors is

finite. These vectors can be viewed as centroids of quantization regions partitioning a state space of infinite cardinality. Clearly, such a partitioning inherently bears a complexity-accuracy trade off, motivating optimal designs to achieve a desirable accuracy for a given affordable complexity. This is one of our future research directions for the problem at hand.

Simulation based evaluation of the proposed algorithms for RL-based caching is now in order.

## 4.6 Numerical tests

In this section, performance of the proposed Q-learning algorithm and its scalable approximation is tested. To compare the proposed algorithms with the optimal offline caching policy, we first simulated a small network with $F = 10$ contents, and caching capacity $M = 2$ at the local SB. Global popularity profile is modeled by a two-state Markov chain with states $\mathbf{p}_G^1$ and $\mathbf{p}_G^2$,that are drawn from Zipf distributions having parameters $\eta_1^G = 1$ and $\eta_2^G = 1.5$, respectively [27]; see also Fig. 4.4. That is, for state $i \in \{1, 2\}$, the $F$ contents are assigned a random ordering of popularities, and then sorted accordingly in a descending order. Given this ordering and the Zipf distribution parameter $\eta_i^G$, the popularity of the $f$-th content is set to

$$\left[\mathbf{p}_{\mathrm{G}}^i\right]_f = \frac{1}{f^{\eta_i} \sum_{l=1}^{F} 1 \,/\, l^{\eta_i^G}} \quad \text{for } \; i = 1, 2$$

where the summation normalizes the components to follow a valid probability mass function, while $\eta_i^G \geq 0$ controls the skewness of popularities. Specifically, $\eta_i^G = 0$ yields a uniform spread of popularity among contents, while a large value of $\eta_i$ generates more skewed popularities. Furthermore, state transition probabilities of the Markov chain modeling global popularity profiles are

$$\boldsymbol{P}^G := \begin{bmatrix} p_{11}^G & p_{12}^G \\ p_{21}^G & p_{22}^G \end{bmatrix} = \begin{bmatrix} 0.8 & 0.2 \\ 0.75 & 0.25 \end{bmatrix}.$$

Similarly, local popularities are modeled by a two-state Markov chain, with states $\mathbf{p}_L^1$ and $\mathbf{p}_L^2$, whose entries are drawn from Zipf distributions with parameters $\eta_1^L = 0.7$ and $\eta_2^L = 2.5$,

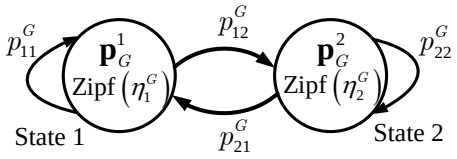



Figure 4.2: Global popularity Markov chain.

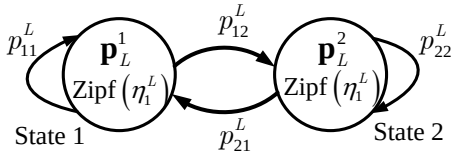



Figure 4.3: Local popularity Markov chain.

Figure 4.4: Popularity profiles Markov chains.

respectively. The transition probabilites of the local popularity Markov chain are

$$\boldsymbol{P}^L := \begin{bmatrix} p_{11}^L & p_{12}^L \\ p_{21}^L & p_{22}^L \end{bmatrix} = \begin{bmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{bmatrix}.$$

Caching performance is assessed under two cost-parameter settings: (s1) $\lambda_1 = 10, \lambda_2 = 600, \lambda_3 = 1000$; and, (s2) $\lambda_1 = 600, \lambda_2 = 10, \lambda_3 = 1000$. For both (s1) and (s2), the optimal offline caching policy is found by utilizing the policy iteration with known transition probabilities. In addition, Q-learning in Alg. 1 and its scalable approximation in Alg. 2 are run with $\beta_t = 0.8$, $\alpha_G = \alpha_L = \alpha_R = 0.005$, and $\epsilon_t = 1$ / iteration index, thus promoting exploration in the early iterations, and exploitation in later iterations.

Fig. 4.5 depicts the observed cost versus iteration (time) index averaged over 100 realizations. It is seen that the cashing cost via Q-learning, and through its scalable approximation converge to that of the optimal offline policy. As anticipated, even for the small size of this network, namely $|\mathcal{P}_G| = |\mathcal{P}_L| = 2$ and $|\mathcal{A}| = 45$, the Q-learning algorithm converges slowly to the optimal policy, especially under s2, while its scalable approximation exhibits faster convergence.

In order to highlight the trade-off between global and local popularity mismatches, the percentage of accommodated requests via cache is depicted in Fig. 4.6 for settings (s3) $\lambda_1 = \lambda_3 = 0, \lambda_2 = 1,000$, and (s4) $\lambda_1 = \lambda_2 = 0, \lambda_3 = 1,000$. Observe that penalizing local popularity-mismatch in (s3) forces the caching policy to adapt to local request dynamics, thus

accommodating a higher percentage of requests via cache, while (s4) prioritizes tracking global popularities, leading to a lower cache-hit in this setting. Due to slow convergence of the exact Q-learning under (s3) and (s4), only the performance of the scalable solver is presented here.

Furthermore, the convergence rate of Algs. 1 and 2 is illustrated in Fig. 4.7, where average normalized error is evaluated in terms of the "exploitation index." Specifically, a pure exploration is taken for the first $T_{\text{explore}}$ iterations of the algorithms, i.e., $\epsilon_t = 1$ for $t = 1, 2, \ldots, T_{\text{explore}}$; and a pure exploitation with $\epsilon_t = 0$ is adopted afterwards. We have set $\alpha = 0.005$, and selected $\beta_t = \beta \in (0, 1)$ so that the fastest convergence is achieved. As the plot demonstrates, the exact Q-learning Alg. 1 exhibits slower convergence, whereas just a few iterations suffice for the scalable Alg. 2 to converge to the optimal solution, thanks to the reduced dimension of the problem as well as the multiple updates that can be afforded per iteration.

Having established the accuracy and efficiency of the Alg. 2, we next simulated a larger network with $F = 1,000$ available files, and a cache capacity of $M = 10$, giving rise to a total of $\binom{1000}{10} \simeq 2 \times 10^{23}$ feasible caching actions. In addition, we set the local and global popularity Markov chains to have $|\mathcal{P}_L| = 40$ and $|\mathcal{P}_G| = 50$ states, for which the underlying state transition probabilities are drawn randomly, and Zipf parameters are drawn uniformly over the interval $(2, 4)$.

Fig. 4.8 plots the performance of Alg. 2 under (s5) $\lambda_1 = 100$, $\lambda_2 = 20$, $\lambda_3 = 20$, (s6) $\lambda_1 = 0$, $\lambda_2 = 0$, $\lambda_3 = 1,000$, and (s7) $\lambda_1 = 0$, $\lambda_2 = 1,000$, $\lambda_3 = 600$. Exploration-exploitation parameter is set to $\epsilon_t = 1$ for $t = 1, 2, \ldots, 5,000$, in order to greedily explore the entire state-action space in initial iterations, and $\epsilon_t = 1$ / iteration index for $t > 5,000$. Finding the optimal offline policy in (s6) and (s7) requires prohibitively sizable memory as well as extremely high computational complexity, and it is thus unaffordable for this network. However, having large cache-refreshing cost with $\lambda_1 \gg \lambda_2, \lambda_3$ in (s5) forces the optimal caching policy to freeze its cache contents, making the optimal caching policy predictable in this setting. Despite the very limited storage capacity, of $10 \ / \ 1,000 = 0.01$ of available files, utilization of RL-enabled caching offers a considerable reduction in incurred costs, while the proposed approximated Q-learning endows the approach with scalability and light-weight updates.

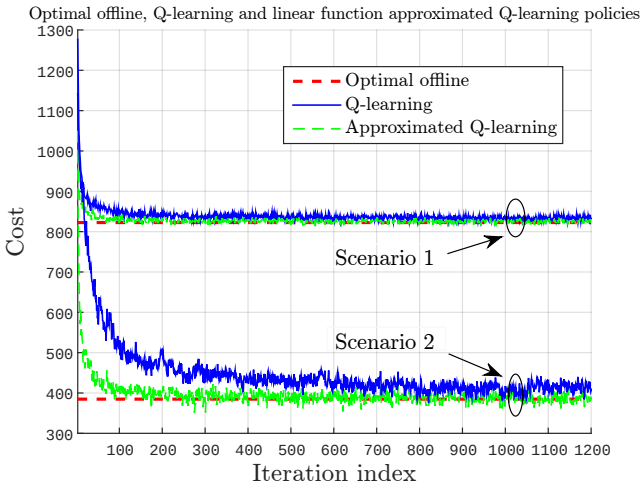



Figure 4.5: Performance of the proposed algorithms.

## 4.7 Conclusions

The present Chapter addressed caching in 5G cellular networks, where space-time popularity of requested files is modeled via local and global Markov chains. By considering local and global popularity mismatches as well as cache-refreshing costs, 5G caching is cast as a reinforcement-learning task. A Q-learning algorithm is developed for finding the optimal caching policy in an online fashion, and its linear approximation is provided to offer scalability over large networks. The novel RL-based caching offers an asynchronous and semi-distributed caching scheme, where adaptive tuning of parameters can readily bring about policy adjustments to space-time variability of file requests via light-weight updates.

## 4.8 Deep Reinforcement Learning for Adaptive Caching in Hierarchical Content Delivery Networks

## 4.9 Introduction

Deep neural networks (DNNs) have lately boosted the notion of “learning from data” with field-changing performance improvements reported in diverse artificial intelligence tasks [63]. DNNs can cope with the ‘curse of dimensionality’ by providing compact low-dimensional

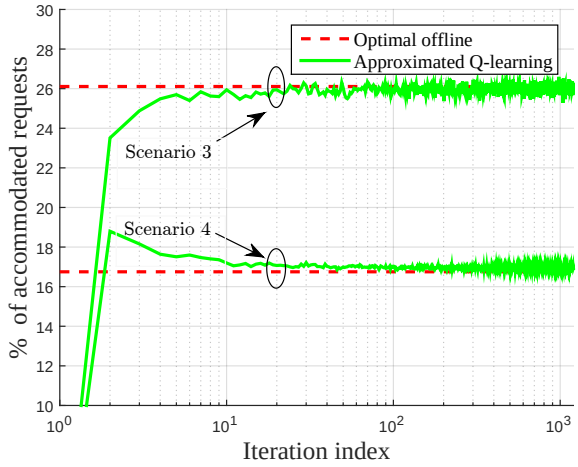



Figure 4.6: Percentage of accommodated requests via cache.

representations of high-dimensional data [14]. Combining deep learning with RL, deep (D) RL has created the first artificial agents to achieve human-level performance across many challenging domains [128, 114]. As another example, a DNN system was built to operate Google's data centers, and shown able to consistently achieve a 40% reduction in energy consumption for cooling [58]. This system provides a general-purpose framework to understand complex dynamics, which has also been applied to address other challenges including e.g., dynamic spectrum access [132], multiple access and handover control [208], [186], as well as resource allocation in fog-radio access networks [172, 46] or software-defined networks [207, 69].

In realistic networks, popularities exhibit dynamics, which motivate well the so-termed *dynamic* caching. A Poisson shot noise model was adopted to approximate the evolution of popularities in [177], for which an age-based caching solution was developed in [97]. RL based methods have been pursued in [151, 170, 154, 72]. Specifically, a Q-learning based caching scheme was developed in [151] to model global and local content popularities as Markovian processes. Considering Poisson shot noise popularity dynamics, a policy gradient RL based caching scheme was devised in [170]. Assuming stationary file popularities and service costs, a dual-decomposition based Q-learning approach was pursued in [154]. Albeit reasonable for discrete states, these approaches cannot deal with large continuous state-action spaces. To cope with such spaces, DRL approaches have been considered for content caching in e.g.,

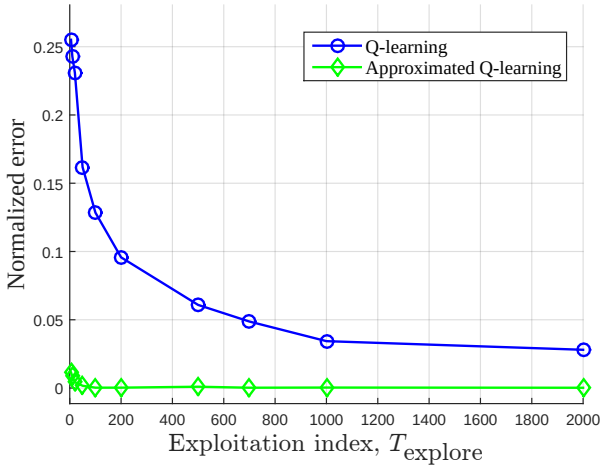



Figure 4.7: Convergence rate of the exact and scalable Q-learning.

[72, 216, 73, 71, 114, 219]. Encompassing finite-state time-varying Markov channels, a deep Q-network approach was devised in [72]. An actor-critic method with deep deterministic policy gradient updates was used in [216]. Boosted network performance using DRL was documented in several other applications, such as connected vehicular networks [73], and smart cities [71].

The aforementioned works focus on devising caching policies for a *single* caching entity. A more common setting in next-generation networks however, involves a network of interconnected caching nodes. It has been shown that considering a network of connected caches jointly can further improve performance [116, 25]. For instance, leveraging network topology and the broadcast nature of links, the coded caching strategy in [116] further reduces data traffic over a network. This idea has been extended in [140] to an online setting, where popularities are modeled Markov processes. Collaborative and distributed online learning approaches have been pursued [41, 25, 184]. Indeed, today's content delivery networks such as Akamai [133], have tree network structures. Accounting for the hierarchy of caches has become a common practice in recent works; see also [43, 166, 176]. Joint routing and in-network content caching in a hierarchical cache network was formulated in [43], for which greedy schemes with provable performance guarantees can be found in [166].

We identify the following challenges that need to be addressed when designing practical caching methods for network of caches.

**c1)** *Networked caching.* Caching decisions of a node, in a network of caches, influences

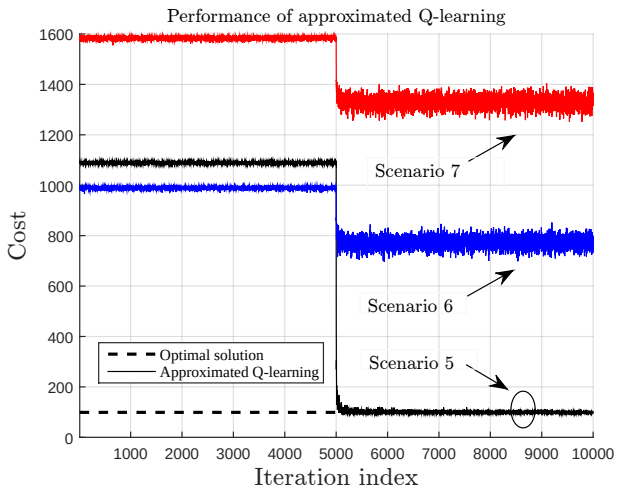



Figure 4.8: Performance in large state-action space scenaria.

decisions of all other nodes. Thus, a desired caching policy must adapt to the network topology and policies of neighboring nodes.

**c2)** *Complex dynamics.* Content popularities are random and exhibit unknown space-time, heterogeneous, and often non-stationary dynamics over the entire network.

**c3)** *Large continuous state space.* Due to the shear size of available content, caching nodes, and possible realizations of content requests, the decision space is huge.

### 4.9.1 This section

Prompted by the recent interest in hierarchical caching, here we focus on a two-level network caching, where a parent node is connected to multiple leaf nodes to serve end-user file requests. Such a two-level network constitutes the building block of the popular tree hierarchical cache networks in e.g., [133]. To model the interaction between caching decisions of parent and leaf nodes along with the space-time evolution of file requests, a scalable DRL approach based on hyper deep Q-networks (DQNs) is developed. As corroborated by extensive numerical tests, the novel caching policy for the parent node can adapt itself to local policies of leaf nodes and space-time evolution of file requests. Moreover, our approach is simple-to-implement, and performs close to the optimal policy.

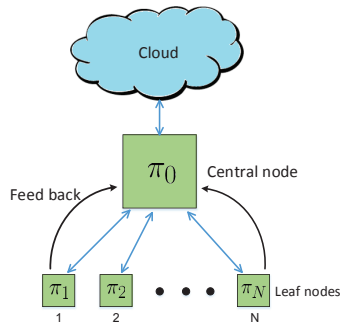



Figure 4.9: A network of caching nodes.

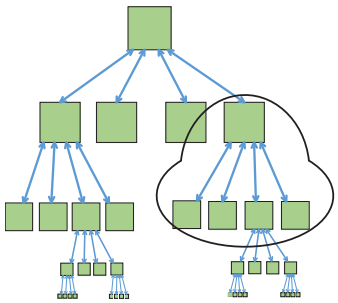


Figure 4.10: A hierarchical tree network cache system.

## 4.10 Modeling and Problem Statement

Consider a two-level network of interconnected caching nodes, where a parent node is connected to $N$ leaf nodes, indexed by $n \in \mathcal{N} := \{1, \ldots, N\}$. The parent node is connected to the cloud through a (typically congested) back-haul link; see Fig. 4.9. One could consider this network as a part of a large hierarchical caching system, where the parent node is connected to a higher level caching node instead of the cloud; see Fig. 4.10. In a content delivery network for instance, edge servers (a.k.a. points of presence or PoPs) are the leaf nodes, and a fog server acts as the parent node. Likewise, (small) base stations in a 5G cellular network are the leaf nodes, while a serving gate way (S-GW) may be considered as the parent node; see also [40, p. 110].

All nodes in this network store files to serve file requests. Every leaf node serves its locally connected end users, by providing their requested files. If a requested content is locally available at a leaf node, the content will be served immediately at no cost. If it is not locally available due to limited caching capacity, the content will be fetched from its parent node, at a certain

cost. Similarly, if the file is available at the parent node, it will be served to the leaf at no cost; otherwise, the file must be fetched from the cloud at a higher cost.

To mitigate the burden with local requests on the network, each leaf node stores 'anticipated' locally popular files. In addition, this paper considers that each parent node stores files to serve requests that are *not* locally served by leaf nodes. Since leaf nodes are closer to end users, they frequently receive file requests that exhibit rapid temporal evolution at a *fast* timescale. The parent node on the other hand, observes aggregate requests over a large number of users served by the $N$ leaf nodes, which naturally exhibit smaller fluctuations and thus evolve at a *slow* timescale.

This motivated us to pursue a two-timescale approach to managing such a network of caching nodes. To that end, let $\tau = 1, 2, \ldots$ denote the slow time intervals, each of which is further divided into $T$ fast time slots indexed by $t = 1, \ldots, T$; see Fig. 4.11 for an illustration. Each fast time slot may be e.g., 1-2 minutes depending on the dynamics of local requests, while each slow time interval is a period of say 4-5 minutes. We assume that the network state remains unchanged during each fast time slot $t$, but can change from $t$ to $t + 1$.

Consider a total of $F$ files in the cloud, which are collected in the set $\mathcal{F} = \{1, \ldots, F\}$. At the beginning of each slot $t$, every leaf node $n$ selects a subset of files in $\mathcal{F}$ to prefetch and store for possible use in this slot. To determine which files to store, every leaf node relies on a local caching policy function denoted by $\pi_n$, to take (cache or no-cache) action $\boldsymbol{a}_n(t+1, \tau) = \pi_n(\boldsymbol{s}_n(t, \tau))$ at the beginning of slot $t + 1$, based on its *state* vector $\boldsymbol{s}_n$ at the end of slot $t$. We assume this action takes a negligible amount of time relative to the slot duration; and define the state vector $\boldsymbol{s}_n(t, \tau) := \boldsymbol{r}_n(t, \tau) := [r_n^1(t, \tau) \cdots r_n^F(t, \tau)]^\top$ to collect the number of requests received at leaf node $n$ for individual files over the duration of slot $t$ on interval $\tau$. Likewise, to serve file requests that have not been served by leaf nodes, the parent node takes action $\boldsymbol{a}_0(\tau)$ to store files at the beginning of every interval $\tau$, according to a certain policy $\pi_0$. Again, as aggregation smooths out request fluctuations, the parent node observes slowly varying file requests, and can thus make caching decisions at a relatively slow timescale. In the next section, we present a two-timescale approach to managing such a network of caching nodes.

## 4.11 Two-timescale Problem Formulation

File transmission over any network link consumes resources, including e.g., energy, time, and bandwidth. Hence, serving any requested file that is not locally stored at a node, incurs a cost. Among possible choices, the present paper considers the following cost for node $n \in \mathcal{N}$, at slot $t+1$ of interval $\tau$

$$\boldsymbol{c}_n(\pi_n(\boldsymbol{s}_n(t,\tau)), \boldsymbol{r}_n(t+1,\tau), \boldsymbol{a}_0(\tau)) := \boldsymbol{r}_n(t+1,\tau) \odot (\mathbf{1} - \boldsymbol{a}_0(\tau)) \odot (\mathbf{1} - \boldsymbol{a}_n(t+1,\tau)) + \boldsymbol{r}_n(t+1,\tau) \odot (\mathbf{1} - \boldsymbol{a}_n(t+1,\tau)) \quad (4.26)$$

where $\boldsymbol{c}_n(\cdot) := [c_n^1(\cdot) \ \cdots \ c_n^F(\cdot)]^\top$ concatenates the cost for serving individual files per node $n$; symbol $\odot$ denotes entry-wise vector multiplication; entries of $\boldsymbol{a}_0$ and $\boldsymbol{a}_n$ are either 1 (cache, hence no need to fetch), or, 0 (no-cache, hence fetch); and $\mathbf{1}$ stands for the all-one vector. Specifically, the second summand in (4.26) captures the cost of the leaf node fetching files for end users, while the first summand corresponds to that of the parent fetching files from the cloud.

We model user file requests as Markov processes with unknown transition probabilities [151]. Per interval $\tau$, a reasonable caching scheme for leaf node $n \in \mathcal{N}$ could entail minimizing the expected cumulative cost; that is,

$$\pi_{n,\tau}^* := \arg\min_{\pi_n \in \Pi_n} \mathbb{E}\Big[\sum_{t=1}^{T} \mathbf{1}^\top \boldsymbol{c}_n(\pi_n(\boldsymbol{s}_n(t,\tau)), \boldsymbol{r}_n(t+1,\tau), \boldsymbol{a}_0(\tau))\Big] \quad (4.27)$$

where $\Pi_n$ represents the set of all feasible policies for node $n$. Although solving (4.27) is in general challenging, efficient near-optimal solutions have been introduced in several recent contributions; see e.g., [151, 170, 21], and references therein. In particular, a RL based approach using tabular $Q$-learning was pursued in our precursor [151], which can be employed here to tackle this fast timescale optimization. The remainder of this paper will thus be on designing the caching policy $\pi_0$ for the parent node, that can learn, track, and adapt to the leaf node policies as well as user file requests.

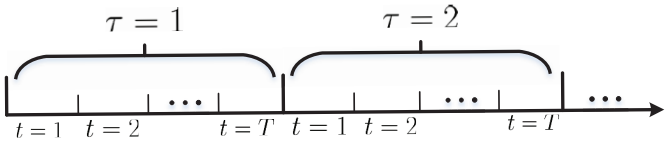



Figure 4.11: Slow and fast time slots.

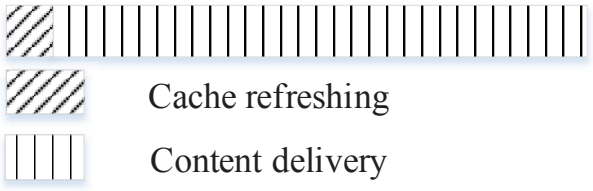



Figure 4.12: Structure of slots and intervals.

## 4.12 Reinforcement Learning for Adaptive Caching with Dynamic Storage Pricing

## 4.13 Introduction

To target different objectives such as content-access latency, energy, storage or bandwidth utilization, corresponding deterministic cost parameters are defined, and the aggregated cost is minimized in [95, 142]. Deterministic cost parameters, however, may be inaccurate in modeling practical settings, as spatio-temporal popularity evolutions, network resources such as bandwidth and cache capacity are *random* and subject to change over time and space, due to e.g., time-varying data traffic over links, previous cache decisions, or channel fluctuations. Therefore, this necessitates modeling the caching problem from a *stochastic optimization* perspective, while accounting for the inherently random nature of available resources and file requests.

*Contributions:* This Section aspires to fill this gap by relying on dual decomposition techniques which transform the limits on the available resources in the original (primal) optimization into stochastic prices in the dual problem. Building on this approach, the goal is to design more flexible caching schemes by introducing a generic dynamic pricing formulation, while enabling SBs to learn the optimal fetching-caching decisions using low-complexity techniques. Our contributions are listed as follows.

1) A general formulation of the caching problem by introducing time-varying and stochastic costs is presented, in which the fetching and caching decisions are found through a constrained optimization with the objective of reducing the overall cost, aggregated across files and time instants (Section 4.14).

2) Since the caching decision in a given time slot not only affects the instantaneous cost, but also influences the cache availability in the future, the problem is indeed a dynamic programming (DP), and therefore can be effectively solved by reinforcement learning-based approaches. By assuming known and stationary distributions for the costs and popularities, and upon relaxing the limited cache capacity constraint, the proposed generic optimization problem is shown to become separable across files, and thus can be efficiently solved using the value-iteration algorithm (Section 4.15).

3) Subsequently, it is shown that the particular case where the cache capacity is limited and the distribution of the pertinent parameters are unknown can be handled by the proposed generic formulation. Thus, in order to address these issues, a dual-decomposition technique is developed to cope with the coupling constraint associated with the storage limitation. Finally, an online low complexity ($Q$-function based) reinforcement learning solver is put forth for learning the optimal fetch-cache decisions on-the-fly (Section 4.16).

4) The separability of the objective across files together with the use of marginalized value functions [108] enable the decomposition of the original problem into smaller-dimension sub-problems. This in turn leads to circumventing the so-called curse of dimensionality, which commonly arises in reinforcement learning problems (Sections 4.15 and 4.16).

The effectiveness of the proposed scheme in terms of efficiency as well as scalability is corroborated by various numerical tests. Although our proposed approach enjoys theoretical guarantees in learning the optimal fetch-cache decisions in stationary settings, numerical tests also corroborate its merits in non-stationary scenarios.

## 4.14 Operating conditions and costs

Consider a memory-enabled SB responsible for serving file (content) requests denoted by $f = 1, 2, \ldots, F$ across time. The requested contents are transmitted to users either by fetching

through a (costly) back-haul transmission link connecting the SB to the cloud, or, by utilizing the local storage unit in the SB where popular contents have been proactively cached ahead of time. The system is considered to operate in a slotted fashion with $t = 1, 2, \ldots$ denoting time.

During slot $t$ and given the available cache contents, the SB receives a number of file requests whose provision incurs certain costs. Specifically, for a requested file $f$, fetching it from the cloud through the back-haul link gives rise to scheduling, routing and transmission costs, whereas its availability at the cache storage in the SB will eliminate such expenses. However, local caching also incurs a number of (instantaneous) costs corresponding to memory or energy consumption. This gives rise to an inherent caching-versus-fetching trade-off, where one is promoted over the other depending on their relative costs. The objective here is to propose a simple yet sufficiently general framework to minimize the sum-average cost over time by optimizing fetch-cache decisions while adhering to the constraints inherent to the operation of the system at hand, and user-specific requirements. The variables, constraints, and costs involved in this optimization are described in the ensuing subsections.

### 4.14.1 Variables and constraints

Consider the system at time slot $t$, where the binary variable $r_t^f$ represents the incoming request for file $f$; that is, $r_t^f = 1$ if the file $f$ is requested during slot $t$, and $r_t^f = 0$, otherwise. Here, we assume that $r_t^f = 1$ necessitates serving the file to the user and dropping requests is not allowed; thus, requests must be carried out either by fetching the file from the cloud or by utilizing the content currently available in the cache. Furthermore, at the end of each slot, the SB will decide if content $f$ should be stored in the cache for its possible reuse in a subsequent slot.

To formalize this, let us define the "fetching" *decision* variable $w_t^f \in \{0, 1\}$ along the "caching" *decision* variable $a_t^f \in \{0, 1\}$. Setting $w_t^f = 1$ implies "fetching" file $f$ at time $t$, while $w_t^f = 0$ means "no-fetching." Similarly, $a_t^f = 1$ implies that content $f$ will be stored in cache at the end of slot $t$ for the next slot, while $a_t^f = 0$ implies that it will not. Furthermore, let the storage *state* variable $s_t^f \in \{0, 1\}$ account for the availability of files at the local cache. In particular, $s_t^f = 1$ if file $f$ is available in the cache at the beginning of slot $t$, and $s_t^f = 0$ otherwise. Since the availability of file $f$ directly depends on the caching decision at time $t - 1$, we have

$$\text{C1:} \quad s_t^f = a_{t-1}^f, \quad \forall f, t, \tag{4.28}$$

which will be incorporated into our optimization as constraints.

Moreover, since having $r_t^f = 1$ implies transmission of file $f$ to the user(s), it requires either having the file in cache ($s_t^f = 1$) or fetching it from the cloud ($w_t^f = 1$), giving rise to the second set of constraints

$$\text{C2:} \quad r_t^f \leq w_t^f + s_t^f, \quad \forall f, t. \tag{4.29}$$

Finally, the caching decision $a_t^f$ can be set to 1 only when the content $f$ is available at time $t$; that is, only if either fetching is carried out ($w_t^f = 1$) or the current cache state is $s_t^f = 1$. This in turn implies the third set of constraints as

$$\text{C3:} \quad a_t^f \leq s_t^f + w_t^f, \quad \forall f, t. \tag{4.30}$$

### 4.14.2 Prices and aggregated costs

To account for the caching and fetching costs, let $\rho_t^f$ and $\lambda_t^f$ denote the (generic) costs associated with $a_t^f = 1$ and $w_t^f = 1$, respectively. Focusing for now on the caching cost and with $\sigma_f$ denoting the size of content $f$, a simple form for $\rho_t^f$ is

$$\rho_t^f = \sigma_f(\rho'_t + \rho'^f_t) + (\rho''_t + \rho''^f_t), \tag{4.31}$$

where the first term is proportional to the file size $\sigma_f$, while the second one is constant. Note also that we consider file-dependent costs (via variables $\rho'^f_t$ and $\rho''^f_t$), as well as cost contributions which are common across files (via $\rho'_t$ and $\rho''_t$). In most practical setups, the latter will dominate over the former. For example, the caching cost per bit is likely to be the same regardless of the particular type of content, so that $\rho'^f_t = \rho''^f_t = 0$. From a modeling perspective, variables $\rho_t^f$ can correspond to actual prices paid to an external entity (e.g., if associated with energy consumption costs), marginal utility or cost functions, congestion indicators, Lagrange multipliers associated with constraints, or linear combinations of those (see, e.g., [108, 59, 38, 181] and Section 4.16). Accordingly, the corresponding form for the fetching cost is

$$\lambda_t^f = \sigma_f(\lambda'_t + \lambda'^f_t) + (\lambda''_t + \lambda''^f_t). \tag{4.32}$$

As before, if the transmission link from the cloud to the SB is the same for all contents, the prices $\lambda'_t$ and $\lambda''_t$ are expected to dominate their file-dependent counterparts $\lambda'^f_t$ and $\lambda''^f_t$.

Upon defining the corresponding cost for a given file as $c_t^f(a_t^f, w_t^f; \rho_t^f, \lambda_t^f) = \rho_t^f a_t^f + \lambda_t^f w_t^f$, the aggregate cost at time $t$ is given by

$$c_t := \sum_{f=1}^{F} c_t^f(a_t^f, w_t^f; \rho_t^f, \lambda_t^f) = \sum_{f=1}^{F} \rho_t^f a_t^f + \lambda_t^f w_t^f, \tag{4.33}$$

which is the basis for the DP formulated in the next section. For future reference, Fig. 4.13 shows a schematic of the system model and the notation introduced in this section.

## 4.15 Optimal caching with time-varying costs

Since decisions are coupled across time [cf. constraint (4.28)], and the future values of prices as well as state variables are inherently random, our goal is to sequentially make fetch-cache decisions to minimize the long-term average discounted aggregate cost

$$\bar{\mathcal{C}} := \mathbb{E}\left[\sum_{t=0}^{\infty}\sum_{f=1}^{F} \gamma^t c_t^f\left(a_t^f, w_t^f; \rho_t^f, \lambda_t^f\right)\right] \tag{4.34}$$

where the expectation is taken with respect to (w.r.t.) the random variables $\boldsymbol{\theta}_t^f := \{r_t^f, \lambda_t^f, \rho_t^f\}$, and $0 < \gamma < 1$ is the discounting factor whose tuning trades off current versus more uncertain future costs. To address the optimization, the following assumptions are considered:

AS1) The values of $\boldsymbol{\theta}_t^f$ are drawn from a stationary distribution.

AS2) The distribution of $\boldsymbol{\theta}_t^f$ is known.

AS3) The drawn value of $\boldsymbol{\theta}_t^f$ is revealed at the beginning of each slot $t$, before fetch-cache decisions are made.

AS1 and AS2 allow finding the expectations in this section, and will be relaxed in Section III-E to further generalize our approach to settings where the distributions are unknown. In practice, one may estimate these distributions through e.g., historical data.

The ultimate goal here is to take *real-time* fetch-cache decisions by minimizing the expected *current plus future cost* while adhering to operational constraints, giving rise to the following

optimization

$$\text{(P1)} \quad \min_{\{(w_k^f, a_k^f)\}_{f,k\geq t}} \bar{\mathcal{C}}_t := \sum_{k=t}^{\infty} \sum_{f=1}^{F} \gamma^{k-t} \mathbb{E}\left[c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right)\right]$$
$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, a_{k-1}^f), \quad \forall f,\ k \geq t$$

where

$$\mathcal{X}(r_k^f, a_{k-1}^f) := \Big\{(w,a) \;\Big|\; w \in \{0,1\},\, a \in \{0,1\},$$
$$s_k^f = a_{k-1}^f,\ r_k^f \leq w + s_k^f,\ \ a \leq s_k^f + w\Big\},$$

and the expectation is taken w.r.t. $\{\boldsymbol{\theta}_k^f\}_{\forall k \geq t+1}$.

The presence of the set $\mathcal{X}(r_k^f, a_{k-1}^f)$ in the constraints demonstrates that the cache state at a given time depends on previous cache decisions, thus coupling the optimization variables across time. It also implies that any instantaneous decision will influence the optimization problem in subsequent slots, having a long-standing influence on future costs. The coupling of the optimization variables across time indeed necessitates utilization of DP tools, motivating the implementation to reinforcement learning algorithms to design efficient solvers.

To find the solution of the DP in (P1) we implement the following steps: a) identifying the current and expected future aggregate costs (the latter gives rise to the so-called value functions); b) expressing the corresponding Bellman equations over the value functions; and c) proposing a method to estimate the value functions accordingly. This is the subject of the ensuing subsections, which start by further exploiting the structure of our problem to reduce the complexity of the proposed solution.

### 4.15.1 Bellman equations for the per-content problem

Focusing on (P1), one can readily deduce that: (i) consideration of the content-dependent prices renders the objective in (P1) separable across $f$, and (ii) the constraints in (P1) are also separable across $f$. Furthermore, the decisions $a_t^f$ and $w_t^f$ for a given $f$, do not affect the values (distribution) of $\boldsymbol{\theta}_{k'}^{f'}$ for files $f' \neq f$ and for times $t' > t$. Thus, (P1) naturally gives rise to the

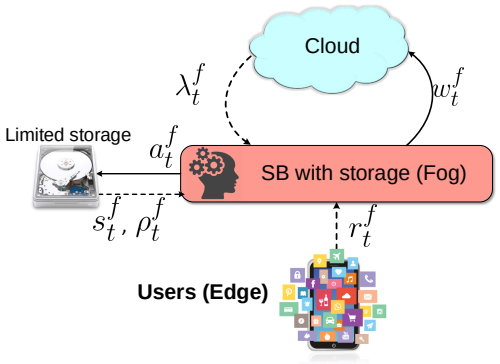



Figure 4.13: System model and main notation. The state variables (dashed lines) are the storage indicator $s_t^f$ and the content request $r_t^f$, as well as the dynamic caching and fetching prices $\rho_t^f$ and $\lambda_t^f$. The optimization variables (solid lines) are the caching and fetching decisions $a_t^f$ and $w_t^f$. The instantaneous per-file cost is $c_t^f = \rho_t^f a_t^f + \lambda_t^f w_t^f$. Per slot $t$, the SB collects the state variables $\{s_t^f, r_t^f; \rho_t^f, \lambda_t^f\}_{f=1}^F$, and decides the values of $\{a_t^f, w_t^f\}_{f=1}^F$ considering not only the cost at time $t$ but also the cost at time instants $t' > t$.

per-file optimization

$$\text{(P2)} \quad \min_{\{(w_k^f, a_k^f)\}_{k\geq t}} \bar{\mathcal{C}}_t^f := \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E}\left[c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right)\right]$$
$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, a_{k-1}^f), \quad k \geq t$$

which must be solved for $f = 1, ..., F$. Indeed, the aggregate cost associated with (P2) will not depend on variables corresponding to files $f' \neq f$ [108]. This is the case if, for instance, the involved variables are independent of each other (which is the setup considered here), or when the focus is on a large system where the contribution of an individual variable to the aggregate network behavior is practically negligible.

*Bellman equations and value function:* The DP in (P2) can be solved with the corresponding Bellman equations, which require finding the associated value functions. To this end, consider the system at time $t$, where the cache state as well as the file requests and cost parameters are all given, so that we can write $s_t^f = s_0^f$ and $\boldsymbol{\theta}_t^f = \boldsymbol{\theta}_0^f$. Then, the optimal fetch-cache decision $(w_t^{f*}, a_t^{f*})$ is readily expressible as the solution to (4.35). The objective in (4.35) is rewritten in (4.36) as the summation of current and discounted average future costs. The form of (4.36) is testament to the fact that problem (P2) is a DP and the caching decision $a$

$$\left(w_t^{f*}, a_t^{f*}\right) := \underset{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)}{\arg\min} \left\{ \mathbb{E}_{\boldsymbol{\theta}_k^f} \left[ \min_{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[ c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| a_t^f = a, w_t^f = w, \boldsymbol{\theta}_t^f = \boldsymbol{\theta}_0^f \right] \right\} \right] \right\} \tag{4.35}$$

$$= \underset{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)}{\arg\min} \left\{ c_t^f(a, w; \rho_t^f, \lambda_t^f) + \mathbb{E}_{\boldsymbol{\theta}_k^f} \left[ \min_{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)} \sum_{k=t+1}^{\infty} \gamma^{k-t} \left[ c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| s_{t+1}^f = a \right] \right] \right\} \tag{4.36}$$

$$V^f\left(s^f, r^f; \rho^f, \lambda^f\right) := \min_{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)} \left\{ \mathbb{E}_{\boldsymbol{\theta}_k^f} \left[ \min_{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[ c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| a_t^f = a, w_t^f = w, \boldsymbol{\theta}_t^f = \boldsymbol{\theta}^f \right] \right\} \right] \right\} \tag{4.37}$$

$$\begin{aligned} \bar{V}^f(s^f) :=& \mathbb{E}_{\boldsymbol{\theta}^f} \left[ \min_{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)} \left\{ \mathbb{E}_{\boldsymbol{\theta}_k^f} \left[ \min_{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[ c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| a_t^f = a, w_t^f = w, \boldsymbol{\theta}_t^f = \boldsymbol{\theta}^f \right] \right\} \right] \right\} \right] \\ =& \mathbb{E}_{\boldsymbol{\theta}^f} \min_{(w,a)\in\mathcal{X}(r^f, s^f)} \left\{ c_0^f(a, w; \rho^f, \lambda^f) + \gamma \bar{V}^f(a) \right\} \end{aligned} \tag{4.38}$$

---

influences not only the current cost $c_t^f(\cdot)$, but also future costs through the second term as well. Bellman equations can be leveraged for tackling such a DP. Under the stationarity assumption for variables $r_t^f$, $\rho_t^f$ and $\lambda_t^f$, the term accounting for the future cost can be rewritten in terms of the *stationary value function* $V^f\left(s^f, r^f; \rho^f, \lambda^f\right)$. This function, formally defined in (4.37), captures the minimum sum average cost for the "state" $(s^f, r^f)$, parametrized by $(\lambda^f, \rho^f)$, where for notational convenience, we define $\boldsymbol{\theta}^f := [r^f, \rho^f, \lambda^f]$.

### 4.15.2 Marginalized value-function

If one further assumes that price parameters and requests are i.i.d. across time, it can be shown that the optimal solution to (P2) can be expressed in terms of the *reduced value function* [108]

$$\bar{V}^f\left(s^f\right) := \mathbb{E}_{\boldsymbol{\theta}^f}\left[V^f\left(s^f, r^f; \rho^f, \lambda^f\right)\right], \tag{4.39}$$

where the expectation is w.r.t $\boldsymbol{\theta}^f$. Marginalization of the value function is important not only because it captures the average future cost of file $f$ for cache state $s^f \in \{0, 1\}$, but also because $\bar{V}^f(\cdot)$ is a function of a binary variable, and therefore its estimation requires only estimating two values. This is in contrast with the original four-dimensional value function in (4.37), whose estimation is more difficult due to its continuous arguments.

By rewriting the proposed alternative value function $\bar{V}^f(\cdot)$ in a recursive fashion as the summation of instantaneous cost and discounted future values $\bar{V}^f(\cdot)$, one readily arrives at the Bellman equation form provided in (4.38). Thus, the problem reduces to finding $\bar{V}^f(0)$ and $\bar{V}^f(1)$ for all $f$, after which the optimal fetch-cache decisions $(w_t^{f*}, a_t^{f*})$ are easily found as the solution to

$$\begin{aligned} \text{(P3)} \quad \min_{(w,a)} \quad & c_t^f(a, w; \rho_t^f, \lambda_t^f) + \gamma \bar{V}^f(a) \\ \text{s.t.} \quad & (w, a) \in \mathcal{X}(r_t^f, a_{t-1}^f). \end{aligned}$$

If the value-function is known, so that we have access to $\bar{V}^f(0)$ and $\bar{V}^f(1)$, the corresponding optimal (Bellman) decisions can be found as

$$w_t^f = a_t^f,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f \geq \lambda_t^f + \rho_t^f\}} \qquad \text{if } (r_t^f, s_t^f) = (0,0) \tag{4.40a}$$

$$w_t^f = 0,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f \geq \rho_t^f\}} \qquad \text{if } (r_t^f, s_t^f) = (0,1) \tag{4.40b}$$

$$w_t^f = 1,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f \geq \rho_t^f\}} \qquad \text{if } (r_t^f, s_t^f) = (1,0) \tag{4.40c}$$

$$w_t^f = 0,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f \geq \rho_t^f\}} \qquad \text{if } (r_t^f, s_t^f) = (1,1) \tag{4.40d}$$

where $\Delta\bar{V}_\gamma^f$ represents the *future* marginal cost, which is obtained as $\Delta\bar{V}_\gamma^f = \gamma(\bar{V}^f(1) - \bar{V}^f(0))$, and $\mathbb{I}_{\{\cdot\}}$ is an indicator function that yields value one if the condition in the argument holds, and zero otherwise.

The next subsection discusses how $\bar{V}^f(0)$ and $\bar{V}^f(1)$ can be calculated, but first a remark is in order.

**Remark 1 (Augmented value functions)**. The value function $\bar{V}^f(s^f)$ can be redefined to account for extra information on $r_t^f$, $\rho_t^f$ or $\lambda_t^f$, if available. For instance, consider the case where the distribution of $r_t^f$ can be parametrized by $p^f$, which measures content "popularity" [27]. In such cases, the value function can incorporate the popularity parameter as an additional input to yield $\bar{V}^f(s^f, p^f)$. Consequently, the optimal decisions will depend not only on the current requests and prices, but also on the (current) popularity $p^f$. This indeed broadens the scope of the proposed approach, as certain types of *non-stationarity* in the distribution of $r_t^f$ can be handled by allowing $p^f$ to (slowly) vary with time.

$$\begin{aligned}\bar{V}_1 &= (1-p)\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[\gamma\bar{V}_0(1-a)+(\rho+\gamma\bar{V}_1)a\,\Big|s=1,r=0\Big]\Big)+p\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[\gamma\bar{V}_0(1-a)+(\rho+\gamma\bar{V}_1)a\,\Big|s=1,r=1\Big]\Big)\\ &= \gamma\bar{V}_0\Pr\left(\rho\geq\Delta\bar{V}_\gamma\right)+\mathbb{E}\Big(\rho+\gamma\bar{V}_1\Big|\rho<\Delta\bar{V}_\gamma\Big)\Pr\left(\rho<\Delta\bar{V}_\gamma\right) \qquad (4.41)\end{aligned}$$

$$\begin{aligned}\bar{V}_0 &= (1-p)\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[\gamma\bar{V}_0(1-a)+(\lambda+\rho+\gamma\bar{V}_1)a\,\Big|s=0,r=0\Big]\Big) \qquad (4.42)\\ &+ p\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[(\lambda+\gamma\bar{V}_0)(1-a)+(\lambda+\rho+\gamma\bar{V}_1)a\,\Big|s=0,r=1\Big]\Big)\\ &= (1-p)\Big(\gamma\bar{V}_0\Pr\left(\lambda+\rho\geq\Delta\bar{V}_\gamma\right)+\mathbb{E}\left(\lambda+\rho+\gamma\bar{V}_1\,\Big|\lambda+\rho<\Delta\bar{V}_\gamma\right)\Pr\left(\lambda+\rho<\Delta\bar{V}_\gamma\right)\Big) \qquad (4.43)\\ &+ p\Big(\mathbb{E}[\lambda]+\gamma\bar{V}_0\Pr\left(\rho\geq\Delta\bar{V}_\gamma\right)+\mathbb{E}\left(\rho+\gamma\bar{V}_1\,\Big|\rho\leq\Delta\bar{V}_\gamma\right)\Pr\left(\rho\leq\Delta\bar{V}_\gamma\right)\Big)\end{aligned}$$

**Algorithm 6** Value iteration for finding $\bar{V}(\cdot)$

**Initialize** $\gamma<1$, probability density function of $\rho,\lambda$ and $r$, precision $\epsilon$, in order to stop
**Initialize** $\bar{V}_0,\bar{V}_1$
While $|\bar{V}_s^i-\bar{V}_s^{i+1}|<\epsilon; s\in\{0,1\}$
For $s=0,1$
$\bar{V}_s^{i+1}=\mathbb{E}_{r,\rho,\lambda}\min_{(w,a)\in\mathcal{X}(r,s)}\left\{c(a,w;\rho,\lambda)+\gamma\bar{V}_a^i\right\}\ i=i+1$

### 4.15.3 Value function in closed form

For notational brevity, we have removed the superscript $f$ in this subsection, and use $\bar{V}_0$ and $\bar{V}_1$ in lieu of $\bar{V}(0)$, and $\bar{V}(1)$. Denoting the *long-term* popularity of the content as $p:=\mathbb{E}[r_t]$, using the expressions for the optimal actions in (4.40a)-(4.40d), and leveraging the independence among $r_t$, $\lambda_t$, and $\rho_t$, the expected cost-to-go function can be readily derived as in (4.41)-(4.43). The expectation in (4.41) is w.r.t. $\rho$, while that in (4.42) is w.r.t. both $\lambda$ and $\rho$.

Solving the system of equations in (4.41)-(4.43) yields the optimal values for $\bar{V}_1$ and $\bar{V}_0$. A simple solver would be to perform exhaustive search over the range of these values since it is only a two-dimensional search space. However, a better alternative to solving the given system of equations is to rely on the well known *value iteration* algorithm. In short, this is an offline algorithm, which per iteration $i$ updates the estimates $\{\bar{V}_0^{i+1},\bar{V}_1^{i+1}\}$ by computing the expected cost using $\{\bar{V}_0^i,\bar{V}_1^i\}$, until the desired accuracy is achieved. This scheme is tabulated in detail in Algorithm 1, for which the distributions of $r,\rho,\lambda$ are assumed to be known.

**Remark 2 (Finite-horizon approximate policies).** In the proposed algorithms, namely exhaustive search as well as Algorithm 1, the solver is required to compute an expectation, which can be burdensome in setups with limited computational resources. For such scenarios, the class

of finite-horizon policies emerges as a computationally affordable suboptimal alternative. The idea behind such policies is to truncate the infinite summation in the objective of (P1); thus, only considering the impact of the current decision on a few number of future time instants denoted by $h$, typically referred to as the *horizon*. The extreme case of a finite-horizon policy is that of a *myopic policy* with $h = 0$, which ignores any future impact of current decision, a.k.a. zero-horizon policy, thus taking the action which minimizes the instantaneous cost. This is equivalent to setting the future marginal cost to zero, hence solving (4.40a)-(4.40d) with $\Delta\bar{V}_\gamma = \Delta\bar{V}_\gamma^{h=0} = 0$.

Another commonly used alternative is to consider the impact of the current decision for only the next time instant, which corresponds to the so-called horizon-1 policy. This entails setting the future cost at $h = 1$ as $\Delta\bar{V}_\gamma^{h=1} = \gamma(\bar{V}_1^{h=0} - \bar{V}_0^{h=0})$ with

$$\begin{aligned}
\bar{V}_0^{h=0} &= (1-p)\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0} | s = 0, r = 0] \\
&+ p\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0} | s = 0, r = 1] = p\mathbb{E}[\lambda] \qquad (4.44) \\
\bar{V}_1^{h=0} &= (1-p)\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0} | s = 1, r = 0] \\
&+ p\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0} | s = 1, r = 1] = 0, \qquad (4.45)
\end{aligned}$$

which are then substituted into (4.40a)-(4.40d) to yield the actions $w^{h=1}$ and $a^{h=1}$. The notation $w^{h=0}$ and $a^{h=0}$ in (4.44) and (4.45) is used to denote the actions obtained when (4.40a)-(4.40d) are solved using the future marginal cost at horizon zero $\Delta\bar{V}_\gamma^{h=0}$, which as already mentioned, is zero; that is, under the myopic policy in lieu of the original optimal solution. Following an inductive argument, the future marginal cost at $h = 2$ is obtained as $\Delta\bar{V}_\gamma^{h=2} = \gamma(\bar{V}_1^{h=1} - \bar{V}_0^{h=1})$ with

$$\begin{aligned}
\bar{V}_0^{h=1} &= (1-p)\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma\bar{V}_a^{h=0} | s = 0, r = 0] \\
&+ p\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma\bar{V}_a^{h=0} | s = 0, r = 1], \\
\bar{V}_1^{h=1} &= (1-p)\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma\bar{V}_a^{h=0} | s = 1, r = 0] \\
&+ p\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma\bar{V}_a^{h=0} | s = 1, r = 1],
\end{aligned}$$

which will allow to obtain the actions $w^{h=2}$ and $a^{h=2}$. While increasing horizons can be used, as $h$ grows large, solving the associated equations becomes more difficult and computation of the optimal stationary policies, is preferable.

$$Q\left(s_t, r_t, w_t, a_t; \rho_t, \lambda_t\right) := \mathbb{E}\left[\min_{\left\{(w_k,a_k)\in\mathcal{X}(r_k,a_{k-1})\right\}_{k=t+1}^{\infty}} \left\{\sum_{k=t}^{\infty} \gamma^{k-t}\left[c_k(a_k, w_k; \rho_k, \lambda_k)\Big| a_t, w_t, \boldsymbol{\theta}_t = \boldsymbol{\theta}\right]\right\}\right] \tag{4.46}$$

$$= \underbrace{c_t(a_t, w_t; \rho_t, \lambda_t)}_{\text{Immediate cost}} + \gamma\, \underbrace{\mathbb{E}\left[\min_{\left\{(w_k,a_k)\in\mathcal{X}(r_k,a_{k-1})\right\}_{k=t+1}^{\infty}} \left\{\sum_{k=t+1}^{\infty} \gamma^{k-(t+1)}\left[c_k(a_k, w_k; \rho_k, \lambda_k)\Big| s_{t+1} = a_t\right]\right\}\right]}_{\text{Average minimum future cost}} \tag{4.47}$$

$$\begin{aligned}\bar{Q}_{r_t,s_t}^{w_t,a_t} &:= \mathbb{E}_{\rho_t,\lambda_t}\left[Q\left(s_t, r_t, w_t, a_t; \rho_t, \lambda_t\right)\right], \quad \forall (w_t, a_t) \in \mathcal{X}(r_t, a_{t-1}) \\ &= \mathbb{E}_{\rho_t,\lambda_t}\left[c_t(a_t, w_t; \rho_t, \lambda_t)\right] + \gamma\left[\mathbb{E}_{\boldsymbol{\theta}_{t+1}}\left[Q\left(s_{t+1}, r_{t+1}, w_{t+1}^*, a_{t+1}^*; \rho_{t+1}, \lambda_{t+1}\right)\Big|\boldsymbol{\theta}_{t+1}, s_{t+1} = a_t\right]\right].\end{aligned} \tag{4.48}$$

$$\begin{aligned}\bar{Q}_{r,s}^{w,a} = \mathbb{E}[\lambda]w + \mathbb{E}[\rho]a &+ \gamma(1-p)\sum_{\forall (z_1,z_2)\in\mathcal{X}(0,a)} \bar{Q}_{0,a}^{z_1,z_2}\Pr\left((w_{t+1}^*, a_{t+1}^*) = (z_1, z_2)|(s_{t+1}, r_{t+1}) = (a, 0)\right) \\ &+ \gamma p \sum_{\forall (z_1,z_2)\in\mathcal{X}(1,a)} \bar{Q}_{1,a}^{z_1,z_2}\Pr\left((w_{t+1}^*, a_{t+1}^*) = (z_1, z_2)|(s_{t+1}, r_{t+1}) = (a, 1)\right).\end{aligned} \tag{4.49}$$

### 4.15.4 State-action value function ($Q$-function):

In many practical scenarios, knowing the underlying distributions for $\rho_t$, $\lambda_t$ and $r_t$ may not be possible, which motivates the introduction of online solvers that can learn the parameters on-the-fly. As clarified in the ensuing sections, in such scenarios, the so-called $Q$-function (or state-action value function) becomes helpful, since there are rigorous theoretical guarantees on the convergence of its stochastic estimates; see [136] and [189]. Motivated by this fact, instead of formulating our dynamic program using the value (cost-to-go) function, we can alternatively formulate it using the $Q$-function. Aiming at an online solver, let us tackle the DP through the estimation (learning) of the $Q$-function. Equation[2] (4.46) defines the $Q$-function for a specific file under a given state $(s_t, r_t)$, parametrized by cost parameters $(\rho_t, \lambda_t)$. Under stationarity distribution assumption for $\{\rho_t, \lambda_t, r_t\}$, the $Q$-function $Q\left(s_t, r_t, w_t, a_t; \rho_t, \lambda_t\right)$ accounts for the minimum average aggregate cost at state $(s_t, r_t)$, and taking specific fetch-cache decision $(w_t, a_t)$ as for the first decision, while followed by the best possible decisions in next slots. This function is parametrized by $(\rho_t, \lambda_t)$ since while making the current cache-fetch decision, the current values for these cost parameters are assumed to be known. The original $Q$-function in (4.46) needs to be learned over all values of $\{s_t, r_t, w_t, a_t, \rho_t, \lambda_t, r_t\}$, thus suffering from the curse of dimensionality, especially due to the fact that $\rho_t$ and $\lambda_t$ are continuous variables.

To alleviate this burden, we define the *marginalized* $Q$-function $Q(s_t, r_t, w_t, a_t)$ in (4.48).

[2]Equations (4.46)-(4.48), and (4.49) are shown at the top of page 7.

By changing the notation for clarity of exposition, the marginalized $Q$-function, $\bar{Q}_{r_t,s_t}^{w_t,a_t}$, can be rewritten in a more compact form as

$$\bar{Q}_{r_t,s_t}^{w_t,a_t} = \mathbb{E}\Big[\lambda_t w_t + \rho_t a_t + \gamma \bar{Q}_{r_{t+1},a_t}^{w_{t+1}^*,a_{t+1}^*}\Big] \; \forall (w_t,a_t) \in \mathcal{X}(r_t, a_{t-1}). \tag{4.50}$$

Note that, while the marginalized value-function is only a function of the state, the marginalized $Q$-function depends on both the state $(r, s)$ and the immediate action $(w, a)$. The main reason one prefers to learn the value-function rather than the $Q$-function is that the latter is computationally more complex. To see this, note that the input space of $\bar{Q}_{r_t,s_t}^{w_t,a_t}$ is a four-dimensional binary space, hence the function has $2^4 = 16$ different inputs and one must estimate the corresponding $16$ outputs. Each of these possible values are called $Q$-factors, and under the stationarity assumption, they can be found using (4.49) defined for all $(r, s, w, a)$. In this expression, we have $(z_1, z_2) \in \{0, 1\}^2$ and the term $\Pr\big(\big(w_{t+1}^* a_{t+1}^*\big) = (z_1, z_2)\big)$ stands for the probability of specific action $(z_1, z_2)$ to be optimal at slot $t + 1$. This action is random because the optimal decision at $t + 1$ depends on $\rho_{t+1}$, $\lambda_{t+1}$ and $r_{t+1}$, which are not known at slot $t$. Although not critical for the discussion, if needed, one can show that half of the 16 $Q$-factors can be discarded, either for being infeasible – recall that $(w_t, a_t) \in \mathcal{X}(r_t, a_{t-1})$ – or suboptimal. This means that (4.49) needs to be computed only for $8$ of the $Q$-factors.

From the point of view of offline estimation, working with the $Q$-function is more challenging than working with the $V$-function, since more parameters need to be estimated. In several realistic scenarios however, the distributions of the state variables are unknown, and one has to resort to stochastic schemes in order to learn the parameters on-the-fly. In such scenarios, the $Q$-function based approach is preferable, because it enables learning the optimal decisions in an online fashion even when the underlying distributions are unknown.

### 4.15.5 Stochastic policies: Reinforcement learning

As discussed in Section 4.15.3, there are scenarios where obtaining the optimal value function (and, hence, the optimal stationary policy associated with it) is not computationally feasible. The closing remark in that section discussed policies which, upon replacing the optimal value function with approximations easier to compute, trade reduced complexity for loss in optimality. However, such reduced-complexity methods still require knowledge of the state distribution [cf. (4.44) and (4.45)]. In this section, we discuss stochastic schemes to approximate the value function under

unknown distributions, thus relaxing assumption AS2 made earlier. The policies resulting from such stochastic methods offer a number of advantages since they: (a) incur a reduced complexity; (b) do not require knowledge of the underlying state distribution; (c) are able to handle some non-stationary environments; and in some cases, (d) they come with asymptotic optimality guarantees. To introduce this scheme, we first start by considering a simple method that updates stochastic estimates of the value function itself, and then proceed to a more advanced method which tracks the value of the $Q$-function. Specifically, the presented method is an instance of the celebrated $Q$-learning algorithm [188], which is the workhorse of stochastic approximation in DP.

### Stochastic value function estimates

The first method relies on current stochastic estimates of $\bar{V}_0$ and $\bar{V}_1$, denoted by $\hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t)$ at time $t$ (to be defined rigorously later). Given $\hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t)$ at time $t$, the (stochastic) actions $\hat{w}_t$ and $\hat{a}_t$ are taken via solving (4.40a)-(4.40d) with $\Delta\bar{V}_\gamma = \gamma(\hat{\bar{V}}_0(t) - \hat{\bar{V}}_1(t))$. Then, stochastic estimates of the value functions $\hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t)$ are updated as

- If $s_t = 0$, then $\hat{\bar{V}}_1(t+1) = \hat{\bar{V}}_1(t)$ and $\hat{\bar{V}}_0(t+1) = (1-\beta_t)\hat{\bar{V}}_0(t)+\beta_t(\hat{w}_t\lambda_t+\hat{a}_t\rho_t+\gamma\hat{\bar{V}}_{\hat{a}_t}(t))$;
- If $s_t = 1$, then $\hat{\bar{V}}_0(t+1) = \hat{\bar{V}}_0(t)$ and $\hat{V}_1(t+1) = (1-\beta_t)\hat{\bar{V}}_1(t) + \beta_t(\hat{w}_t\lambda_t + \hat{a}_t\rho_t + \gamma\hat{\bar{V}}_{\hat{a}_t}(t))$;

where $\beta_t > 0$ denotes the stepsize. While easy to implement (only two recursions are required), this algorithm has no optimality guarantees.

### *Q*-learning algorithm

Alternatively, one can run a stochastic approximation algorithm on the $Q$-function. This entails replacing the $Q$-factors $\bar{Q}^{w,a}_{r,s}$ with stochastic estimates $\hat{\bar{Q}}^{w,a}_{r,s}(t)$. To describe the algorithm, suppose for now that at time $t$, the estimates $\hat{\bar{Q}}^{w,a}_{r,s}(t)$ are known for all $(r, s, w, a)$. Then, in a given slot $t$ with $(r_t, s_t)$, action $(\hat{w}^*_t, \hat{a}^*_t)$ is obtained via either an exploration or an exploitation step. When exploring, which happens with a small probability $\epsilon_t$, a random and feasible action $(\hat{w}^*_t, \hat{a}^*_t) \in \mathcal{X}(r_t, a_{t-1})$ is taken. In contrast, in the exploitation mode, which happens with a

probability $1-\epsilon_t$, the optimal action according to the current estimate of $\hat{\bar{Q}}^{w,a}_{r,s}(t)$ is

$$+1\,(\hat{w}^*_t, \hat{a}^*_t) := \underset{(w,a)\in\mathcal{X}(r_t,a_{t-1})}{\arg\min} \; w\lambda_t + a\rho_t + \gamma \hat{\bar{Q}}^{w,a}_{r_t,s_t}(t). \tag{4.51}$$

After taking this action, going to next slot $t+1$, and observing $\rho_{t+1}, \lambda_{t+1}$, and $r_{t+1}$, the $Q$-function estimate is updated as

$$\hat{\bar{Q}}^{w,a}_{r,s}(t+1) = \begin{cases} \hat{\bar{Q}}^{w,a}_{r,s}(t) & \text{if} \quad (r,s,w,a) \neq (r_t, s_t, \hat{w}^*_t, \hat{a}^*_t) \\ (1-\beta_t)\hat{\bar{Q}}^{\hat{w}^*_t,\hat{a}^*_t}_{r_t,s_t}(t) + \beta_t\Big(\hat{w}^*_t\lambda_t + \hat{a}^*_t\rho_t + \gamma \hat{\bar{Q}}^{\hat{w}^*_{t+1},\hat{a}^*_{t+1}}_{r_{t+1},\hat{a}^*_t}(t)\Big) & \text{o.w.,} \end{cases} \tag{4.52}$$

where "o.w." stands for "otherwise", $(\hat{w}^*_{t+1}, \hat{a}^*_{t+1})$ is the optimal action for the next slot and, if needed, the stepsize $\beta_t$ can be adapted for each particular state-action pair. This update rule describes one of the possible implementations of the $Q$-learning algorithm, which was originally introduced in [188]. This online algorithm enables making sequential decisions in an unknown environment, and is guaranteed to learn optimal decision-making rules under certain conditions specified next [189].

Regarding convergence of the Q-learning algorithm, the following necessary conditions should hold [189, 24]: (c1) all feasible state $(r,s)$ and action $(w,a)$ pairs should be continuously updated; and, (c2) the learning rate $\beta_t$ should be a diminishing step size. Under these conditions, the factors $\hat{\bar{Q}}^{w,a}_{r,s}$ converge to their optimal value $\bar{Q}^{*w,a}{}_{r,s}$ with probability 1; see [24] for details. To satisfy (c1), various exploration-exploitation algorithms have been proposed [147, p. 839]. Particularly, any such scheme needs to be *greedy in the limit of infinite exploration*, or GLIE [147, p. 840]. A common choice to meet this property is the $\epsilon$-greedy approach, as considered in this work, with $\epsilon_t = 1/t$, which provides guaranteed yet slow convergence. In practice however, $\epsilon_t$ can be set to a small value for faster convergence [24], [136]. To satisfy the diminishing step size rule in (c2), let us define $t^{w,a}_{r,s}$ as the index of the $t$-th time when the state-action pair $(r,s)$ and $(w,a)$ is visited, and updated with the corresponding learning rate $\beta_{t^{w,a}_{r,s}}$. Condition (c2) requires $\sum_{t=1}^{\infty}\beta_{t^{w,a}_{r,s}} = \infty$, and $\sum_{t=1}^{\infty}\beta^2_{t^{w,a}_{r,s}} < \infty$ to hold for all feasible state-action pairs, a typical choice for which is setting $\beta_{t^{w,a}_{r,s}} = 1/t$. Similar to $\epsilon_t$, a constant but small learning rate is preferred in practice as it endows the algorithm to adapt to possible changes of pertinent

**Algorithm 7** $Q$-learning algorithm to estimate $\bar{Q}_{r,s}^{w,a}$ for a given file $f$

**Initialize** $\hat{\bar{Q}}_{r,s}^{w,a}(1) = 0$, $s_1 = 0$, $\{r_0, \rho_0, \lambda_0\}$ are revealed
**Output** $\hat{\bar{Q}}_{r,s}^{w,a}(t+1)$
For $t = 1, 2, \ldots$
For the current state $(r_t, s_t)$, choose $(\hat{w}_t^*, \hat{a}_t^*)$

$$(\hat{w}_t^*, \hat{a}_t^*) = \begin{cases} \text{Solve (4.51)w.p. } 1-\epsilon_t \\ \text{random } (w,a) \in \mathcal{X}_t(r_t, s_t) \text{w.p. } \epsilon_t \end{cases} \tag{4.53}$$

Update state $s_{t+1} = \hat{a}_t^*$
Request and cost parameters, $\boldsymbol{\theta}_{t+1}$, are revealed
Update $Q$ factor by (4.52)

parameters in dynamic settings.

The resultant algorithm for the problem at hand is tabulated in Algorithm 7. It is important to stress that in our particular case, we expect the algorithm to converge fast. That is the case because, under the decomposition approach followed in this paper as well as the introduction of the marginalized $Q$-function, the state-action space of the resultant $Q$-function has very low dimension and hence, only a small number of $Q$-factors need to be estimated.

## 4.16 Limited storage and back-haul transmission rate via dynamic pricing

So far, we have considered that the prices $\{\rho_t^f, \lambda_t^f\}$ are provided by the system, and we have not assumed any explicit limits (bounds) neither on the capacity of the local storage nor on the back-haul transmission link between the SB and the cloud. In this section, we discuss such limitations, and describe how by leveraging dual decomposition techniques, one can redefine the prices $\{\rho_t^f, \lambda_t^f\}$ to account for capacity constraints.

### 4.16.1 Limiting the instantaneous storage rate

In this subsection, practical limitations on the cache storage capacity are explored. Suppose that the SB is equipped with a *single* memory device that can store $M$ files. Clearly, the cache

decisions should then satisfy the following constraint per time slot

$$\text{C4:} \quad \sum_{f=1}^{F} a_t^f \sigma^f \leq M, \quad t = 1, 2, \ldots$$

In order to respect such hard capacity limits, the original optimization problem in (P1) can be simply augmented with C4, giving rise to a new optimization problem which we will refer to as (P4). Solving (P4) is more challenging than (P1), since the constraints in C4 must be enforced at each time instant, which subsequently couples the optimization across files. In order to deal with this, one can dualize C4 by augmenting the cost with the primal-dual term $\mu_t(\sum_{f=1}^{F} \sigma_f a_t^f - M)$, where $\mu_t$ denotes the Lagrange multiplier associated with the capacity constraint C4. The resultant problem is separable across files, but requires finding $\mu_t^*$, the optimal value of the Lagrange multiplier, at each and every time instant.

If the solution to the original unconstrained problem (P1) does satisfy C4, then $\mu_t^* = 0$ due to complementary slackness. On the other hand, if the storage limit is violated, then the constraint is active, the Lagrange multiplier satisfies $\mu_t^* > 0$, and its exact value must be found using an iterative algorithm. Once the value of the multiplier is known, the optimal actions associated with (P4) can be found using the expressions for the optimal solution to (P1) provided that the original storage price $\rho_t^f$ is replaced with the new storage price $\rho_{t,aug}^f = \rho_t^f + \mu_t^* \sigma_f$ [cf. (4.31)]. The reason for this will be explained in detail in the following subsection, after introducing the ensemble counterpart of C4.

### 4.16.2 Limiting the long-term storage rate

Consider now the following constraint [cf. C4]

$$\text{C5:} \quad \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E}\left[\sum_{f=1}^{F} a_k^f \sigma^f\right] \leq \sum_{k=t}^{\infty} \gamma^{k-t} M' \tag{4.54}$$

where the expectation is taken w.r.t. all state variables. By setting $M' = M$, one can view C5 as a relaxed version of C4. That is, while C4 enforces the limit to be respected at every time instant, C5 only requires it to be respected *on average*. From a computational perspective, dealing with C5 is easier than its instantaneous counterpart, since in the former only one constraint is enforced and, hence, only one Lagrange multiplier, denoted by $\mu$, must be found. This comes at

the price that guaranteeing C5 with $M' = M$ does not imply that C4 will always be satisfied. Alternatively, enforcing C5 with $M' < M$, will increase the probability of satisfying C4, since the solution will guarantee that "on average" there exists free space on the cache memory. A more formal discussion on this issue will be provided in the remark closing the subsection.

To describe in detail how accounting for C5 changes the optimal schemes, let (P5) be the problem obtained after augmenting (P1) with C5. Suppose now that to solve (P5) we dualize the single constraint in C5. Rearranging terms, the augmented objective associated with (P5) is given by

$$\sum_{k=t}^{\infty}\sum_{f=1}^{F}\gamma^{k-t}\mathbb{E}\left[c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right) + \mu a_k^f \sigma^f\right] - \sum_{k=t}^{\infty}\gamma^{k-t}M'. \tag{4.55}$$

Equation (4.55) demonstrates that after dualization and provided that the multiplier $\mu$ is known, decisions can be optimized separately across files. To be more precise, note that the term $\sum_{k=t}^{\infty}\gamma^{k-t}M'$ in the objective is constant, so that it can be ignored, and define the modified instantaneous cost as

$$\begin{aligned}\check{c}_k^f &:= c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right) + \mu\sigma^f a_k^f \\ &= \left(\rho_k^f + \mu\sigma^f\right)a_k^f + \lambda_k^f w_k^f.\end{aligned} \tag{4.56}$$

The last equation not only reflects that the dualization indeed facilitates separate per-file optimization, but it also reveals that term $\mu\sigma^f$ can be interpreted as an additional storage cost associated with the long-term caching constraint. More importantly, by defining the modified (augmented) prices $\rho_{t,\text{aug}}^f := \rho_t^f + \mu\sigma^f$ for all $t$ and $f$, the optimization of (4.56) can be carried out with the schemes presented in the previous sections, provided that $\rho_t^f$ is replaced with $\rho_{t,\text{aug}}^f$.

Note however that in order to run the optimal allocation algorithm, the value of $\mu$ needs to be known. Since the dual problem is always convex, one option is to use an iterative dual subgradient method, which computes the satisfaction/violation of the constraint C5 per iteration [135], [26, p.223]. Clearly, this requires knowledge of the state distribution, since the constraint involves an expectation. When such knowledge is not available, or when the computational complexity to carry out the expectations cannot be afforded, stochastic schemes are worth considering. For the particular case of estimating Lagrange multipliers associated with long-term constraints, a simple but powerful alternative is to resort to *stochastic dual* subgradient schemes [135], [26],

which for the problem at hand, estimate the value of the multiplier $\mu$ at every time instant $t$ using the update rule

$$\hat{\mu}_{t+1} = \left[\hat{\mu}_t + \zeta\left(\sum_{f=1}^{F} \hat{a}_t^{f*}\sigma^f - M'\right)\right]^+. \tag{4.57}$$

In the last expression, $\zeta > 0$ is a (small) positive constant, the update multiplied by $\zeta$ corresponds to the violation of the constraint after removing the expectation, the notation $[\cdot]^+$ stands for the $\max\{0, \cdot\}$, and $\hat{a}_t^{f*}$ denotes the optimal caching actions obtained with the policies described in Section 4.15 provided that $\rho_t^f$ is replaced by $\hat{\rho}_{t,\text{aug}}^f = \rho_t^f + \hat{\mu}_t\sigma^f$.

We next introduce another long-term constraint that can be considered to limit the storage rate. This constraint is useful not only because it gives rise to alternative novel caching-fetching schemes, but also because it will allow us to establish connections with well-known algorithms in the area of congestion control and queue management. To start, define the variables $\alpha_{in,t}^f := [a_t^f - s_t^f]^+$ and $\alpha_{out,t}^f := [s_t^f - a_t^f]^+$ for all $f$ and $t$. Clearly, if $\alpha_{in,t}^f = 1$, then content $f$ that was not in the local cache at time $t-1$, has been stored at time $t$; and as a result, less storage space is available. On the other hand, if $\alpha_{out,t}^f = 1$, then content $f$ was removed from the cache at time $t$, thus freeing up new storage space. With this notation at hand, we can consider the long term constraint

$$\text{C6: } \sum_{k=t}^{\infty}\gamma^{k-t}\mathbb{E}\left[\sum_{f=1}^{F}\alpha_{in,k}^f\sigma^f\right] \leq \sum_{k=t}^{\infty}\gamma^{k-t}\mathbb{E}\left[\sum_{f=1}^{F}\alpha_{out,k}^f\sigma^f\right], \tag{4.58}$$

which basically ensures the long-term stability of the local-storage. That is, the amount of data stored in the local memory is no larger than that taken out from the memory, guaranteeing that in the long term stored data does not grow unbounded.

To deal with C6 we can follow an approach similar to that of C5, under which we first dualize C6 and then use a stochastic dual method to estimate the associated dual variable. With a slight abuse of notation, supposing that the Lagrange multiplier associated with stability is by also denoted $\mu$, the counterpart of (4.57) for the constraint C6 is

$$\hat{\mu}_{t+1} = \left[\hat{\mu}_t + \zeta\sum_{f=1}^{F}[\hat{a}_t^{f*} - s_t^f]^+ - [s_t^f - \hat{a}_t^{f*}]^+\right]^+. \tag{4.59}$$

Note that the update term in the last iteration follows after removing the expectations in C6 and replacing $\alpha^f_{in,t}$, and $\alpha^f_{out,t}$ with their corresponding definitions. The modifications that the expressions for the optimal policies require to account for this constraint are a bit more intricate. If $s^f_t = 0$, the problem structure is similar to that of the previous constraints, and we just need to replace $\rho^f_t$ with $\hat{\rho}^f_{t,\text{aug}} = \rho^f_t + \hat{\mu}_t \sigma^f$. However, if $s^f_t = 1$, it turns out that: i) deciding $\hat{a}^{f*}_t = 1$ does not require modifying the caching price, but ii) deciding $\hat{a}^{f*}_t = 0$ requires considering the *negative* caching price $-\hat{\mu}_t \sigma^f$. In other words, while our formulation in Section 4.15 only considers incurring a cost when $a^f_t = 1$ (and assumes that the instantaneous cost is zero for $a^f_t = 0$), to fully account for C6, we would need to modify our original formulation so that costs can be associated with the decision $a^f_t = 0$ as well. This can be done either by considering a new cost term or, simply by replacing $\gamma \bar{V}^f(0)$ by $\gamma \bar{V}^f(0) - \hat{\mu}_t \sigma^f$ in (4.40a)-(4.40d), which are Bellman's equations describing the optimal policies.

**Remark 3 (Role of the stochastic multipliers).** It is well-established that the Lagrange multipliers can be interpreted as the marginal price that the system must pay to (over-)satisfy the constraint they are associated with [26, p.241]. When using stochastic methods for estimating the multipliers, further insights on the role of the multipliers can be obtained [59, 122, 38]. Consider for example the update in (4.57). The associated constraint C5 establishes that the long-term storage rate cannot exceed $M'$. To guarantee so, the stochastic scheme updates the estimated price in a way that, if the constraint for time $t$ is oversatisfied, the price goes down, while if the constraint is violated, the price goes up. Intuitively, if the price estimate $\hat{\mu}_t$ is far from its optimal value and the constraint is violated for several consecutive time instants, the price will keep increasing, and eventually will take a value sufficiently high so that storage decisions are penalized/avoided. How quickly the system reacts to this violation can be controlled via the constant $\zeta$. Interestingly, by tuning the values of $M'$ and $\zeta$, and assuming some regularity properties on the distribution of the state variables, conditions under which deterministic short-term limits as those in C4 are satisfied can be rigorously derived; see, e.g., [38] for a related problem in the context of distributed cloud networks. A similar analysis can be carried out for the update in (4.59) and its associated constraint C6. Every time the instantaneous version of the constraint is violated because the amount of data stored in the memory exceeds the amount exiting the memory, the corresponding price $\hat{\mu}_t$ increases, thus rendering future storage decisions more costly. In fact, if we initialize the multiplier at $\hat{\mu}_t = 0$ and set $\zeta = 1$, then the corresponding price is the total amount of information stored at time $t$ in the local memory. In

other words, the update in (4.59) exemplifies how the dynamic prices considered in this paper can be used to account for the actual state of the caching storage. Clearly, additional mappings from the instantaneous storage level to the instantaneous storage price can be considered. The connections between stochastic Lagrange multipliers and storing devices have been thoroughly explored in the context of demand response, queuing management and congestion control. We refer the interested readers to, e.g., [59, 122].

### 4.16.3 Limits on the back-haul transmission rate

The previous two subsections dealt with limited caching storage, and how some of those limitations could be accounted for by modifying the caching price $\rho_t^f$. This section addresses limitations on the back-haul transmission rate between the SB and the cloud as well as their impact on the fetching price $\lambda_t^f$.

While our focus has been on optimizing the decisions at the SB, contemporary networks must be designed following a holistic (cross-layer) approach that accounts for the impact of local decisions on the rest of the network. Decomposition techniques (including those presented in this paper) are essential to that end [135]. For the system at hand, suppose that $\mathbf{x}_{CD}$ includes all variables at the cloud network, $\bar{C}_{CD}(\mathbf{x}_{CD})$ denotes the associated cost, and the feasible set $\mathcal{X}_{CD}$ accounts for the constraints that cloud variables $\mathbf{x}_{CD}$ must satisfy. Similarly, let $\mathbf{x}_{SB}$, $\bar{C}_{SB}(\mathbf{x}_{SB})$, and $\mathcal{X}_{SB}$ denote the corresponding counterparts for the SB optimization analyzed in this paper. Clearly, the fetching actions $w_t^f$ are included in $\mathbf{x}_{SB}$, while the variable $b_t$ representing back-haul transmission rate (capacity) of the connecting link between the cloud and the SB, is included in $\mathbf{x}_{CD}$. This transmission rate will depend on the resources that the cloud chooses to allocate to that particular link, and will control the communication rate (and hence the cost of fetching requests) between the SB and the cloud. As in the previous section, one could consider two types of capacity constraints

$$\text{C7}a: \quad \sum_{f=1}^{F} w_t^f \sigma^f \leq b_t, \quad t = 1, \ldots, \tag{4.60a}$$

$$\text{C7}b: \quad \sum_{k=t}^{\infty} \gamma^{k-t} \sum_{f=1}^{F} \mathbb{E}[w_t^f \sigma^f] \leq \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E}[b_k], \tag{4.60b}$$

depending on whether the limit is imposed in the short term or in the long term.

With these notational conventions, one could then consider the *joint* resource allocation problem

$$\begin{aligned} &\min_{\mathbf{x}_{CD},\mathbf{x}_{SB}} \bar{C}_{CD}(\mathbf{x}_{CD}) + \bar{C}_{SB}(\mathbf{x}_{SB}) \\ &\quad \text{s.t.} \quad \mathbf{x}_{CD} \in \mathcal{X}_{CD}, \;\; \mathbf{x}_{SB} \in \mathcal{X}_{SB}, \;\; (\text{C7}) \end{aligned} \tag{4.61}$$

where the constraint C7 – either the instantaneous one in C7$a$ or the lon-term version in C7$b$ – couples both optimizations. It is then clear that if one dualizes C7, and the value of the Lagrange multiplier associated with C7 is known, then two separate optimizations can be run: one focusing on the cloud network and the other one on the SB. For this second optimization, consider for simplicity that the average constraint in (4.60b) is selected and let $\nu$ denote the Lagrange multiplier associated with such a constraint. The optimization corresponding to the SB is then

$$\min_{\mathbf{x}_{SB}} \; \bar{C}_{SB}(\mathbf{x}_{SB}) + \sum_{k=t}^{\infty} \gamma^{k-t} \sum_{f=1}^{F} \mathbb{E}[w_t^f \nu \sigma^f] \quad \text{s.t.} \;\; \mathbf{x}_{SB} \in \mathcal{X}_{SB}. \tag{4.62}$$

Clearly, solving this problem is equivalent to solving the original problem in Section 4.15, provided that the original cost is augmented with the primal-dual term associated with the coupling constraint. To address the modified optimization, we will follow steps similar to those in Section 4.16.2, defining first a stochastic estimate of the Lagrange multiplier as

$$\hat{\nu}_{t+1} = \left[ \hat{\nu}_t + \zeta \left( \sum_{f=1}^{F} \hat{w}_t^{f*} \sigma^f - b_t \right) \right]^+, \tag{4.63}$$

and then obtaining the optimal caching-fetching decisions running the schemes in Section 4.15 after replacing the original fetching cost $\lambda_t^f$ with the augmented one $\lambda_{t,\text{aug}}^f = \lambda_t^f + \hat{\nu}_t \sigma_f$.

For simplicity, in this section we will limit our discussion to the case where $\hat{\nu}_t$ corresponds to the value of a Lagrange multiplier corresponding to a communication constraint. However, from a more general point of view, $\hat{\nu}_t$ represents the marginal price that the cloud network has to pay to transmit the information requested by the SB. In that sense, there exists a broad range of options to set the value of $\hat{\nu}_t$, including the congestion level at the cloud network (which is also represented by a Lagrange multiplier), or the rate (power) cost associated with the back-haul link. While a detailed discussion on those options is of interest, it goes beyond the scope of the

present work.

### 4.16.4 Modified online solver based on $Q$-learning

We close this section by providing an online reinforcement-learning algorithm that modifies the one introduced in Section 4.15 to account for the multipliers introduced in Section 4.16.

By defining per file cost $\hat{c}_k^f$ as

$$\hat{c}_k^f\left(w_k^f, a_k^f; \rho_k^f, \lambda_k^f, \hat{\mu}_k, \hat{\nu}_k\right) := \left(\rho_k^f + \hat{\mu}_k \sigma^f\right) a_k^f + \left(\lambda_k^f + \hat{\nu}_k \sigma^f\right) w_k^f \tag{4.64}$$

the problem of caching under limited cache capacity and back-haul link reduces to per file optimization as follows

$$\begin{aligned} \text{(P8)} \quad \min_{\{(w_k^f, a_k^f)\}_{k\geq t}} \; & \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E}\left[\hat{c}_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f, \hat{\mu}_k, \hat{\nu}_k\right)\right] \\ \text{s.t.} \quad & (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, a_{k-1}^f), \quad \forall f, \; k \geq t \end{aligned}$$

where the updated dual variables $\hat{\mu}_k$ and $\hat{\nu}_k$ are obtained respectively by iteration (4.57) and (4.63). If we plug $\hat{c}_k^f$ instead of $c_k^f$ into the marginalized $Q$-function in (4.48), then the solution for (P8) in current iteration $k$ for a given file $f$ can readily be found by solving

$$\arg\min_{(w,a)\in\mathcal{X}(r_t, a_{t-1})} \bar{Q}_{r_t, s_t}^{w,a} + w(\lambda_t + \hat{\nu}_t \sigma^f) + a(\rho_t + \hat{\mu}_t \sigma^f). \tag{4.65}$$

Thus, it suffices to form a marginalized $Q$-function for each file and solve (4.65), which can be easily accomplished through exhaustive search over $8$ possible cache-fetch decisions $(w, a) \in \mathcal{X}(r_t, a_{t-1})$.

To simplify notation and exposition, we focus on the *limited caching capacity* constraint, and suppose that the back-haul is capable of serving any requests, thus $\hat{\nu}_t = 0, \; \forall t$. Modifications to account also for $\hat{\nu}_t \neq 0$ are straightforward.

The modified $Q$-learning (MQ-learning) algorithm, tabulated in Algorithm 8, essentially learns to make optimal fetch-cache decisions while accounting for the limited caching capacity constraint in C4 and/or C5. In particular, to provide a computationally efficient solver the

stochastic updates corresponding to C5 are used. Subsequently, if C4 needs to be enforced, the obtained solution is projected into the feasible set through projection algorithm $\Pi_{\mathrm{C4}}(\cdot)$. The projection $\Pi_{\mathrm{C4}}(.)$ takes the obtained solution $\{\breve{w}_t^{f*}, \breve{a}_t^{f*}\}_{\forall f}$, the file sizes, as well as the marginalized $Q$-functions as input, and generates a feasible solution $\{w_t^{f*}, a_t^{f*}\}_{\forall f}$ satisfying C4 as follows: it sorts the files with $\breve{a}_t^{f*} = 1$ in ascending $Q$-function order, and caches the files with the lowest $Q$-values until the cache capacity is reached. Overall, our modified algorithm performs a "double" learning: i) by using reinforcement schemes it learns the optimal policies that map states to actions, and ii) by using a stochastic dual approach it learns the mechanism that adapt the prices to the saturation and congestion conditions in the cache. Given the operating conditions and the design approach considered in the paper, the proposed algorithm has moderate complexity, and thanks to the reduced input dimensionality, it also converges in a moderate number of iterations.

## 4.17 Numerical tests

In this section, we numerically assess the performance of the proposed approaches for learning optimal fetch-cache decisions. Two sets of numerical tests are provided. In the first set, summarized in Figs 4.14-4.18, the performance of the value iteration-based scheme in Alg. 1 is evaluated, and in the second set, summarized in Figs. 4.19-4.20, the performance of the $Q$-learning solver is investigated. In both sets, the cache and fetch cost parameters are drawn with equal probability from a finite number of values, where the mean is $\bar{\rho}^f$ and $\bar{\lambda}^f$, respectively. Furthermore, the request variable $r^f$ is modeled as a Bernoulli random variable with mean $p^f$, whose value indicates the popularity of file $f$.

In the first set, it is assumed that $p^f$ as well as the distribution of $\rho^f, \lambda^f$, are known a priori. Simulations are carried out for a content of unit size, and can be readily extended to files of different sizes. To help readability, we drop the superscript $f$ in this section.

Fig. 4.14 plots the sum average cost $\bar{\mathcal{C}}$ versus $\bar{\rho}$ for different values of $\bar{\lambda}$ and $p$. The fetching cost is set to $\bar{\lambda} \in \{43, 45, 50, 58\}$ for two different values of popularity $p \in \{0.3, 0.5\}$. As depicted, higher values of $\bar{\rho}, \bar{\lambda}, p$ generally lead to a higher average cost. In particular, when $\bar{\rho} \ll \bar{\lambda}$, caching is considerably cheaper than fetching, thus setting $a_t = 1$ is optimal for most $t$. As a consequence, the total cost linearly increases with $\bar{\rho}$ as most requests are met via cached contents rather than fetching. Interestingly, if $\bar{\rho}$ keeps increasing, the aggregate cost gradually

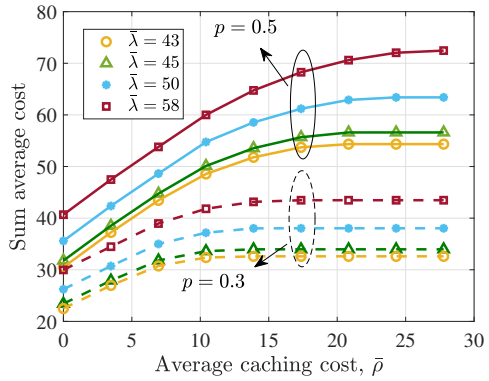



Figure 4.14: Average cost versus $\bar{\rho}$ for different values of $p, \bar{\lambda}$.

saturates and does not grow anymore. The reason behind this observation is the fact that, for very high values of $\bar{\rho}$, fetching becomes the optimal decision for meeting most file requests and, hence, the aggregate cost no longer depends on $\bar{\rho}$. While this behavior occurs for the two values of $p$, we observe that for the smallest one, the saturation is more abrupt and takes place at a lower $\bar{\rho}$. The intuition in this case is that for lower popularity values, the file is requested less frequently, thus the caching cost aggregated over a (long) period of time often exceeds the "reward" obtained when (infrequent) requests are served by the local cache. As a consequence, fetching in the infrequent case of $r_t = 1$ incurs less cost than the caching cost aggregated over time.

To corroborate these findings, Fig. 4.15 depicts the sum average cost versus $p$ for different values of $\bar{\rho}$ and $\bar{\lambda}$. The results show that for large values of $\bar{\rho}$, fetching is the optimal action, resulting in a linear increase in the total cost as $p$ increases. In contrast, for small values of $\bar{\rho}$, caching is chosen more frequently, resulting in a sub-linear cost growth.

To investigate the caching-versus-fetching trade-off for a broader range of $\bar{\rho}$ and $\bar{\lambda}$, let us define the *caching ratio* as the aggregated number of positive caching decisions (those for which $a_t = 1$) divided by the total number of decisions. Fig. 4.16 plots this ratio for different values of $(\bar{\rho}, \bar{\lambda})$ and fixed $p = 0.5$. As the plot demonstrates, when $\bar{\rho}$ is small and $\bar{\lambda}$ is large, files are cached almost all the time, with the caching ratio decreasing (non-symmetrically) as $\bar{\rho}$ increases and $\bar{\lambda}$ decreases. Similarly, the caching ratio is plotted by setting $p = 0.05$ in Fig. 4.17, in which fetching is mostly preferred over a wide range of storage costs due to the small value of $p$.

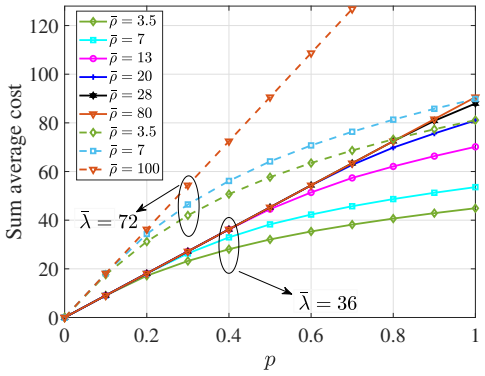



Figure 4.15: Average cost versus $p$ for different values of $\bar{\lambda}, \bar{\rho}$.

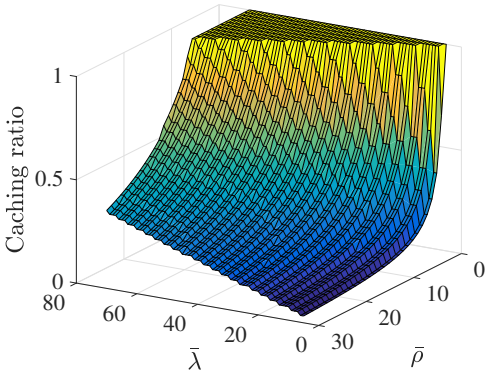



Figure 4.16: Caching ratio vs. $\bar{\rho}$ and $\bar{\lambda}$ for $p = 0.5$ and $s = r = 1$.

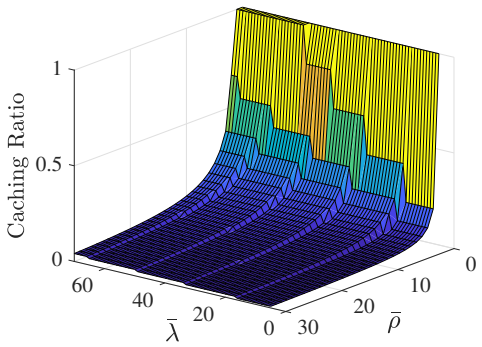



Figure 4.17: Caching ratio vs. $\bar{\rho}$ and $\bar{\lambda}$ for $p = 0.05$ and $s = r = 1$.

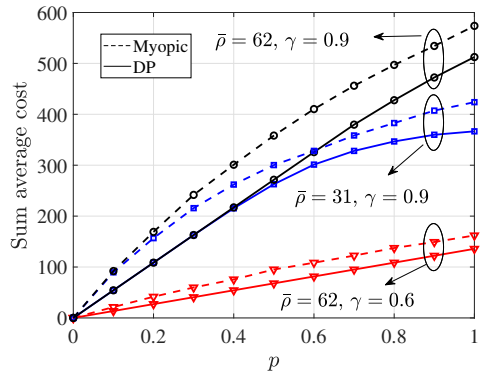



Figure 4.18: Performance of DP versus myopic caching for $\bar{\lambda} = 53$.

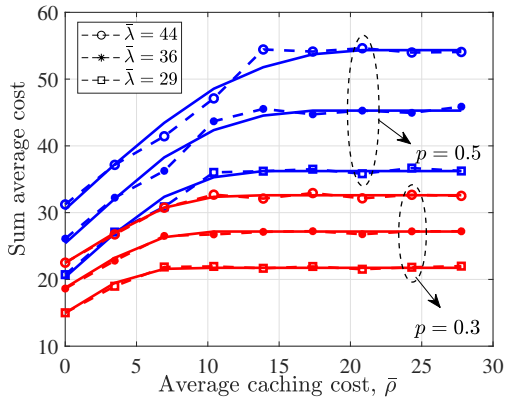



Figure 4.19: Average cost versus $\bar{\rho}$ for different values of $\bar{\lambda}, p$. Solid line is for value iteration while dashed lines are for $Q$-learning based solver.

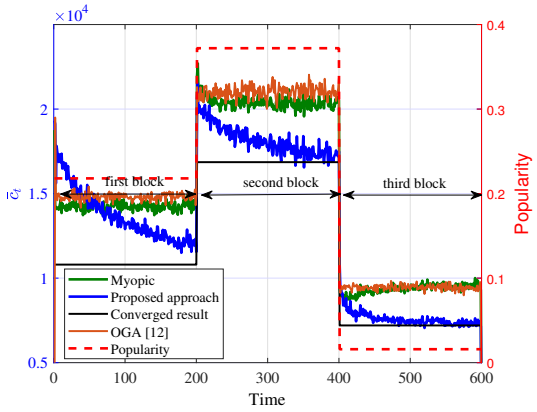



Figure 4.20: Averaged immediate cost over 1000 realizations in a non-stationary setting, and a sample from popularities.

Interestingly this is true despite high fetching costs as well, and can be intuitively explained as follows: due to low popularity, deciding to cache may result in idle storing of the file in cache, thus entailing an unnecessary aggregated caching cost before the stored file can be utilized to meet user request, rendering caching suboptimal. The comparison between Fig. 4.16 and 4.17 clearly demonstrates the effect of different values of $p$ on the performance of the cache-fetch decisions, while the proposed approach automatically adjusts to the underlying popularities.

Finally, Fig. 4.18 compares the performance of the proposed DP-based strategy with that of a myopic one. The myopic policy sets $a_t = 1$ if $\lambda_t > \rho_t$ and the content is locally available (either because $w_t = 1$ or because $s_t = 1$), and sets $a_t = 0$ otherwise. The results indicate that the proposed strategy outperforms the myopic one for all values of $\bar{\rho}, \bar{\lambda}, p$ and $\gamma$.

In the second set of tests, the performance of the online Q-learning solvers is investigated. As explained in Section 4.15, under the assumption that the underlying distributions are stationary, the performance of the Q-learning solver should converge to the optimal one found through the value iteration algorithm. Corroborating this statement, Fig. 4.19 plots the sum average cost $\bar{C}$ versus $\bar{\rho}$ of both the marginalized value iteration and the Q-learning solver, with $\bar{\lambda} \in \{29, 36, 44\}$ and $p \in \{0.3, 0.5\}$. The solid lines are obtained when assuming a priori knowledge of the distributions and then running the marginalized value iteration algorithm; the results and analysis are similar to the ones reported for Fig. 4.14. The dashed curves however, are found by assuming unknown distributions and running the Q-learning solver. Sum average cost is reported after first 1000 iterations. As the plot suggests, despite the lack of a priori knowledge on the distributions,

the Q-learning solver is able to find the optimal decision making rule. As a result, it yields the same sum average cost as that of value-iteration under known distributions.

The last experiment investigates the impact of the instantaneous cache capacity constraint in C4 as well as non-stationary distributions for popularities and costs. To this end, 1,000 different realizations (trajectories) of the random state processes are drawn, each of length $T = 600$. For every realization, the cost $c_t$ [cf. (4.33)] at each and every time instant is found, and the cost trajectory is averaged across the 1,000 realizations. Specifically, let $c_t^i$ denote the $i$th realization cost at time $t$, and define the averaged cost trajectory as $\bar{c}_t := \frac{1}{1000}\sum_{i=1}^{1000} c_t^i$. Fig. 4.20 reports the average trajectory of $\bar{c}_t$ in a setup where the total number of files is set to $F = 500$, the file sizes are drawn uniformly at random from the interval $[1, 100]$, and the total cache capacity is set to $40\%$ of the aggregate file size. Adopted parameters for the MQ-learning solver are set to $\beta_t = 0.3$, and $\epsilon = 0.01$. Three blocks of iterations are shown in the figure, where in each block a specific distribution of popularities and costs are considered. For instance, the dashed line shows the popularity of a specific file in one of the realizations, where in the fist block $p = 0.23$, in the second block $p = 0.37$, and in the third one $p = 0.01$. The cost parameters have means $\bar{\lambda} = 44, \bar{\rho} = 2$, $\bar{\lambda} = 40, \bar{\rho} = 5$, and $\bar{\lambda} = 38, \bar{\rho} = 2$ in the consecutive blocks, respectively.

As Fig. 4.20, the proposed MQ-learning algorithm incurs large costs during the first few iterations. Then, it gradually adapts to the file popularities and cost distributions, and learns how to make optimal fetch-cache decisions, decreasing progressively the cost in each of the blocks. To better understand the behavior of the algorithm and assess its effectiveness, we compare it with that of Online Gradient Ascent (OGA) [139] as a representative state-of-the-art method among the class of online expert algorithms, the myopic policy and the stationary policy serving as the benchmark, respectively. In contrast to the OGA method, our decision variables are not continuous, but binary. Hence, caching decisions in OGA are projected into the binary feasible set for fair comparison. In general, since OGA and the myopic caching only use blue the current state and requests, their performance is inferior to that of our proposed method, where knowledge of the underlying request and price distributions is carefully utilized. During the first iterations however, when the MQ-learning algorithm has not adapted to the distribution of pertinent parameters, OGA and the myopic policy perform better; on the other hand, as the learning proceeds, the MQ-learning starts to make more precise decisions and, remarkably, in a couple of hundreds of iterations it is able to perform very close to the optimal policy.

Furthermore, to investigate the scalability of our proposed approach, Tables 4.1, 4.2, and 4.3

| $\%M/F$ | $1K$ | $2K$ | $3K$ | $4K$ | $6K$ | $8K$ | $10K$ |
|---|---|---|---|---|---|---|---|
| 10 % | 148 | 240 | 337 | 444 | 662 | 870 | 1089 |
| 20 % | 141 | 229 | 327 | 435 | 625 | 858 | 1052 |
| 40 % | 139 | 232 | 326 | 422 | 610 | 815 | 980 |
| 60 % | 149 | 251 | 372 | 497 | 699 | 949 | 1086 |

Table 4.1: Run-time of the proposed caching.

| $\%M/F$ | $1K$ | $2K$ | $3K$ | $4K$ | $6K$ | $8K$ | $10K$ |
|---|---|---|---|---|---|---|---|
| 10 % | 70 | 120 | 176 | 241 | 411 | 622 | 808 |
| 20 % | 73 | 123 | 183 | 250 | 389 | 569 | 796 |
| 40 % | 70 | 122 | 182 | 272 | 406 | 585 | 779 |
| 60 % | 92 | 170 | 254 | 356 | 551 | 736 | 908 |

Table 4.2: Run-time of OGA caching.

| $\%M/F$ | $1K$ | $2K$ | $3K$ | $4K$ | $6K$ | $8K$ | $10K$ |
|---|---|---|---|---|---|---|---|
| 10 % | 70 | 126 | 205 | 286 | 491 | 721 | 969 |
| 20 % | 84 | 153 | 232 | 317 | 507 | 736 | 1025 |
| 40 % | 88 | 157 | 236 | 328 | 575 | 822 | 1105 |
| 60 % | 87 | 161 | 240 | 336 | 563 | 834 | 1141 |

Table 4.3: Run-time of Myopic caching.

report the run-time (in seconds[3]) versus the number of files $F$ as well as the storage capacity $M$, set as a ratio of the total aggregated file sizes. Although the proposed approach has slightly higher run-time due to the utilized dual-decomposition technique and the solution of the arising integer DP, all methods scale gracefully (linearly) as the number of files increases from 1K to 10K.

## 4.18 Conclusions

A generic setup where a caching unit makes sequential fetch-cache decisions based on dynamic prices and user requests was investigated. Critical constraints were identified, the aggregated cost across files and time instants was formed, and the optimal adaptive caching was then formulated

[3]We run these simulations in parallel with 4 pools of workers, utilizing a machine with Intel(R) Core(TM) i7-4770 CPU @ 3.4 GHz specifications.

as a stochastic optimization problem. Due to the effects of the current cache decisions on future costs, the problem was cast as a dynamic program. To address the inherent functional estimation problem that arises in this type of programs, while leveraging the underlying problem structure, several computationally efficient algorithms were developed, including off-line (batch) approaches, as well as online (stochastic) approaches based on Q-learning. The last part of the paper was devoted to dynamic pricing mechanisms that allowed handling constraints both in the storage capacity of the cache memory, as well as on the back-haul transmission link connecting the caching unit with the cloud.

**Algorithm 8** Modified $Q$-learning for online caching

**Initialize** $0 < \gamma,\ \beta_t < 1,\ \hat{\mu}_0, \zeta, \epsilon_t,\ M$

**Output** $\hat{\tilde{Q}}^{w^f, a^f}_{r^f, s^f}\ (t+1)$

Set $\hat{\tilde{Q}}^{w^f, a^f}_{r^f, s^f}(1) = 0$ for all factors

Set $s_0^f = 0$ and variables $\boldsymbol{\theta}_0^f = \{r_0^f, \rho_0^f, \lambda_0^f\}$ are revealed

For $t = 0, 1 \ldots$ For the current state $(r_t^f, s_t^f)$, choose $(\breve{w}_t^{f*}, \breve{a}_t^{f*})$

$$(\breve{w}_t^{f*}, \breve{a}_t^{f*}) = \begin{cases} \text{Solve (4.51)} & \text{w.p. } 1 - \epsilon_t \\ \text{random } (w, a) \in \mathcal{X}_t^f(r_t^f, s_t^f) & \text{w.p. } \epsilon_t \end{cases}$$

Update dual variable

$$\hat{\mu}_{t+1} = \left[ \hat{\mu}_t + \zeta \left( \sum_{f=1}^{F} \breve{a}_t^{f*} \sigma^f - M \right) \right]^+$$

Incur cost $\breve{c}_t^f := c_t^f(\breve{a}_t^{f*}, \breve{w}_t^{f*}; \rho_t^f, \lambda_t^f) + \hat{\mu}_t \breve{a}_t^{f*} \sigma^f$

Apply $\Pi_{\mathrm{C4}}(\cdot)$ to guarantee C4 (if required)

$$\Pi_{\mathrm{C4}} \left[ \left\{ (\breve{w}_t^{f*}, \breve{a}_t^{f*}) \right\}_f \right] \to \left\{ w_t^{f*}, a_t^{f*} \right\}_f$$

Update state $s_{t+1}^f = a_t^{f*}$

Request and cost parameters, $\boldsymbol{\theta}_{t+1}^f$, are revealed

Update all $\hat{\tilde{Q}}$ factors as

$$\begin{aligned} \hat{\tilde{Q}}^{w_t^{f*}, a_t^{f*}}_{r_t^f, s_t^f}(t+1) = (1 - \beta_t)\ & \hat{\tilde{Q}}^{w_t^{f*}, a_t^{f*}}_{r_t^f, s_t^f}(t) \\ + \beta_t \Bigg[ \breve{c}_t^f + \gamma & \min_{(w^f, a^f) \in \mathcal{X}_{t+1}^f} \hat{\tilde{Q}}^{w^f, a^f}_{r_{t+1}^f, s_{t+1}^f}(t) \Bigg] \end{aligned}$$

# Chapter 5

# Data-driven, Reinforced, and Robust Learning Approaches for a Smarter Power Grid

## 5.1 Introduction

Frequent and sizable voltage fluctuations caused by the growing deployment of electric vehicles, demand response programs, and renewable energy sources, challenge modern distribution grids. Electric utilities are currently experiencing major issues related to the unprecedented levels of load peaks as well as renewable penetration. For instance, a solar farm connected at the end of a long distribution feeder in a rural area can cause voltage excursions along the feeder, while the apparent power capability of a substation transformer is strained by frequent reverse power flows. Moreover, over-voltage happens during midday when photovoltaic (PV) generation peaks and load demand is relatively low; whereas voltage sags occur mostly overnight due to low PV generation even when load demand is high [31]. This motivates why voltage regulation, the task of maintaining bus voltage magnitudes within desirable ranges, is critical in modern distribution grids.

Early approaches to regulating the voltages at a residential level have mainly relied on utility-owned devices, including load-tap-changing transformers, voltage regulators, and capacitor banks, to name a few. They offer a convenient means of controlling reactive power, through which

the voltage profile at their terminal buses as well as at other buses can be regulated [92, p. 678]. Obtaining the optimal configuration for these devices entails solving mixed-integer programs, which are NP-hard in general. To optimize the tap positions, a semi-definite relaxation heuristic was used in [143, 13]. Control rules based on heuristics were developed in [178, 31]. However, these approaches can be computationally demanding, and do not guarantee optimal performance. A batch reinforcement learning (RL) scheme based on linear function approximation was lately advocated in [197]. [1]

Another characteristic inherent to utility-owned equipment is their limited life cycle, which prompts control on a daily or even monthly basis. Such configurations have been effective in traditional distribution grids without (or with low) renewable generation, and with slowly varying load. Yet, as distributed generation grows in residential networks nowadays [171], [76], rapid voltage fluctuations occur frequently. According to a recent landmark bill, California mandated $50\%$ of its electricity to be powered by renewable resources by $2025$ and $60\%$ by $2030$. The power generated by a solar panel can vary by 15% of its nameplate rating within one-minute intervals [183]. Voltage control would entail more frequent switching actions, and further installation of control devices.

Smart power inverters on the other hand, come with contemporary distributed generation units, such as PV panels, and wind turbines. Embedded with computing and communication units, these can be commanded to adjust reactive power output within seconds, and in a continuously-valued fashion. Indeed, engaging smart inverters in reactive power control has recently emerged as a promising solution [85]. Computing the optimal setpoints for inverters' reactive power output is an instance of the optimal power flow task, which is non-convex [53]. To deal with the renewable uncertainty as well as other communication issues (e.g., delay and packet loss), stochastic, online, decentralized, and localized reactive control schemes have been advocated [85, 221, 86, 183, 182, 105, 215].

RL refers to a collection of tools for solving Markovian decision processes (MDPs), especially when the underlying transition mechanism is unknown [174]. In settings involving high-dimensional, continuous action and/or state spaces however, it is well known that conventional RL approaches suffer from the so-called 'curse of dimensionality,' which limits their impact in practice [128]. Deep neural networks (DNNs) can address the curse of dimensionality in the high-dimensional and continuous state space by providing compact low-dimensional

[1] The results of this Chapter have been published in [203, 205, 206, 200, 201, 202, 203]

representations of high-dimensional inputs [63]. Wedding deep learning with RL (using a DNN to approximate the action-value function), deep (D) RL has offered artificial agents with human-level performance across diverse application domains [128, 156]. (D)RL algorithms have also shown great potential in several challenging power systems control and monitoring tasks [45, 51, 197, 211, 199, 112], and load control [39, 48]. A batch RL scheme using linear function approximation was developed for voltage regulation in distribution systems [197]. For voltage control of transmission networks, DRL was recently investigated to adjust generator voltage setpoints [45]. A shortcoming of the mentioned (D)RL voltage control schemes is their inability to cope with the curse of dimensionality in action space. Moreover, *joint control* of both utility-owned devices and emerging power inverters has not been fully investigated. In addition, the discrete variables describing the on-off operation of capacitors and slow timescale associated with changing capacitor statuses, compared with those of fast-responding inverters further challenges voltage regulation. As a consequence, current capacitor decisions have a long-standing influence on future inverter setpoints. The other way around, current inverter setpoints also affect future commitment of capacitors through the aggregate cost. Indeed, this two-way long-term interaction is difficult to model and cope with.

In this context, voltage control is dealt with in the present Chapter using shunt capacitors and smart inverters. Preliminary results were presented in [206]. A novel two-timescale solution combining first principles based on physical models and data-driven advances is put forth. On the slow timescale (e.g., hourly or daily basis), the optimal configuration (corresponding to the discrete on-off commitment) of capacitors is formulated as a Markov decision process, by carefully defining state, action, and cost according to the available control variables in the grid. The solution of this MDP is approached by means of a DRL algorithm. This framework leverages the merits of the so-termed *target network* and *experience replay*, which can remove the correlation among the sequence of observations, to make the DRL stable and tractable. On the other hand, the setpoints of the inverters' reactive power output, are computed by minimizing the instantaneous voltage deviation using the exact or approximate grid models on the fast timescale (e.g., every few seconds).

Compared with past works, our contributions can be summarized as follows.

**c1)** *Joint control of two types of assets.* A hybrid data- and physics-driven approach to managing both utility-owned equipment as well as smart inverters;

**c2)** *Slow-timescale learning.* Modeling demand and generation as Markovian processes, optimal capacitor settings are learned from data using DRL;

**c3)** *Fast-timescale optimization.* Using exact or approximate grid models, the optimal setpoints for inverters are found relying on the most recent slow-timescale solution; and,

**c4)** *Curse of dimensionality in action space.* Introducing hyper deep $Q$-network to handle the curse of dimensionality emerging due to large number of capacitors.

## 5.2 Voltage Control in Two Timescales

In this section, we describe the system model, and formulate the two-timescale voltage regulation problem.

### 5.2.1 System model

Consider a distribution grid of $N+1$ buses rooted at the substation bus indexed by $i=0$, whose buses are collected into $\mathcal{N}_0 := \{0\} \cup \mathcal{N}$, and lines into $\mathcal{L} := \{1, \ldots, N\}$. For all $i \in \mathcal{N}$ (i.e., without substation bus), let $v_i$ denote their squared voltage magnitude, and $p_i + jq_i$ their complex power injected. For brevity, collect all nodal quantities into column vectors $\boldsymbol{v}$, $\boldsymbol{p}$, $\boldsymbol{q}$. Active power injection is split into its generation $p_i^g$ and consumption $p_i^c$ as $p_i := p_i^g - p_i^c$; likewise, reactive power injection is $q_i := q_i^g - q_i^c$. In distribution grids, it holds that $p_i^g = p_i^c = q_i^c = 0$ and $q_i^g > 0$ if bus $i$ has a capacitor; while $p_i^g = q_i^g = 0$ if bus $i$ is a purely load bus; and $p_i^c \geq 0$, $q_i^c \geq 0$, $p_i^g \geq 0$ if bus $i$ is equipped with a DG. Let us stack generation and consumption components into vectors $\boldsymbol{p}^g$, $\boldsymbol{q}^g$, $\boldsymbol{p}^c$, and $\boldsymbol{q}^c$ accordingly. Predictions of active power consumption and solar generation $(\boldsymbol{p}^c, \boldsymbol{q}^c, \boldsymbol{p}^g)$ can be obtained through the hourly and real-time market (see e.g., [85]), or by running load demand (solar generation) prediction algorithms [214].

As mentioned earlier, there are two types of assets in modern distribution grids that can be engaged in reactive power control; that is, utility-owned equipment featuring discrete actions and limited lifespan, as well as smart inverters controllable within seconds and in a continuously-valued fashion. As the aggregate load varies in a relatively slow way, traditional devices have been sufficient for providing voltage support; while fast-responding solutions using inverters become indispensable with the increase of uncertain renewable penetration. In this context, the present work focuses on voltage regulation by capitalizing on the reactive control capabilities of

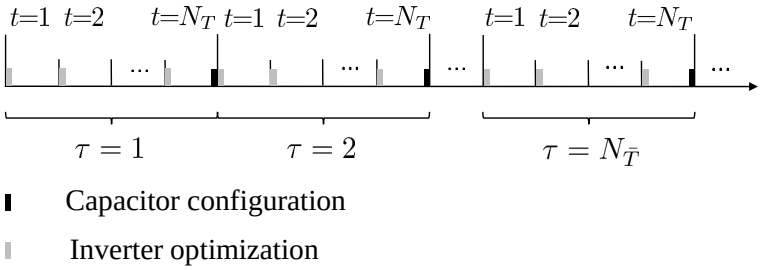



Figure 5.1: Two-timescale partitioning of a day for joint capacitor and inverter control.

both capacitors and inverters, while our framework can also account for other reactive power control devices. To this end, we divide every day into $N_{\bar{T}}$ intervals indexed by $\tau = 1, \ldots, N_{\bar{T}}$. Each of these $N_{\bar{T}}$ intervals is further partitioned into $N_T$ time slots which are indexed by $t = 1, \ldots, N_T$, as illustrated in Fig. 5.1. To match the slow load variations, the on-off decisions of capacitors are made (at the end of) every interval $\tau$, which can be chosen to be e.g., an hour; yet, to accommodate the rapidly changing renewable generation, the inverter output is adjusted (at the beginning of) every slot $t$, taken to be e.g., a minute. We assume that quantities $\boldsymbol{p}^g(\tau, t)$, $\boldsymbol{p}^c(\tau, t)$, and $\boldsymbol{q}^c(\tau, t)$ remain the same within each $t$-slot, but may change from slot $t$ to $t + 1$.

Suppose there are $N_a$ shunt capacitors installed in the grid, whose bus indices are collected in $\mathcal{N}_a$, and are in one-to-one correspondence with entries of $\mathcal{K} := \{1, \ldots, N_a\}$ (a simple renumbering). Assume that every bus is equipped with either a shunt capacitor or a smart inverter, but not both. The remaining buses, after removing entries in $\mathcal{N}_a$ from $\mathcal{N}$, collected in $\mathcal{N}_r$, are assumed equipped with inverters. This assumption is made without loss of generality as one can simply set the upper and lower bounds on the reactive output to zero at buses having no inverters installed.

As capacitor configuration is performed on a slow timescale (every $\tau$), the reactive compensation $q_i^g(\tau, t)$ provided by capacitor $k_i \in \mathcal{K}$ (i.e., capacitor at bus $i$) is represented by

$$q_i^g(\tau, t) = \hat{y}_{k_i}(\tau) q_{a,k_i}^g, \quad \forall i \in \mathcal{N}_a, \tau, t \tag{5.1}$$

where $\hat{y}_{k_i}(\tau) \in \{0, 1\}$ is the on-off commitment of capacitor $k_i$ for the entire interval $\tau$. Clearly, if $\hat{y}_{k_i}(\tau) = 1$, a constant amount (nameplate value) of reactive power $q_{a,k_i}^g$ is injected in the grid during this interval, and 0 otherwise. For convenience, the on-off decisions of capacitor units at interval $\tau$ are collected in a column vector $\hat{\boldsymbol{y}}(\tau)$.

On the other hand, the reactive power $q_{r,i}^g(\tau,t)$ generated by inverter $i$ is adjusted on the fast timescale (every $t$), and it is constrained by $|q_{r,i}^g(\tau,t)| \leq \sqrt{(\bar{s}_i)^2-(p_i^g(\tau,t))^2}$, where $\bar{s}_i$ is the power capability of inverter $i$. Traditionally, inverter $i$ is designed as $\bar{s}_i = \bar{p}_i^g$, where $\bar{p}_i^g$ is the active power capacity of the renewable generation unit installed at bus $i$. However, when maximum output is reached, i.e., $p_i^g(\tau,t) = \bar{p}_i^g$, no reactive power can be provided. To address this, oversized inverters' nameplate capacity has been advocated such that $\bar{s}_i > \bar{p}_i^g$ [85]. For instance, choosing $\bar{s}_i = 1.08\bar{p}_i^g$ and limiting $q_{r,i}^g(\tau,t)$ to $\sqrt{(\bar{s}_i)^2-(\bar{p}_i^g)^2}$ instead of $\sqrt{(\bar{s}_i)^2-(p_i^g(\tau,t))^2}$, the reactive power compensation provided by inverter $i$ is $|q_{r,i}^g(\tau,t)| \leq 0.4\bar{p}_i^g$, regardless of the instantaneous PV output $p_i^g(\tau,t)$ [85]. As such, $q_{r,i}^g(\tau,t)$ generated by inverter $i$ is constrained as

$$|q_{r,i}^g(\tau,t)| \leq \bar{q}_i^g := \sqrt{(\bar{s}_i)^2-(\bar{p}_i^g)^2}, \quad \forall i \in \mathcal{N}_r, t. \tag{5.2}$$

### 5.2.2 Two-timescale voltage regulation formulation

Given two-timescale load consumption and generation that we model as Markovian processes [30], the task of voltage regulation is to find the optimal reactive power support per slot by configuring capacitors in every interval and adjusting inverter outputs in every slot, such that the *long-term* average voltage deviation is minimized. As voltage magnitudes $\boldsymbol{v}(\tau,t)$ depend solely on the control variables $\boldsymbol{q}^g(\tau,t)$, they are expressed as implicit functions of $\boldsymbol{q}^g(\tau,t)$, yielding $\boldsymbol{v}_{\tau,t}(\boldsymbol{q}^g(\tau,t))$, whose actual function forms for postulated grid models will be given Section 5.3. The novel two-timescale voltage control scheme entails solving the following stochastic optimization problem

$$\underset{\substack{\{\boldsymbol{q}_\tau^g(\tau,t)\}\\ \{\boldsymbol{y}(\tau)\in\{0,1\}^{N_a}\}}}{\text{minimize}} \quad \mathbb{E}\left[\sum_{\tau=1}^{\infty}\sum_{t=1}^{N_T}\gamma^\tau \left\|\boldsymbol{v}_{\tau,t}(\boldsymbol{q}^g(\tau,t))-v_0\mathbf{1}\right\|^2\right] \tag{5.3a}$$

$$\text{subject to} \quad q_i^g(\tau,t) = \hat{y}_{k_i}(\tau)q_{a,k_i}^g, \qquad \forall i \in \mathcal{N}_a, \tau, t \tag{5.3b}$$

$$q_i^g(\tau,t) = q_{r,i}^g(\tau,t), \qquad \forall i \in \mathcal{N}_r, \tau, t \tag{5.3c}$$

$$|q_{r,i}^g(\tau,t)| \leq \bar{q}_i^g, \qquad \forall i \in \mathcal{N}_r, \tau, t \tag{5.3d}$$

for some discount factor $\gamma \in (0,1)$, where the expectation is taken over the joint distribution of $(\boldsymbol{p}^c(\tau,t), \boldsymbol{q}^c(\tau,t), \boldsymbol{p}^g(\tau,t))$ across all intervals and slots. Clearly, the optimization problem

(5.3) involves infinitely many variables $\{\boldsymbol{q}_r^g(\tau,t)\}$ and $\{\hat{\boldsymbol{y}}(\tau)\}$, which are coupled across time via the cost function and the constraint (5.3b). Moreover, discrete variables $\hat{\boldsymbol{y}}(\tau) \in \{0,1\}^{N_a}$ render problem (5.3) nonconvex and generally *NP-hard*. Last but not least, it is a multi-stage optimization, whose decisions are not all made at the same stage, and must also account for the power variability during real-time operation. In words, tackling (5.3) exactly is challenging.

Instead, our goal is to design algorithms that sequentially observe predictions $\{(\boldsymbol{p}^c(\tau,t),\boldsymbol{q}^c(\tau,t)),\boldsymbol{q}^g(\tau,t)\}$, and solve near optimally problem (5.3). The assumption is that, although no distributional knowledge of those stochastic processes involved is given, their realizations can be made available in real time, by means of e.g., accurate forecasting methods [214]. In this sense, the physics governing the electric power system will be utilized together with data to solve (5.3) in real time. Specifically, on the slow timescale, say at the end of each interval $\tau-1$, the optimal on-off capacitor decisions $\boldsymbol{y}(\tau)$ will be set through a DRL algorithm that can learn from the predictions collected within the current interval $\tau-1$; while, on the fast timescale, namely at the beginning of each slot $t$ within interval $\tau$, our two-stage control scheme will compute the optimal setpoints for inverters, by minimizing the instantaneous bus voltage deviations while respecting physical constraints, given the current on-off commitment of capacitor units $\hat{\boldsymbol{y}}(\tau)$ found at the very end of interval $(\tau-1)$. These two timescales are detailed in Sections 5.3 and 5.4, respectively.

## 5.3 Fast-timescale Optimization of Inverters

As alluded earlier, the actual forms of $\boldsymbol{v}_{\tau,t}(\boldsymbol{q}^g(\tau,t))$ will be specified in this section, relying on the exact AC model or a linearized approximant of it. Leveraging convex relaxation to deal with the nonconvexity, the considered AC model yields a second-order cone program (SOCP), whereas the linearized one leads to a linearly constrained quadratic program. In contrast, the latter offers an approximate yet computationally more affordable alternative to the former. Selecting between these two models relies on affordable computational capabilities.

### 5.3.1 Branch flow model

Due to the radial structure of distribution grids, every non-root bus $i \in \mathcal{N}$ has a unique parent bus termed $\pi_i$. The two are joined through the $i$-th distribution line represented by $(\pi_i, i) \in \mathcal{L}$ having impedance $r_i + jx_i$. Let $P_i(\tau,t) + jQ_i(\tau,t)$ stand for the complex power flowing from

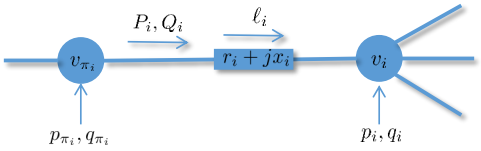



Figure 5.2: Bus $i$ is connected to its unique parent $\pi_i$ via line $i$.

buses $\pi_i$ to $i$ seen at the 'front' end at time slot $t$ of interval $\tau$, as depicted in Fig. 5.2. Throughout this section, the interval index $\tau$ will be dropped when it is clear from the context.

With further $\ell_i$ denoting the squared current magnitude on line $i \in \mathcal{L}$, the celebrated *branch flow model* is described by the following equations for all buses $i \in \mathcal{N}$, and for all $t$ within every interval $\tau$ [7, 109]

$$p_i(t) = \sum_{j \in \chi_i} P_j(t) - (P_i(t) - r_i \ell_i(t)) \tag{5.4a}$$

$$q_i(t) = \sum_{j \in \chi_i} Q_j(t) - (Q_i(t) - x_i \ell_i(t)) \tag{5.4b}$$

$$v_i(t) = v_{\pi_i}(t) - 2(r_i P_i(t) + x_i Q_i(t)) + (r_i^2 + x_i^2)\ell_i(t) \tag{5.4c}$$

$$\ell_i(t) = \frac{P_i^2(t) + Q_i^2(t)}{v_{\pi_i}(t)} \tag{5.4d}$$

where we have ignored the dependence on $\tau$ for brevity, and $\chi_i$ denotes the set of all children buses for bus $i$.

Clearly, the set of equations in (5.4d) is quadratic in $P_i(t)$ and $Q_i(t)$, yielding a nonconvex set. To address this challenge, consider relaxing the equalities (5.4d) into inequalities (a.k.a. hyperbolic relaxation, see e.g., [53])

$$P_i^2(t) + Q_i^2(t) \leq v_{\pi_i}(t)\ell_i(t), \quad \forall i \in \mathcal{N}, t \tag{5.5}$$

which can be equivalently rewritten as the following second-order cone constraints

$$\left\| \begin{array}{c} 2P_i(t) \\ 2Q_i(t) \\ \ell_i(t) - v_{\pi_i}(t) \end{array} \right\| \leq v_{\pi_i}(t) + \ell_i(t), \quad \forall i \in \mathcal{N}. \tag{5.6}$$

Equations (5.4a)-(5.4c) and (5.6) now define a convex feasible set. The procedure of leveraging

this relaxed set (instead of the nonconvex one) is known as SOCP relaxation [109]. Interestingly, it has been shown that under certain conditions, SOCP relaxation is exact in the sense that the set of inequalties (5.6) holds with equalities at the optimum [57].

Given the capacitor configuration $\hat{\boldsymbol{y}}(\tau)$ found at the end of the last interval $\tau-1$, under the aforementioned relaxed grid model, the voltage regulation on the fast timescale based on the exact AC model can be described as follows

$$\begin{aligned}
\underset{\boldsymbol{v}(t),\boldsymbol{q}_r^g(t),\boldsymbol{P}(t),\boldsymbol{Q}(t)}{\text{minimize}} \quad & \|\boldsymbol{v}(t)-v_0\mathbf{1}\|^2 && \text{(5.7a)}\\
\text{subject to} \quad & \text{(5.4a)}-\text{(5.4d)} &&\\
& q_i^g(t)=\hat{y}_{k_i}(\tau)q_{a,k_i}^g, \quad \forall i\in\mathcal{N}_a && \text{(5.7b)}\\
& q_i^g(t)=q_{r,i}^g(t), \qquad \forall i\in\mathcal{N}_r && \text{(5.7c)}\\
& |q_{r,i}^g(t)|\le \bar{q}_i^g, \qquad \forall i\in\mathcal{N}_r && \text{(5.7d)}
\end{aligned}$$

which is readily a convex SOCP and can be efficiently solved by off-the-shelf convex programming toolboxes. The optimal setpoints of smart inverters for the exact AC model are found as the $\boldsymbol{q}_r^g$-minimizer of (5.7).

However, solving SOCPs could be computationally demanding when dealing with relatively large-scale distribution grids, say of several hundred buses. Trading off modeling accuracy for computational efficiency, our next instantiation of the fast-timescale voltage control relies on an approximate grid model.

### 5.3.2 Linearized power flow model

As line current magnitudes $\{\ell_i\}$ are relatively small compared to line flows, the last term in (5.4a)-(5.4c) can be ignored yielding the next set of linear equations for all $i,t$ [8]

$$p_i(t)=\sum_{j\in\chi_i}P_j(t)-P_i(t) \tag{5.8a}$$

$$q_i(t)=\sum_{j\in\chi_i}Q_j(t)-Q_i(t) \tag{5.8b}$$

$$v_i(t)=v_{\pi_i}(t)-2(r_iP_i(t)+x_iQ_i(t)) \tag{5.8c}$$

which is known as the linearized distribution flow model. In this fashion, all squared voltage magnitudes $\boldsymbol{v}(t)$ can be expressed as linear functions of $\boldsymbol{q}^g(t)$.

Adopting the approximate model (5.8), the optimal setpoints of inverters can be found by solving the following optimization problem per slot $t$ in interval $\tau$, provided $\hat{\boldsymbol{y}}(\tau)$ is available from the last interval on the slow timescale

$$\underset{\boldsymbol{v}(t),\boldsymbol{q}_r^g(t),\boldsymbol{P}(t),\boldsymbol{Q}(t)}{\text{minimize}} \quad \|\boldsymbol{v}(t) - v_0\mathbf{1}\|^2 \tag{5.9a}$$

$$\text{subject to} \quad \text{(5.8a)} - \text{(5.8c)}$$

$$q_i^g(t) = \hat{y}_{k_i}(\tau) q_{a,k_i}^g, \quad \forall i \in \mathcal{N}_a \tag{5.9b}$$

$$q_i^g(t) = q_{r,i}^g(t), \qquad \forall i \in \mathcal{N}_r \tag{5.9c}$$

$$|q_{r,i}^g(t)| \leq \bar{q}_i^g, \qquad \forall i \in \mathcal{N}_r. \tag{5.9d}$$

As all constraints are linear and the cost is quadratic, (5.9) constitutes a standard convex quadratic program. As such, it can be solved efficiently by e.g., primal-dual algorithms, or off-the-shelf convex programming solvers, whose implementation details are skipped due to space limitations.

## 5.4 Slow-timescale Capacitor Reconfiguration

Here we deal with reconfiguration of shunt capacitors on the slow timescale. This amounts to determining their on-off status for the ensuing interval. Past approaches to solving the resultant integer-valued optimization were heuristic, or, relied on semidefinite programming relaxation. They do not guarantee optimality, while they also incur high computational and storage complexities. We take a different route by drawing from advances in artificial intelligence, to develop data-driven solutions that could near optimally learn, track, as well as adapt to unknown generation and consumption dynamics.

### 5.4.1 A data-driven solution

Clearly from (5.7b)–(5.9b), the capacitor decisions $\hat{\boldsymbol{y}}(\tau)$ made at the end of interval $\tau - 1$ (slow-timescale learning) influence inverters' setpoints during the entire interval $\tau$ (fast-timescale optimization). The other way around, inverters' regulation on voltages influences the capacitor

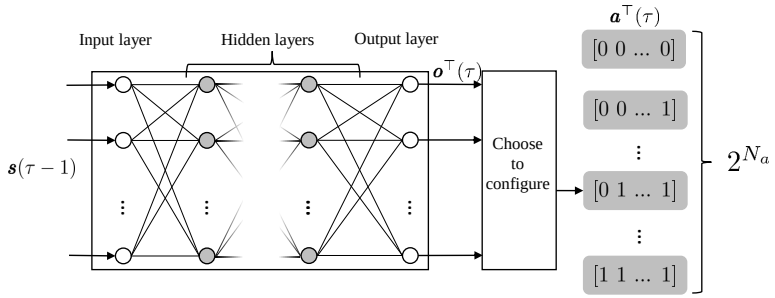



Figure 5.3: Deep $Q$-network

commitment for the next interval. This two-way between the capacitor configuration and the optimal setpoints of inverters motivates our RL formulation. Dealing with learning policy functions in an environment with action-dependent dynamically evolving states and costs, RL seeks a policy function (of states) to draw actions from, in order to minimize the average cumulative cost [174].

Modeling load demand and renewable generation as Markovian processes, the optimal configuration of capacitors can be formulated as an MDP, which can be efficiently solved through RL algorithms. An MDP is defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma)$, where $\mathcal{S}$ is a set of states; $\mathcal{A}$ is a set of actions; $\mathcal{P}$ is a set of transition matrices; $c : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a cost function such that, for $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}$, $c = (c(\boldsymbol{s}, \boldsymbol{a}))_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}}$ are the real-valued instantaneous costs after the system operator takes an action $\boldsymbol{a}$ at state $\boldsymbol{s}$; and $\gamma \in [0, 1)$ is the discount factor. These components are defined next before introducing our voltage regulation scheme.

*Action space* $\mathcal{A}$. Each action corresponds to one possible on-off commitment of capacitors 1 to $N_a$, giving rise to an action vector $\boldsymbol{a}(\tau) = \boldsymbol{y}(\tau)$ per interval $\tau$. The set of binary action vectors constitutes the action space $\mathcal{A}$, whose cardinality is exponential in the number of capacitors, meaning $|\mathcal{A}| = 2^{N_a}$.

*State space* $\mathcal{S}$. This includes per interval $\tau$ the average active power at all buses except for the substation, along with the current capacitor configurations; that is, $\boldsymbol{s}(\tau) := [\bar{\boldsymbol{p}}^\top(\tau), \hat{\boldsymbol{y}}^\top(\tau)]^\top$, which contains both continuous and discrete variables. Clearly, it holds that $\mathcal{S} \subseteq \mathbb{R}^N \times 2^{N_a}$.

The action is decided according to the configuration policy $\pi$ that is a function of the most recent state $\boldsymbol{s}(\tau - 1)$, given as

$$\boldsymbol{a}(\tau) = \pi(\boldsymbol{s}(\tau - 1)). \tag{5.10}$$

*Cost function c.* The cost on the slow timescale is

$$c(\boldsymbol{s}(\tau-1),\boldsymbol{a}(\tau))=\sum_{t=1}^{N_T}\|\boldsymbol{v}_{\tau,t}(\boldsymbol{q}^g(\tau,t))-v_0\mathbf{1}\|^2. \tag{5.11}$$

*Set of transition probability matrices $\mathcal{P}$.* While being at a state $\boldsymbol{s}\in\mathcal{S}$ upon taking an action $\boldsymbol{a}$, the system moves to a new state $\boldsymbol{s}'\in\mathcal{S}$ probabilistically. Let $P^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}$ denote the transition probability matrix from state $\boldsymbol{s}$ to the next state $\boldsymbol{s}'$ under a given action $\boldsymbol{a}$. Evidently, it holds that $\mathcal{P}:=\{P^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}|\forall\boldsymbol{a}\in\mathcal{A}\}$.

*Discount factor $\gamma$.* The discount factor $\gamma\in[0,1)$, trades off the current versus future costs. The smaller $\gamma$ is, the more weight the current cost has in the overall cost.

Given the current state and action, the so-termed action-value function under the control policy $\pi$ is defined as

$$\begin{aligned}&Q_\pi(\boldsymbol{s}(\tau-1),\boldsymbol{a}(\tau)):=\\&\mathbb{E}\left[\sum_{\tau'=\tau}^{\infty}\gamma^{\tau'-\tau}c(\boldsymbol{s}(\tau'-1),\boldsymbol{a}(\tau'))\Big|\pi,\boldsymbol{s}(\tau-1),\boldsymbol{a}(\tau)\right]\end{aligned} \tag{5.12}$$

where the expectation $\mathbb{E}$ is taken with respect to all sources of randomness.

To find the optimal capacitor configuration policy $\pi^*$, that minimizes the average voltage deviation in the long run, we resort to the Bellman optimality equations; see e.g., [174]. Solving those yields the action-value function under the optimal policy $\pi^*$ on the fly, given by

$$Q_{\pi^*}(\boldsymbol{s},\boldsymbol{a})=\mathbb{E}[c(\boldsymbol{s},\boldsymbol{a})]+\gamma\sum_{\boldsymbol{s}'\in\mathcal{S}}P^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\min_{\boldsymbol{a}\in\mathcal{A}}Q_{\pi^*}(\boldsymbol{s}',\boldsymbol{a}'). \tag{5.13}$$

With $Q_{\pi^*}(\boldsymbol{s},\boldsymbol{a})$ obtained, the optimal capacitor configuration policy can be found as

$$\pi^*(\boldsymbol{s})=\arg\min_{\boldsymbol{a}}\ Q_{\pi^*}(\boldsymbol{s},\boldsymbol{a}). \tag{5.14}$$

It is clear from (5.13) that if all transition probabilities $\{P^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\}$ were available, we can derive $Q_{\pi^*}(\boldsymbol{s},\boldsymbol{a})$, and subsequently the optimal policy $\pi^*$ from (5.14). Nonetheless, obtaining those transition probabilities is impractical in practical distribution systems. This calls for approaches that aim directly at $\pi^*$, without assuming any knowledge of $\{P^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\}$.

One celebrated approach of this kind is Q-learning, which can learn $\pi^*$ by approximating

$Q_{\pi^*}(\boldsymbol{s}, \boldsymbol{a})$ 'on-the-fly' [174, p. 107]. Due to its high-dimensional continuous state space $\mathcal{S}$ however, $Q$-learning is not applicable for the problem at hand. This motivates function approximation based $Q$-learning schemes that can deal with continuous state domains.

### 5.4.2 A deep reinforcement learning approach

DQN offers a NN function approximator of the $Q$-function, chosen to be e.g., a fully connected feed-forward NN, or a convolutional NN, depending on the application [128]. It takes as input the state vector, to generate at its output $Q$-values for all possible actions (one for each). As demonstrated in [128], such a NN indeed enables learning the $Q$-values of *all* state-action pairs, from just a few observations obtained by interacting with the environment. Hence, it effectively addresses the challenge brought by the 'curse of dimensionality' [128]. Inspired by this, we employ a feed-forward NN to approximate the $Q$-function in our setting. Specifically, our DNN consists of $L$ fully connected hidden layers with ReLU activation functions, depicted in Fig. 5.3. At the input layer, each neuron is fed with one entry of the state vector $\boldsymbol{s}(\tau - 1)$, which, after passing through $L$ ReLU layers, outputs a vector $\boldsymbol{o}(\tau) \in \mathbb{R}^{2^{N_a}}$, whose elements predict the $Q$-values for all possible actions (i.e., capacitor configurations). Since each output unit corresponds to a particular configuration of all $N_a$ capacitors, there is a total of $2^{N_a}$ neurons at the output layer. For ease of exposition, let us collect all weight parameters of this DQN into a vector $\boldsymbol{\theta}$ which parameterizes the input-output relationship as $\boldsymbol{o}(\tau) = Q_\pi(\boldsymbol{s}(\tau - 1), \boldsymbol{a}(\tau); \boldsymbol{\theta})$ (c.f. (5.12)). At the end of a given interval $\tau - 1$, upon passing the state vector $\boldsymbol{s}(\tau - 1)$ through this DQN, the corresponding predicted $Q$-values $\boldsymbol{o}(\tau)$ for all possible actions become available at the output. Based on these predicted values, the system operator selects the action having the smallest predicted $Q$-value to be in effect over the next interval.

Intuitively, the weights $\boldsymbol{\theta}$ should be chosen such that the DQN outputs match well the actual $Q$-values with input any state vector. Toward this objective, the popular stochastic gradient descent (SGD) method is employed to update $\boldsymbol{\theta}$ 'on the fly' [128]. At the end of a given interval $\tau$, precisely when i) the system operator has made decision $\boldsymbol{a}(\tau)$, ii) the grid has completed the transition from the state $\boldsymbol{s}(\tau - 1)$ to a new state $\boldsymbol{s}(\tau)$, and, (iii) the network has incurred and revealed cost $c(\boldsymbol{s}(\tau - 1), \boldsymbol{a}(\tau))$, we perform a SGD update based on the current estimate $\boldsymbol{\theta}_\tau$ to yield $\boldsymbol{\theta}_{\tau+1}$. The so-termed temporal-difference learning [174] confirms that a sample approximation of the optimal cost-to-go from interval $\tau$ is given by $c(\boldsymbol{s}(\tau -$

**Algorithm 9** Two-timescale voltage regulation scheme.

**Initialize** $\boldsymbol{\theta}_0$ randomly; weight of the target network $\boldsymbol{\theta}_0^{\text{Tar}} = \boldsymbol{\theta}_0$; replay buffer $\mathcal{R}$; and the initial state $\boldsymbol{s}(0)$.
For $\tau = 1, 2, ...$
Take action $\boldsymbol{a}(\tau)$ through exploration-exploitation

$$\boldsymbol{a}(\tau) = \begin{cases} \text{random} \ \ \boldsymbol{a} \in \mathcal{A} & \text{w.p. } \epsilon_\tau \\ \arg\min_{\boldsymbol{a}'} Q(\boldsymbol{s}(\tau-1), \boldsymbol{a}'; \boldsymbol{\theta}_\tau) & \text{w.p. } 1-\epsilon_\tau \end{cases}$$

where $\epsilon_\tau = \max\{1 - 0.1 \times \lfloor \tau/50 \rfloor, \, 0\}$.

Evaluate $\boldsymbol{c}(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau))$ using (5.11).
For $t = 1, 2, ..., N_T$
Compute $\boldsymbol{q}^g(\tau, t)$ using (5.7) or (5.9).
Update $\boldsymbol{s}(\tau)$.
Save $(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau), c(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau))$ into $\mathcal{R}(\tau)$.
Randomly sample $M_\tau$ experiences from $\mathcal{R}(\tau)$.
Form the mini-batch loss $\mathcal{L}^{\text{Tar}}(\boldsymbol{\theta}_\tau; \mathcal{M}_\tau)$ using (5.18).
Update $\boldsymbol{\theta}_{\tau+1}$ using (5.19).
If $\text{mod}(\tau, B) = 0$
Update the target network $\boldsymbol{\theta}_\tau^{\text{Tar}} = \boldsymbol{\theta}_\tau$.

$1), \boldsymbol{a}(\tau)) + \gamma \min_{\boldsymbol{a}' \in \mathcal{A}} Q_\pi(\boldsymbol{s}(\tau), \boldsymbol{a}'; \boldsymbol{\theta}_\tau)$, where $c(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau))$ is the instantaneous cost observed, and $\min_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s}(\tau), \boldsymbol{a}'; \boldsymbol{\theta}_\tau)$ represents the smallest possible predicted cost-to-go from state $\boldsymbol{s}(\tau)$, which can be computed through our DQN with weights $\boldsymbol{\theta}_\tau$, and is discounted by factor $\gamma$. In words, the target value $c(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau)) + \gamma \min_{\boldsymbol{a}' \in \mathcal{A}} Q_\pi(\boldsymbol{s}(\tau), \boldsymbol{a}'; \boldsymbol{\theta}_\tau)$ is readily available at the end of interval $\tau - 1$. Adopting the $\ell_2$-norm error criterion, a meaningful approach to tuning the weights $\boldsymbol{\theta}$ entails minimizing the following loss function

$$\mathcal{L}(\boldsymbol{\theta}) := \Big[ c(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau)) + \gamma \min_{\boldsymbol{a}' \in \mathcal{A}} Q_\pi(\boldsymbol{s}(\tau), \boldsymbol{a}'; \boldsymbol{\theta}_\tau) - Q_\pi(\boldsymbol{s}(\tau-1), \boldsymbol{a}(\tau); \boldsymbol{\theta}) \Big]^2$$

for which the SGD update is given by

$$\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_\tau - \beta_\tau \nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_\tau} \tag{5.15}$$

where $\beta_\tau > 0$ is a preselected learning rate, and $\nabla \mathcal{L}(\boldsymbol{\theta})$ denotes the (sub-)gradient.

However, due to the compositional structure of DNNs, the update (5.15) does not work well in practice. In fact, the resultant DQN oftentimes does not provide a stable result; see

e.g., [195]. To bypass these hurdles, several modifications have been introduced. In this work, we adopt the *target network* and *experience replay* [128]. To this aim, let us define an experience $e(\tau') := (\boldsymbol{s}(\tau'-1), \boldsymbol{a}(\tau')), c(\boldsymbol{s}(\tau'-1), \boldsymbol{a}(\tau')), \boldsymbol{s}(\tau'))$, to be a tuple of state, action, cost, and the next state. Consider also having a replay buffer $\mathcal{R}(\tau)$ on-the-fly, which stores the most recent $R > 0$ experiences visited by the agent. For instance, the replay buffer at any interval $\tau \geq R$ is $\mathcal{R}(\tau) := \{e(\tau - R + 1), \ldots, e(\tau)\}$. Furthermore, as another effective remedy to stabilizing the DQN updates, we replicate the DQN to create a second DNN, commonly referred to as the *target* network, whose weight parameters are concatenated in the vector $\boldsymbol{\theta}^{\mathrm{Tar}}$. It is worth highlighting that this target network is not trained, but its parameters $\boldsymbol{\theta}^{\mathrm{Tar}}$ are only periodically reset to estimates of $\boldsymbol{\theta}$, say every $B$ training iterations of the DQN. Consider now the temporal-difference loss for some randomly drawn experience $e(\tau')$ from $\mathcal{R}(\tau)$ at interval $\tau$

$$\mathcal{L}^{\mathrm{Tar}}(\boldsymbol{\theta}_\tau; e(\tau')) := \frac{1}{2}\Big[c(\boldsymbol{s}(\tau'-1), \boldsymbol{a}(\tau')) + \gamma \min_{\boldsymbol{a}'} Q^{\mathrm{Tar}}(\boldsymbol{s}(\tau), \boldsymbol{a}'; \boldsymbol{\theta}^{\mathrm{Tar}}_{\tau'}) - Q(\boldsymbol{s}(\tau'-1), \boldsymbol{a}(\tau'); \boldsymbol{\theta}_\tau)\Big]^2. \tag{5.16}$$

Upon taking expectation with respect to all sources of randomness generating this experience, we arrive at

$$\mathcal{L}^{\mathrm{Tar}}(\boldsymbol{\theta}_\tau; \mathcal{R}(\tau))) := \mathbb{E}_{e(\tau')}\, \mathcal{L}^{\mathrm{Tar}}(\boldsymbol{\theta}_\tau; e(\tau')). \tag{5.17}$$

In practice however, the underlying transition probabilities are unknown, which challenges evaluating and hence minimizing $\mathcal{L}^{\mathrm{Tar}}(\boldsymbol{\theta}_\tau; \mathcal{R}(\tau)))$ exactly. A commonly adopted alternative is to approximate the expected loss with an empirical loss over a few samples (that is, experiences here). To this end, we draw a mini-batch of $M_\tau$ experiences uniformly at random from the replay buffer $\mathcal{R}(\tau)$, whose indices are collected in the set $\mathcal{M}_\tau$, i.e., $\{e(\tau')\}_{\tau' \in \mathcal{M}_\tau} \sim U(\mathcal{R}(\tau))$. Upon computing for each of those sampled experiences an output using the target network with parameters $\boldsymbol{\theta}^{\mathrm{Tar}}_\tau$, the empirical loss is

$$\mathcal{L}^{\mathrm{Tar}}(\boldsymbol{\theta}_\tau; \mathcal{M}_\tau) := \frac{1}{2M_\tau} \sum_{\tau' \in \mathcal{M}_\tau} \Big[c(\boldsymbol{s}(\tau'-1), \boldsymbol{a}(\tau')) + \gamma \min_{\boldsymbol{a}'} Q^{\mathrm{Tar}}(\boldsymbol{s}(\tau'), \boldsymbol{a}'; \boldsymbol{\theta}^{\mathrm{Tar}}_\tau) - Q(\boldsymbol{s}(\tau'-1), \boldsymbol{a}(\tau'); \boldsymbol{\theta}_\tau)\Big]^2. \tag{5.18}$$

In a nutshell, the weight parameter vector $\boldsymbol{\theta}_\tau$ of the DQN is efficiently updated 'on-the-fly'

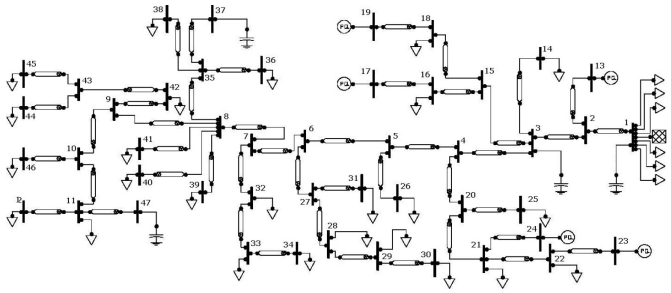


Figure 5.4: Schematic diagram of the 47-bus industrial distribution feeder. Bus 1 is the substation, and the 6 loads connected to it model other feeders on this substation.

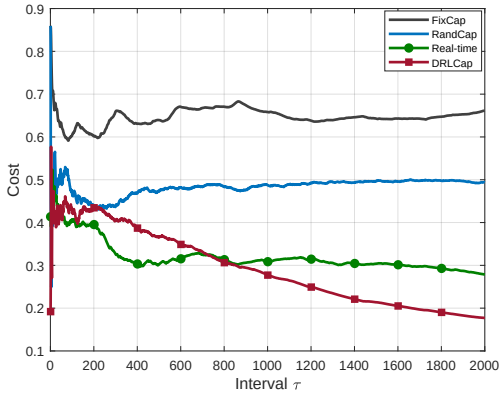



Figure 5.5: Time-averaged instantaneous costs incurred by the four voltage control schemes.

using SGD over the empirical loss $\mathcal{L}^{\text{Tar}}(\boldsymbol{\theta}_\tau; \mathcal{M}_\tau)$, with iterates given by

$$\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_\tau - \beta_\tau \nabla \mathcal{L}^{\text{Tar}}(\boldsymbol{\theta}_\tau; \mathcal{M}_\tau). \tag{5.19}$$

Incorporating *target network* and *experience replay* remedies for stable DRL, our proposed two-timescale voltage regulation scheme is summarized in Alg. 10.

## 5.5 Numerical Tests

In this section, numerical tests on a real-world 47-bus distribution feeder as well as the IEEE 123-bus benchmark system are provided to showcase the performance of our proposed DRL-based voltage control scheme (cf. presented in Alg. 10). As has already been shown in previous

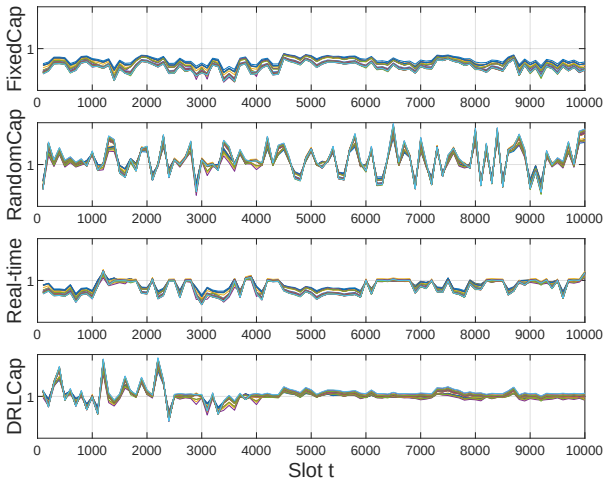



Figure 5.6: Voltage magnitude profiles obtained by the four voltage control schemes over the simulation period of $10,000$ slots.

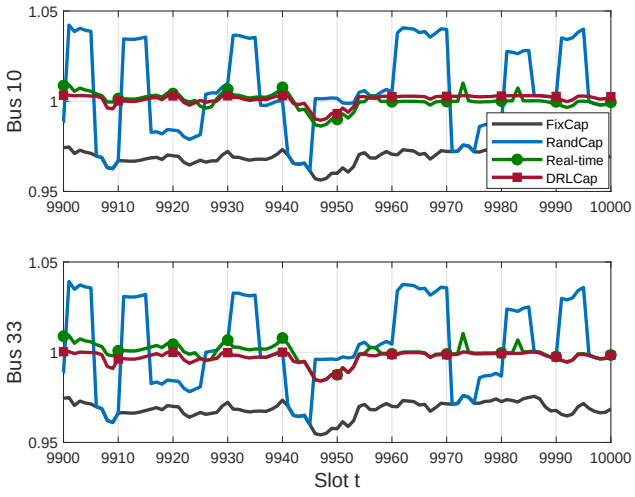



Figure 5.7: Voltage magnitude profiles obtained by the four voltage control schemes at buses $10$ and $33$ from slot $9,900$ to $10,000$.

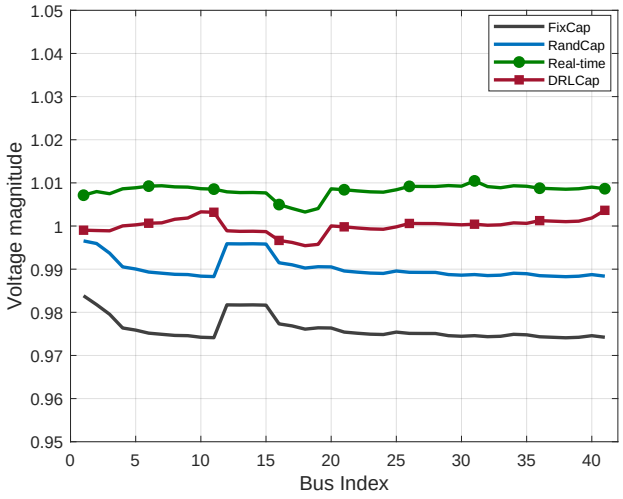



Figure 5.8: Voltage magnitude profiles at all buses at slot $9,900$ obtained by the four voltage control schemes.

works (e.g., [85, 86, 109]), the linearized distribution flow model approximates the exact AC model very well; hence, numerical results based on the linearized model were only reported here.

The first experiment entails the Southern California Edison $47$-bus distribution feeder [53], which is depicted in Fig. 5.4. This feeder is integrated with four shunt capacitors as well as five smart inverters. As the voltage magnitude $v_0$ of the substation bus is regulated to be a constant ($1$ in all our tests) through a voltage transformer, the capacitor at the substation was excluded from our control. Thus, a total of three shunt capacitors along with five smart inverters embedded with large PV plants were engaged in voltage regulation. The rest three capacitors are installed on buses $3$, $37$, and $47$, with capacities $120$, $180$, and $180$ kVar, respectively, while the five large PV plants are located on buses $2$, $16$, $18$, $21$, and $22$, with capacities $300$, $80$, $300$, $400$, and $200$ kW, respectively. To test our scheme in a realistic setting, real consumption as well as solar generation data were obtained from the Smart* project collected on August $24, 2011$ [11], which were first preprocessed by following the procedure described in our precursor work [85].

In our tests, to match the availability of real data, each slot $t$ was set to a minute, and each interval $\tau$ was set to five minutes. A power factor of $0.8$ was assumed for all loads. The DQN used here consists of three fully connected layers, which has $44$ and $12$ units in the first and second hidden layers, respectively. Although simple, it was found sufficient for the task at

hand. ReLU activation functions ($\sigma(x) = \max(x, 0)$) were employed in the hidden layers, and logistic sigmoid functions $s(x) = 1/(1 + e^{-x})$ were used at the output layer. To assess the performance of our proposed scheme, we have simulated three capacitor configuration policies as baselines, that include a fixed capacitor configuration (FixCap), a random capacitor configuration (RandCap), and an (impractical) 'real-time' policy. Specifically, the FixCap uses a fixed capacitor configuration throughout, and the RandCap implements random actions to configure the capacitors on every slow time interval; both of which compute the inverter setpoints by solving (5.9) per slot $t$. The impractical Real-time scheme however, optimizes over inverters and capacitors on a single-timescale, namely at every slot – hence justifying its 'real-time' characterization. To carry out this optimization task, first the binary constraints $y_{k_i}(t) \in \{0, 1\}$ are relaxed to box ones $y_{k_i}(t) \in [0, 1]$, the resulting convex program is solved using an off-the-shelf routine [66], which is followed by a standard rounding step to recover binary solutions for capacitor configurations [9].

In the first experiment, the DRL-based capacitor configuration (DRLCap) voltage control approach was examined. The replay buffer size was set to $R = 10$, the discount factor $\gamma = 0.99$, the mini-batch size $M_\tau = 10$, and the exploration-exploitation parameter $\epsilon_\tau = \max\{1 - 0.1 \times \lfloor \tau/50 \rfloor, 0\}$. During training, the target network was updated every $B = 5$ iterations. The time-averaged instantaneous costs

$$\frac{1}{\tau} \sum_{i=1}^{\tau} c(\boldsymbol{s}(i-1), \boldsymbol{a}(i))$$

incurred by the four schemes over the first $1 \leq \tau \leq 2,000$ intervals are plotted in Fig. 5.5. Evidently, the proposed scheme attains a lower cost than FixCap, RandCap, and Real-time after a short period of learning and interacting with the environment. Even though the real-time scheme optimizes both capacitor configurations and inverter setpoints per slot $t$, its suboptimal performance in this case arises from the gap between the convexified problem and the original nonconvex counterpart. Fig. 5.6 presents the voltage magnitude profiles for all buses regulated by the four schemes sampled at every $100$ slots. Again, after a short period ($\sim 4,500$ slots) of training through interacting with the environment, our DRLCap voltage control scheme quickly learns a stable and (near-) optimal policy. In addition, voltage magnitude profiles regulated by FixCap, RandCap, Real-time, and DRLCap at buses $10$ and $33$ from slot $9,900$ to $10,000$ are shown in Fig. 5.7, while the voltage magnitude profiles at all buses at slot $9,900$ are presented

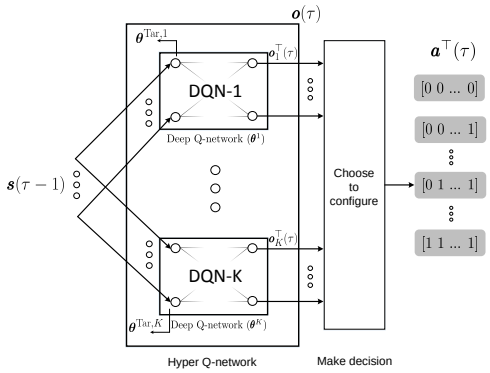



Figure 5.9: Hyper deep $Q$-network for capacitor configuration.

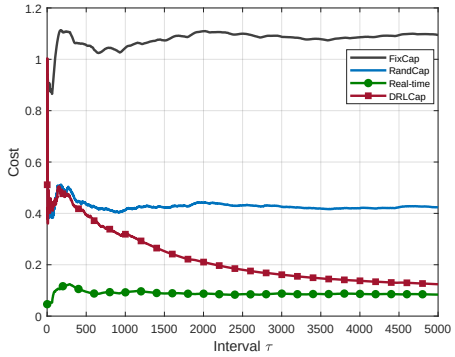



Figure 5.10: Time-averaged instantaneous costs incurred by the four approaches on the IEEE 123-bus feeder.

in Fig. 5.8. Curves showcase the effectiveness of our DRLCap scheme in smoothing voltage fluctuations incurred due to large solar generation as well as heavy load demand.

To deal with distribution systems having a moderately large number of capacitors, we further advocate a hyper deep $Q$-network implementation, that endows our DRL-based scheme with scalability. The idea here is to first split the total number $2^{N_a}$ of $Q$-value predictions $\boldsymbol{o}(\tau) \in \mathbb{R}^{2^{N_a}}$ at the output layer into $K$ smaller groups, each of which is of the same size $2^{N_a}/K$ and is to be predicted by a small-size DQN. This evidently yields the representation $\boldsymbol{o}(\tau) := [\boldsymbol{o}_1^\top(\tau), \ldots, \boldsymbol{o}_K^\top(\tau)]^\top$, where $\boldsymbol{o}_k(\tau) \in \mathbb{R}^{2^{N_a}/K}$for $k = 1, \ldots, K$. By running $K$ DQNs in parallel along with their corresponding target networks, each DQN-$k$ generates predicted $Q$-values $\boldsymbol{o}_k(\tau)$ for the subset of actions corresponding to $k$th group. Note that all DQNs are fed

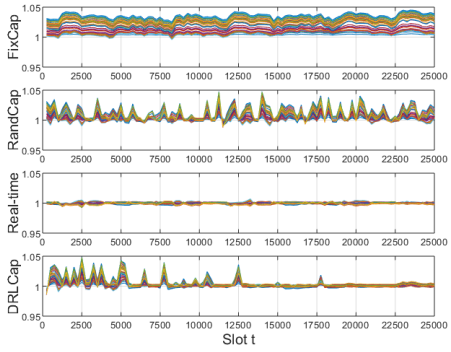



Figure 5.11: Voltage magnitude profiles at all buses over the simulation period of $25,000$ slots on the IEEE 123-bus feeder.

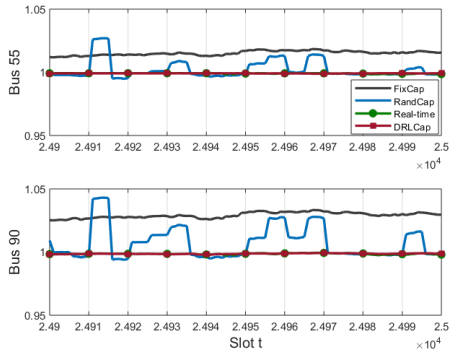



Figure 5.12: Voltage magnitude profiles at buses $55$ and $90$ from slot $24,900$ to $25,000$ obtained by the four approaches on the IEEE 123-bus feeder.

with the same state vector $\boldsymbol{s}(\tau - 1)$; see also Fig. 5.9 for an illustration.

To examine the scalability and performance of this hyper $Q$-network implementation, additional tests using the IEEE 123-bus test feeder with 9 shunt capacitors were performed. Again, the capacitor at bus 1 was excluded from the control, rendering a total number of $2^8 = 256$ actions (capacitor configurations). Renewable (PV) units are located on buses $47$, $49$, $63$, $73$, $104$, $108$, $113$, with capacities $100$, $16$, $70$, $20$, $20$, $30$, and $10$ k, respectively. The $8$ shunt capacitors are installed on buses $3$, $20$, $44$, $93$, $96$, $98$, $100$, and $114$, with capacities $50$, $80$, $100$, $100$, $100$, $100$, $100$, and $60$ kVar. In this experiment, we used a total of $K = 64$ equal-sized DQNs to form the hyper $Q$-network, where each DQN implemented a fully connected 3-layer feed-forward neural network, with ReLU activation functions in the hidden layers, and sigmoid functions at

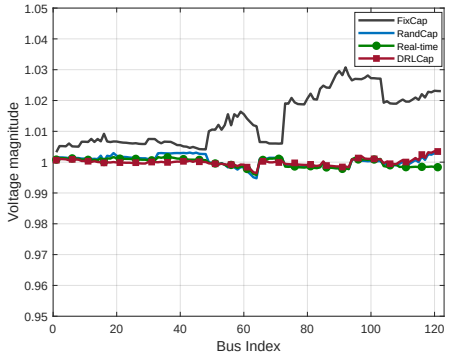



Figure 5.13: Voltage magnitude profiles at all buses on slot $24,900$ obtained by four approaches on the IEEE 123-bus feeder.

the output. The replay buffer size was set to $R = 50$, the batch size to $M_\tau = 8$, and the target network updating period to $B = 10$. The time-averaged instantaneous costs obtained over a simulation period of $5,000$ intervals is plotted in Fig. 5.10. Moreover, voltage magnitude profiles of all buses over the simulation period of $25,000$ slots sampled at every $100$ slots under the four schemes are plotted in Fig. 5.11; voltage magnitude profiles at buses $55$ and $90$ from slot $24,900$ to $25,000$ are shown in Fig. 5.12; and, voltage magnitude profiles at all buses on slot $24,900$ are depicted in 5.13. Evidently, the hyper deep $Q$-network based DRL scheme smooths out the voltage fluctuations after a certain period ($\sim 7,000$ slots) of learning, while effectively handling the curse of dimensionality in the control (action) space. Evidently from Figs. 5.10 and 5.13, both the time-averaged immediate cost as well as the voltage profiles of DRLCap converge to those of the impractical 'real-time' scheme (which jointly optimizes inverter setpoints and capacitor configurations per slot).

## 5.6 Conclusions

In this section, joint control of traditional utility-owned equipment and contemporary smart inverters for voltage regulation through reactive power provision was investigated. To account for the different response times of those assets, a two-timescale approach to minimizing bus voltage deviations from their nominal values was put forth, by combining physics- and data-driven stochastic optimization. Load consumption and active power generation dynamics were modeled as MDPs. On a fast timescale, the setpoints of smart inverters were found by minimizing the

instantaneous bus voltage deviations, while on a slower timescale, the capacitor banks were configured to minimize the long-term expected voltage deviations using a deep reinforcement learning algorithm. The developed two-timescale voltage regulation scheme was found efficient and easy to implement in practice, through extensive numerical tests on real-world distribution systems using real solar and consumption data. This work also opens up several interesting directions for future research, including deep reinforcement learning for real-time optimal power flow as well as unit commitment.

## 5.7 Gauss-Newton Unrolled Neural Networks and Data-driven Priors for Regularized PSSE with Robustness

## 5.8 Introduction

In today's smart grid, reliability and accuracy of state estimation are central for several system control and optimization tasks, including optimal power flow, unit commitment, economic dispatch, and contingency analysis [2]. However, frequent and sizable state variable fluctuations caused by fast variations of renewable generation, increasing deployment of electric vehicles, and human-in-the-loop demand response incentives, are challenging these functions.

As state variables are difficult to measure directly, the supervisory control and data acquisition (SCADA) system offers abundant measurements, including voltage magnitudes, power flows, and power injections. Given SCADA measurements, the goal of PSSE is to retrieve the state variables, namely complex voltages at all buses [2]. PSSE is typically formulated as a (weighted) least-squares (WLS) or a (weighted) least-absolute-value (WLAV) problem. The former can be underdetermined, and nonconvex in general [182], while the latter can be formulated as a linear programming problem [61, 99] if network only consists of PMUs. In practice however, power grids must include conventional RTUs as well, which leads to highly complex and non-convex solution.

To address these challenges, several efforts have been devoted. WLAV-based estimation for instance can be converted into a constrained optimization, for which a sequential linear programming solver was devised in [77], and improved (stochastic) proximal-linear solvers were developed in [180]. On the other hand, focusing on the WLS criterion, the Gauss-Newton solver is widely employed in practice [2]. Unfortunately, due to the nonconvexity and quadratic loss

function, there are two challenges facing the Gauss-Newton solver: i) sensitivity to initialization; and ii) convergence is generally not guaranteed [220]. Semidefinite programming approaches can mitigate these issues to some extent, at the price of rather heavy computational burden [220]. In a nutshell, the grand challenge of these methods, remains to develop fast and robust PSSE solvers attaining or approximating the global optimum.

To bypass the nonconvex optimization hurdle in power system monitoring and control, recent works have focused on developing data- (and model-) driven neural network (NN) solutions [10, 121, 214, 210, 74, 125, 134]. Such NN-based PSSE solvers approximate the mapping from measurements to state variables based on a training set of measurement-state pairs generated using simulators or available from historical data [214]. However, existing NN architectures do not directly account for the power network topology. On the other hand, a common approach to tackling challenging ill-posed problems in image processing has been to regularize the loss function with suitable priors [146]. Popular priors include sparsity, total variation, and low rank [50]. Recent efforts have also focused on data-driven priors that can be learned from exemplary data [106, 160, 3].

Permeating the benefits of [106, 160] and [3] to power systems, this paper advocates a deep (D) NN-based trainable prior for standard ill-posed PSSE, to promote physically meaningful PSSE solutions. To tackle the resulting regularized PSSE problem, an alternating minimization-based solver is first developed, having Gauss-Newton iterations as a critical algorithmic component. As with Gauss-Newton iterations, our solver requires inverting a matrix per iteration, thus incurring a heavy computational load that may discourage its use for real-time monitoring of large networks. To accommodate real-time operations and building on our previous works [214], we unroll this alternating minimization solver to construct a new DNN architecture, that we term Gauss-Newton unrolled neural networks (GNU-NN) with deep priors. As the name suggests, our DNN model consists of a Gauss-Newton iteration as a basic building block, followed by a proximal step to account for the regularization term. Upon incorporating a graph (G) NN-based prior, our model exploits the structure of the underlying power network. Different from [214], our GNU-NN method offers a systematic and flexible framework to incorporate prior information into standard PSSE tasks.

In practice, measurements collected by the SCADA system may be severely corrupted due to e.g., parameter uncertainty, instrument mis-calibration, and unmonitored topology changes [124, 180]. As cyber-physical systems, power networks are also vulnerable to adversarial

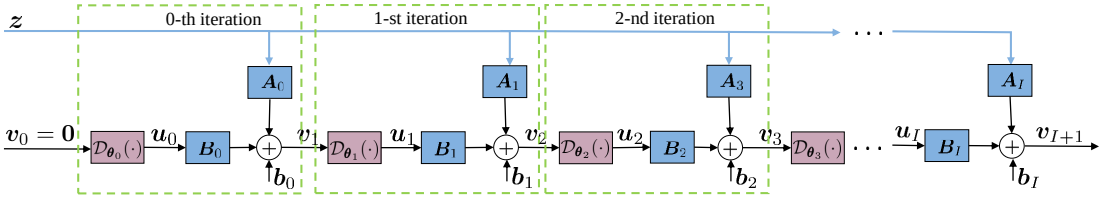



Figure 5.14: The structure of the proposed GNU-NN.

attacks [52, 193], as asserted by the first hacker-caused Ukraine power blackout in 2015 [33]. Furthermore, it has recently been demonstrated that adversarial attacks can markedly deteriorate NNs' performance [93, 126]. Prompted by this, to endow our GNU-NN approach with *robustness* against bad (even adversarial) data, we pursue a principled GNU-NN training method that relies on a distributionally robust optimization formulation. Numerical tests using the IEEE 118-bus benchmark system corroborate the performance and robustness of the proposed scheme.

*Notation.* Lower- (upper-) case boldface letters denote column vectors (matrices), with the exception of vectors $\boldsymbol{V}$, $\boldsymbol{P}$ and $\boldsymbol{Q}$, and normal letters represent scalars. The $(i, j)$th entry, $i$-th row, and $j$-th column of matrix $\boldsymbol{X}$ are $[\boldsymbol{X}]_{i,j}$, $[\boldsymbol{X}]_{i:}$, and $[\boldsymbol{X}]_{:j}$, respectively. Calligraphic letters are reserved for sets except operators $\mathcal{I}$ and $\mathcal{P}$. Symbol $^\top$ stands for transposition; $\mathbf{0}$ denotes all-zero vectors of suitable dimensions; and $\|\boldsymbol{x}\|$ is the $l_2$-norm of vector $\boldsymbol{x}$.

## 5.9 Background and Problem Formulation

Consider an electric grid comprising $N$ buses (nodes) with $E$ lines (edges) that can be modeled as a graph $\mathcal{G} := (\mathcal{N}, \mathcal{E}, \boldsymbol{W})$, where the set $\mathcal{N} := \{1, \ldots, N\}$ collects all buses, $\mathcal{E} := \{(n, n')\} \subseteq \mathcal{N} \times \mathcal{N}$ all lines, and $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ is a weight matrix with its $(n, n')$-th entry $[\boldsymbol{W}]_{nn'} = w_{nn'}$ modeling the impedance between buses $n$ and $n'$. In particular, if $(n, n') \in \mathcal{E}$, then $[\boldsymbol{W}]_{nn'} = w_{nn'}$; and $[\boldsymbol{W}]_{nn'} = 0$ otherwise. For each bus $n \in \mathcal{N}$, let $V_n := v_n^r + jv_n^i$ be its complex voltage with magnitude denoted by $|V_n|$, and $P_n + jQ_n$ its complex power injection. For reference, collect the voltage magnitudes, active and reactive power injections across all buses into the $N$-dimensional column vectors $|\boldsymbol{V}|$, $\boldsymbol{P}$, and $\boldsymbol{Q}$, respectively.

System state variables $\boldsymbol{v} := [v_1^r \; v_1^i \; \ldots v_N^r \; v_N^i]^\top \in \mathbb{R}^{2N}$ can be represented by SCADA measurements, including voltage magnitudes, active and reactive power injections, as well as active and reactive power flows. Let $\mathcal{S}_V$, $\mathcal{S}_P$, $\mathcal{S}_Q$, $\mathcal{E}_P$, and $\mathcal{E}_Q$ denote the sets of buses or lines

where meters of corresponding type are installed. For a compact representation, let us collect the measurements from all meters into $\boldsymbol{z} := [\{|V_n|^2\}_{n\in\mathcal{S}_V}, \{P_n\}_{n\in\mathcal{S}_P}, \{Q_n\}_{n\in\mathcal{S}_Q}, \{P_{nn'}\}_{(n,n')\in\mathcal{E}_P}, \{Q_{nn'}\}_{(n,n')\in\mathcal{E}_Q},]^\top \in \mathbb{R}^M$. Moreover, the $m$-th entry of $\boldsymbol{z} := \{z_m\}_{m=1}^M$, can be described by the following model

$$z_m = h_m(\boldsymbol{v}) + \epsilon_m, \;\; \forall m = 1, \ldots, M \tag{5.20}$$

where $h_m(\boldsymbol{v}) = \boldsymbol{v}^\top \boldsymbol{H}_m \boldsymbol{v}$ for some symmetric measurement matrix $\boldsymbol{H}_m \in \mathbb{R}^{2N\times 2N}$, and $\epsilon_m$ captures the modeling error as well as the measurement noise.

The goal of PSSE is to recover the state vector $\boldsymbol{v}$ from measurements $\boldsymbol{z}$. Specifically, adopting the least-squares criterion and vectorizing the terms in (5.20), PSSE can be formulated as the following nonlinear least-squares (NLS)

$$\boldsymbol{v}^* := \arg\min_{\boldsymbol{v}\in\mathbb{R}^{2N}} \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v})\|^2. \tag{5.21}$$

A number of algorithms have been developed for solving (5.21), including e.g., Gauss-Newton iterations [2], and semidefinite programming-based solvers [220, 96]. Starting from an initial $\boldsymbol{v}_0$, most of these schemes (the former two) iteratively implement a mapping from $\boldsymbol{v}_i$ to $\boldsymbol{v}_{i+1}$, in order to generate a sequence of iterates that hopefully converges to $\boldsymbol{v}^*$ or some point nearby. In the ensuing subsection, we will focus on the 'workhorse' Gauss-Newton PSSE solver.

### 5.9.1 Gauss-Newton Iterations

The Gauss-Newton method is the most commonly used one for minimizing NLS [17, Sec. 1.5.1]. It relies on Taylor's expansion to linearize the function $\boldsymbol{h}(\boldsymbol{v})$. Specifically, at a given point $\boldsymbol{v}_i$, it linearly approximates

$$\tilde{\boldsymbol{h}}(\boldsymbol{v}, \boldsymbol{v}_i) \approx \boldsymbol{h}(\boldsymbol{v}_i) + \boldsymbol{J}_i(\boldsymbol{v} - \boldsymbol{v}_i) \tag{5.22}$$

where $\boldsymbol{J}_i := \nabla \boldsymbol{h}\,(\boldsymbol{v}_i)$ is the $M \times 2N$ Jacobian of $\boldsymbol{h}$ evaluated at $\boldsymbol{v}_i$, with $[\boldsymbol{J}_i]_{m,n} := \partial \boldsymbol{h}_m / \partial \boldsymbol{v}_n$. Subsequently, the Gauss-Newton method approximates the nonlinear term $\boldsymbol{h}(\boldsymbol{v})$ in (5.21) via (5.22), and finds the next iterate as its minimizer; that is,

$$\boldsymbol{v}_{i+1} = \arg\min_{\boldsymbol{v}} \;\; \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v}_i) - \boldsymbol{J}_i(\boldsymbol{v} - \boldsymbol{v}_i)\|^2 . \tag{5.23}$$

Clearly, the per-iteration subproblem (5.23) is convex quadratic. If matrix $\boldsymbol{J}_i^\top \boldsymbol{J}_i$ is invertible, the iterate $\boldsymbol{v}_i$ can be updated in closed-form as

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \left(\boldsymbol{J}_i^\top \boldsymbol{J}_i\right)^{-1} \boldsymbol{J}_i^\top (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v}_i)) \tag{5.24}$$

until some stopping criterion is satisfied. In practice however, due to the matrix inversion, the Gauss-Newton method becomes computationally expensive; it is also sensitive to initialization, and in certain cases it can even diverge. These limitations discourage its use for real-time monitoring of large-scale networks. To address these limitations, instead of solving every PSSE instance (corresponding to having a new set of measurements in $\boldsymbol{z}$) with repeated iterations, an end-to-end approach based on DNNs is pursued next.

## 5.10 Unrolled Gauss-Newton with Deep Priors

As mentioned earlier, PSSE can be underdetermined and thus ill posed due to e.g., lack of observability. To cope with such a challenge, this section puts forth a flexible topology-aware prior that can be incorporated as a regularizer of the PSSE cost function in (5.21). To solve the resultant regularized PSSE, an alternating minimization-based solver is developed. Subsequently, an end-to-end DNN architecture is constructed by unrolling the alternating minimization solver. Such a novel DNN is built using several layers of unrolled Gauss-Newton iterations followed by proximal steps to account for the regularization term. Interestingly, upon utilizing a GNN-based prior, the power network topology can be exploited in PSSE.

### 5.10.1 Regularized PSSE with Deep Priors

In practice, recovering $\boldsymbol{v}$ from $\boldsymbol{z}$ can be ill-posed, for instance when $\boldsymbol{J}_i$ is a rectangular matrix. Building on the data-driven deep priors in image denoising [106, 160, 3], we advocate regularizing any PSSE loss (here, the NLS in (5.21)) with a trainable prior information, as

$$\min_{\boldsymbol{v} \in \mathbb{R}^{2N}} \; \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v})\|^2 + \lambda \, \|\boldsymbol{v} - \mathcal{D}(\boldsymbol{v})\|^2 \tag{5.25}$$

where $\lambda \geq 0$ is a tuning hyper-parameter, while the regularizer promotes states $\boldsymbol{v}$ residing close to $\mathcal{D}(\boldsymbol{v})$. The latter could be a nonlinear $\hat{\boldsymbol{v}}$ estimator (obtained possibly offline) based on training data. To encompass a large family of priors, we advocate a DNN-based estimator $\mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v})$ with

weights $\boldsymbol{\theta}$ that can be learned from historical (training) data. Taking a Bayesian view, the DNN $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ can ideally output the posterior mean for a given input.

Although this regularizer can deal with ill conditioning, the PSSE objective in (5.25) remains nonconvex. In addition, the nested structure of $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ presents further challenges. Similar to the Gauss-Newton method for NLS in (5.21), we will cope with this challenge using an alternating minimization algorithm to iteratively approximate the solution of (5.25). Starting with some initial guess $\boldsymbol{v}_0$, each iteration $i$ uses a linearized data consistency term to obtain the next iterate $\boldsymbol{v}_{i+1}$; that is,

$$\begin{aligned}\boldsymbol{v}_{i+1} &= \arg\min_{\boldsymbol{v}} \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v}_i) - \boldsymbol{J}_i(\boldsymbol{v} - \boldsymbol{v}_i)\|^2 + \lambda\|\boldsymbol{v} - \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}_i)\|^2 \\ &= \boldsymbol{A}_i\boldsymbol{z} + \boldsymbol{B}_i\boldsymbol{u}_i + \boldsymbol{b}_i\end{aligned}$$

where we define

$$\begin{aligned}\boldsymbol{A}_i &:= (\boldsymbol{J}_i^\top\boldsymbol{J}_i + \lambda\boldsymbol{I})^{-1}\boldsymbol{J}_i^\top \\ \boldsymbol{B}_i &:= \lambda(\boldsymbol{J}_i^\top\boldsymbol{J}_i + \lambda\boldsymbol{I})^{-1} \\ \boldsymbol{b}_i &:= (\boldsymbol{J}_i^\top\boldsymbol{J}_i + \lambda\boldsymbol{I})^{-1}\boldsymbol{J}_i^\top(\boldsymbol{J}_i\boldsymbol{v}_i - \boldsymbol{h}(\boldsymbol{v}_i)).\end{aligned}$$

The solution of (5.25) can thus be approached by alternating between the ensuing two steps

$$\boldsymbol{u}_i = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}_i) \tag{5.27a}$$

$$\boldsymbol{v}_{i+1} = \boldsymbol{A}_i\boldsymbol{z} + \boldsymbol{B}_i\boldsymbol{u}_i + \boldsymbol{b}_i. \tag{5.27b}$$

Specifically, with initialization $\boldsymbol{v}_0 = \boldsymbol{0}$ and input $\boldsymbol{z}$, the first iteration yields $\boldsymbol{v}_1 = \boldsymbol{A}_0\boldsymbol{z} + \boldsymbol{B}_0\boldsymbol{u}_0 + \boldsymbol{b}_0$. Upon passing $\boldsymbol{v}_1$ through the DNN $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$, the output $\boldsymbol{u}_1$ at the first iteration, which is also the input to the second iteration, is given by $\boldsymbol{u}_1 = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}_1)$ [cf. (5.27a)]. In principle, state estimates can be obtained by repeating these alternating iterations whenever a new measurement $\boldsymbol{z}$ becomes available. However, at every iteration $i$, the Jacobian matrix $\boldsymbol{J}_i$ must be evaluated, followed by matrix inversions to form $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, and $\boldsymbol{b}_i$. The associated computational burden could be thus prohibitive for real-time monitoring tasks of large-scale power systems.

For fast implementation, we pursue an end-to-end learning approach that trains a DNN constructed by unrolling iterations of this alternating minimizer to approximate directly the

mapping from measurements $\boldsymbol{z}$ to states $\boldsymbol{v}$; see Fig. 5.14 for an illustration of the resulting GNU-NN architecture. Recall that in order to derive the alternating minimizer, the DNN prior $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ in (5.27a) was assumed pre-trained, with weights $\boldsymbol{\theta}$ fixed in advance. In our GNU-NN however, we consider all the coefficients $\{\boldsymbol{A}_i\}_{i=0}^I$, $\{\boldsymbol{B}_i\}_{i=0}^I$, $\{\boldsymbol{b}_i\}_{i=0}^I$, as well as the DNN weights $\{\boldsymbol{\theta}_i\}_{i=0}^I$ to be learnable from data.

This end-to-end GNU-NN can be trained using backpropagation based on historical or simulated measurements $\{\boldsymbol{z}^t\}_{t=1}^T$ and corresponding ground-truth states $\{\boldsymbol{v}^{*t}\}_{t=1}^T$. Entailing only several matrix-vector multiplications, our GNU-NN achieves competitive PSSE performance compared with other iterative solvers such as the Gauss-Newton method. Further, relative to the existing data-driven NN approaches, our GNU-NN can avoid vanishing and exploding gradients. This is possible thanks to direct (a.k.a skipping) connections from the input layer to intermediate and output layers.

*Remark* 6. Albeit the problem remains non-convex, and may converge to a local solution, the key advantage of data-driven-based PSSE comes from utilizing abundant available historical training data. Specifically, the widely used algorithms such as stochastic gradient descent algorithm and its variants, have been successful to escape local minima while updating the NN weights. To prevent practical challenges, such as "overfitting" and offer better generalization performance, large training data sets are oftentimes used in practice. Another feature of NNs and other machine learning approaches is that they alleviate the computational burden at the operation stage by shifting computationally intensive 'hard work' to the off-line training stage. Therefore, the sensitivity, hyper-parameter tuning, and convergence issues are to be tackled mostly during training phase. After the mapping function between the measurement $\boldsymbol{z}$ and state vector $\boldsymbol{v}$ is learned, estimating the states associated with a fresh set of measurements only requires very simple operations, that is, passing the measurements through the learned NN. This would greatly improve the efficiency of PSSE, bringing real-time state estimation within reach.

Interestingly, by carefully choosing the specific model for $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$, desirable properties such as scalability and high estimation accuracy can be also effected. For instance, if we use feed forward NNs as $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$, it is possible to obtain a scalable solution for large power networks. However, feed forward NN can only leverage the grid topology indirectly through simulated MATPOWER data. This prompts us to focus on GNNs, which can explicitly capture the topology and the physics of the power network. The resultant Gauss-Newton unrolled with GNN priors (GNU-GNN) is elaborated next.

### 5.10.2 Graph Neural Network Deep Prior

To allow for richly expressive state estimators to serve in our regularization term, we model $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ through GNNs, that are a prudent choice for networked data. GNNs have recently demonstrated remarkable performance in several tasks, including classification, recommendation, and robotics [89]. By operating directly over graphs, GNNs can explicitly leverage the power network topology. Hence, they are attractive options for parameterization in application domains where data adhere to a graph structure [89].

Consider a graph of $N$ nodes with weighted adjacency matrix $\boldsymbol{W}$ capturing node connectivity. Data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ with $n$-th row $\boldsymbol{x}_n^\top := [\boldsymbol{X}]_{n:}$ representing an $F \times 1$ feature vector of node $n$, is the GNN input. For the PSSE problem at hand, features are real and imaginary parts of the nodal voltage ($F = 2$). Upon pre-multiplying the input $\boldsymbol{X}$ by $\boldsymbol{W}$, features are propagated over the network, yielding a diffused version $\check{\boldsymbol{Y}} \in \mathbb{R}^{N\times F}$ that is given by

$$\check{\boldsymbol{Y}} = \boldsymbol{W}\boldsymbol{X}. \tag{5.28}$$

*Remark* 7. To model feature propagation, a common option is to rely on the adjacency matrix or any other matrix that preserves the structure of the power network (i.e. $\boldsymbol{W}_{nn'} = 0$ if $(n, n') \notin \mathcal{E}$). Examples include the graph Laplacian, the random walk Laplacian, and their normalized versions.

Basically, the shift operation in (5.28) linearly combines the $f$-th features of all neighbors to obtain its propagated feature. Specifically for bus $n$, the shifted feature $[\check{\boldsymbol{Y}}]_{nf}$ is

$$[\check{\boldsymbol{Y}}]_{nf} = \sum_{i=1}^{N} [\boldsymbol{W}]_{ni}[\boldsymbol{X}]_{if} = \sum_{i\in\mathcal{N}_n} w_{ni}x_i^f \tag{5.29}$$

where $\mathcal{N}_n = \{i \in \mathcal{N} : (i, n) \in \mathcal{E}\}$ denotes the set of neighboring buses for bus $n$. Clearly, this interpretation generates a diffused copy or shift of $\boldsymbol{X}$ over the graph.

The 'graph convolution' operation in GNNs exploits topology information to linearly combine features, namely

$$[\boldsymbol{Y}]_{nd} := [\mathcal{H} \star \boldsymbol{X}; \boldsymbol{W}]_{nd} := \sum_{k=0}^{K-1} [\boldsymbol{W}^k\boldsymbol{X}]_{n:}[\boldsymbol{H}_k]_{:d} \tag{5.30}$$

where $\mathcal{H} := [\boldsymbol{H}_0 \ \cdots \ \boldsymbol{H}_{K-1}]$ with $\boldsymbol{H}_k \in \mathbb{R}^{F\times D}$ concatenating all filter coefficients; $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$

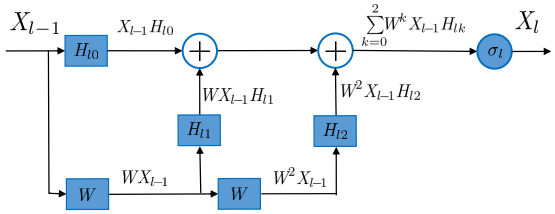



Figure 5.15: The signal diffuses from layer $l-1$ to $l$ with $K=3$.

is the intermediate (hidden) matrix with $D$ features per bus; and $\boldsymbol{W}^k\boldsymbol{X}$ linearly combines features of buses within the $k$-hop neighborhood by recursively applying the shift operator $\boldsymbol{W}$.

To obtain a GNN with $L$ hidden layers, let $\boldsymbol{X}_{l-1}$ denote the output of the $(l-1)$-st layer, which is also the $l$-th layer input for $l=1,\ldots,L$, and $\boldsymbol{X}_0=\boldsymbol{X}$ is the input matrix. The hidden $\boldsymbol{Y}_l\in\mathbb{R}^{N\times D_l}$ with $D_l$ features is obtained by applying the graph convolution operation (5.30) at layer $l$, that is

$$[\boldsymbol{Y}_l]_{nd}=\sum_{k=0}^{K_l-1}[\boldsymbol{W}^k\boldsymbol{X}_{l-1}]_{n:}[\boldsymbol{H}_{lk}]_{:g} \tag{5.31}$$

where $\boldsymbol{H}_{lk}\in\mathbb{R}^{F_{l-1}\times F_l}$ are the graph convolution coefficients for $k=0,\ldots,K_l-1$. The output $\boldsymbol{X}_l$ at layer $l$ is found by applying a graph convolution followed by a point-wise nonlinear operation $\sigma_l(\cdot)$, such as the rectified linear unit (ReLu) $\sigma_l(t):=\max\{0,\,t\}$ for $t\in\mathbb{R}$; see Fig. 5.15 for a depiction. Rewriting (5.31) in a compact form, we arrive at

$$\boldsymbol{X}_l=\sigma_l(\boldsymbol{Y}_l)=\sigma_l\left(\sum_{k=0}^{K_l-1}\boldsymbol{W}^k\boldsymbol{X}_{l-1}\boldsymbol{H}_{lk}\right). \tag{5.32}$$

The GNN-based PSSE provides a nonlinear functional operator $\boldsymbol{X}_L=\boldsymbol{\Phi}(\boldsymbol{X}_0;\boldsymbol{\Theta},\boldsymbol{W})$ that maps the GNN input $\boldsymbol{X}_0$ to voltage estimates by taking into account the graph structure through $\boldsymbol{W}$, through

$$\begin{aligned}&\boldsymbol{\Phi}(\boldsymbol{X}_0;\boldsymbol{\Theta},\boldsymbol{W})=\\&\sigma_L\left(\sum_{k=0}^{K_L-1}\boldsymbol{W}^k\left(\ldots\left(\sigma_1\left(\sum_{k=0}^{K_1-1}\boldsymbol{W}^k\boldsymbol{X}_0\boldsymbol{H}_{1k}\right)\ldots\right)\right)\boldsymbol{H}_{Lk}\right)\end{aligned} \tag{5.33}$$

where the parameter set $\boldsymbol{\Theta}$ contains all the filter weights; that is, $\boldsymbol{\Theta}:=\{\boldsymbol{H}_{lk},\forall l,k\}$, and also recall that $\boldsymbol{X}_0=\boldsymbol{X}$.

**Algorithm 10** PSSE Solver with GNN Priors.

**Training phase:**
**Input:** Training samples $\{(\boldsymbol{z}^t, \boldsymbol{v}^{*t})\}_{t=1}^T$
**Initialize:**
$\boldsymbol{\omega}^1 := [\{\boldsymbol{\Theta}_i^1\}_{i=0}^I, \{\boldsymbol{A}_i^1\}_{i=0}^I, \{\boldsymbol{B}_i^1\}_{i=0}^I, \{\boldsymbol{b}_i^1\}_{i=0}^I]$, $\boldsymbol{v}_0 = 0$.
For $t = 1, 2, \ldots, T$
Feed $\boldsymbol{z}^t$ and $\boldsymbol{v}_0$ as input into GNU-GNN.
For $i = 0, 1, \ldots, I^2$
Reshape $\boldsymbol{v}_i \in \mathbf{R}^{2N}$ to get $\boldsymbol{X}_0^i \in \mathbb{R}^{N\times 2}$.
Feed $\boldsymbol{X}_0^i$ into GNN.
Vectorize the GNN output $\boldsymbol{X}_L^i \in \mathbb{R}^{N\times 2}$ to get $\boldsymbol{u}_i$.
Obtain $\boldsymbol{v}_{i+1} \in \mathbf{R}^{2N}$ using (5.27b).
Obtain $\boldsymbol{v}_{I+1}^t$ using (5.27b).
Minimize the loss $\ell(\boldsymbol{v}^{*t}, \boldsymbol{v}_{I+1}^t)$ and update $\boldsymbol{\omega}^t$.
**Output:** $\boldsymbol{\omega}^T$ **Inference phase:**
For $t = T+1, \ldots, T'$
Feed real-time $\boldsymbol{z}^t$ to the trained GNU-GNN.
Obtain the estimated voltage $\boldsymbol{v}^t$.

*Remark* 8. With $L$ hidden layers, $F_l$ features and $K_l$ filters per layer, the total number of parameters to be learned is $|\boldsymbol{\Theta}| = \sum_{l=1}^L K_l \times F_l \times F_{l-1}$.

To accommodate the GNN implementation over the proposed unrolled architecture, at the $i$-th iteration, we reshape the states $\boldsymbol{v}_i \in \mathbb{R}^{2N}$ to form the $N \times 2$ GNN input matrix $\boldsymbol{X}_0^i \in \mathbb{R}^{N\times 2}$. Next, we vectorize the GNN output $\boldsymbol{X}_L^i \in \mathbb{R}^{N\times 2}$ to obtain the vector $\boldsymbol{u}_i \in \mathbb{R}^{2N}$ (cf. (5.27a)). For notational brevity, we concatenate all trainable parameters of the GNU-GNN in vector $\boldsymbol{\omega} := [\{\boldsymbol{\Theta}_i\}_{i=0}^I, \{\boldsymbol{A}_i\}_{i=0}^I, \{\boldsymbol{B}_i\}_{i=0}^I, \{\boldsymbol{b}_i^1\}_{i=0}^I]$, and let $\boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega})$ denote the end-to-end GNU-GNN parametric model, which for given measurements $\boldsymbol{z}$ predicts the voltages across all buses, meaning $\hat{\boldsymbol{v}} = \boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega})$. The GNU-GNN weights $\boldsymbol{\omega}$ can be updated using backpropagation, after specifying a certain loss $\ell(\boldsymbol{v}^*, \boldsymbol{v}_{I+1})$ measuring how well the estimated voltages $\boldsymbol{v}_{I+1}$ by the GNU-GNN matches the ground-truth ones $\boldsymbol{v}^*$. The proposed method is summarized in Alg. 10.

## 5.11 Robust PSSE Solver

In real-time inference, our proposed GNU-GNN that has been trained using past data, outputs an estimate of the state $\boldsymbol{v}^t$ per time slot $t$ based on the observed measurements $\boldsymbol{z}^t$. However,

due to impulsive communication noise and possibly cyberattacks, our proposed GNU-GNN in Section 5.10 can yield grossly biased estimation results. A natural extension of our approach is to consider these imperfections in the PSSE problem. Therefore, after proposing our method to inject prior information and training the DNN for normal input, we robustify our method in the presence of imperfections in this section.

To obtain estimators robust to bad data, classical formulations including Hüber estimation, Hüber M-estimation, and Schweppe-Hüber generalized M-estimation, rely on the premise that measurements obey $\epsilon$-contaminated probability models; see e.g., [182]. Instead, the present paper postulates that measured and ground-truth voltages are drawn from some nominal yet unknown distribution $P_0$ supported on $\mathcal{S} = \mathcal{Z} \times \mathcal{V}$, that is $(\boldsymbol{z}, \boldsymbol{v}^*) \sim P_0$. Therefore, to obtain the end-to-end GNU-GNN parametric model $\boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega})$, the trainable parameters $\boldsymbol{\omega}$ are optimized by solving $\min_{\boldsymbol{\omega}} \mathbb{E}_{P_0}\big[\ell(\boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega}), \boldsymbol{v}^*)\big]$ [126]. In practice, $P_0$ is unknown but i.i.d. training samples $\{(\boldsymbol{z}^t, \boldsymbol{v}^{*t})\}_{t=1}^T \sim P_0$ are available. In this context, our PSSE amounts to solving for the minimizer of the empirical loss as

$$\min_{\boldsymbol{\omega}} \ \bar{\mathbb{E}}_{\widehat{P}_0^{(T)}}[\ell(\boldsymbol{\pi}(\boldsymbol{z}^t; \boldsymbol{\omega}), \boldsymbol{v}^{*t})] = \frac{1}{T} \sum_{t=1}^{T} \ell(\boldsymbol{\pi}(\boldsymbol{z}^t; \boldsymbol{\omega}), \boldsymbol{v}^{*t}). \tag{5.34}$$

To cope with uncertain and adversarial environments, the solution of (5.34) can be robustified by optimizing over a set $\mathcal{P}$ of probability distributions centered around $\widehat{P}_0^{(T)}$, and minimizing the *worst-case* expected loss with respect to the choice of any distribution $P \in \mathcal{P}$. Concretely, this can be formulated as the following *distributionally robust* optimization

$$\min_{\boldsymbol{\omega}} \sup_{P \in \mathcal{P}} \ \mathbb{E}_P[\ell(\boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega}), \boldsymbol{v}^*)]. \tag{5.35}$$

Compared with (5.34), the worst-case formulation in (5.35) ensures a reasonable performance across a continuum of distributions in $\mathcal{P}$. A broad range of ambiguity sets $\mathcal{P}$ could be considered here. Featuring a strong duality enabled by the optimal transport theory [179], such distributionally robust optimization approaches have gained popularity in robustifying machine learning models [6]. Indeed, this tractability is the key impetus for this section.

Considering probability density functions $P$ and $Q$ defined over support $\mathcal{S}$, let $\Pi(P, Q)$ be the set of all joint probability distributions with marginals $P$ and $Q$. Also let $c : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$

[2]For brevity the superscript $t$ is removed from inner iteration $i$.

be some cost function representing the cost of transporting a unit of mass from $(\boldsymbol{z}, \boldsymbol{v}^*)$ in $P$ to another element $(\boldsymbol{z}', \boldsymbol{v}^*)$ in $Q$ (here we assume that attacker can compromise the measurements $\boldsymbol{z}$ but not the actual system state $\boldsymbol{v}^*$). The so-called optimal transport between two distributions $P$ and $Q$ is given by [179, Page 111]

$$W_c(P, Q) := \inf_{\pi \in \Pi} \mathbb{E}_\pi \big[ c(\boldsymbol{z}, \boldsymbol{z}') \big]. \tag{5.36}$$

Intuitively, $W_c(P, Q)$ denotes the minimum cost associated with transporting all the mass from distribution $P$ to $Q$. Under mild conditions over the cost function and distributions, $W_c$ gives the well-known Wasserstein distance between $P$ and $Q$; see e.g., [169].

Having introduced the distance $W_c$, let us define an uncertainty set for the given empirical distribution $\widehat{P}_0^{(T)}$, as $\mathcal{P} := \{P | W_c(P, \widehat{P}_0^{(T)}) \leq \rho\}$ that includes all probability distributions having at most $\rho$-distance from $P_0^{(T)}$. Incorporating $\mathcal{P}$ into (5.35) yields the following optimization for distributionally robust GNU-GNN estimation

$$\min_{\boldsymbol{\omega}} \sup_{P} \ \mathbb{E}_P[\ell(\boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega}), \boldsymbol{v}^*)] \tag{5.37a}$$

$$\text{s.t. } W_c(P, \widehat{P}_0^{(T)}) \leq \rho. \tag{5.37b}$$

Notice that the inner functional optimization in (5.37a) runs over all probability distributions $P$ characterized by (5.37b). It is intractable to optimize directly over the infinite-dimension distribution functions. Fortunately, for continuous loss as well as transportation cost functions, the optimal objective value of the inner maximization is equal to its dual optimal objective value. In addition, the dual problem involves optimization over only a one-dimension variable. These two observations prompt us to solve (5.37) in the dual domain. To formally obtain this tractable surrogate, we call for a result from [20].

**Proposition 3.** *Let the loss $\ell : \boldsymbol{\omega} \times \mathcal{Z} \times \mathcal{V} \to [0, \infty)$, and transportation cost $c : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$ be continuous functions. Then, for any given $\widehat{P}_0^{(T)}$, and $\rho > 0$, it holds*

$$\sup_{P \in \mathcal{P}} \mathbb{E}_P[\ell(\boldsymbol{\pi}(\boldsymbol{z}; \boldsymbol{\omega}), \boldsymbol{v}^*)] = \tag{5.38}$$

$$\inf_{\gamma \geq 0} \big\{ \bar{\mathbb{E}}_{(\boldsymbol{z}, \boldsymbol{v}^*) \sim \widehat{P}_0^{(T)}} \big[ \sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \ell(\boldsymbol{\pi}(\boldsymbol{\zeta}; \boldsymbol{\omega}), \boldsymbol{v}^*) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta})) \big] \big\}$$

*where* $\mathcal{P} := \Big\{P | W_c(P, \widehat{P}_0^{(T)}) \leq \rho\Big\}$.

*Remark* 9. Thanks to the strong duality, the right-hand side in (5.38) simply is a univariate dual reformulation of the primal problem given on the left-hand side. In contrast with the primal formulation, the expectation in the dual domain is taken only over the empirical distribution $\widehat{P}_0^{(T)}$ rather than over any $P \in \mathcal{P}$. Furthermore, since this reformulation circumvents the need for finding the optimal coupling $\pi \in \Pi$ to define $\mathcal{P}$, and characterizing the primal objective for all $P \in \mathcal{P}$, it is practically appealing and convenient.

Capitalizing on Proposition 3, the inner maximization can be replaced with its dual reformulation. As a consequence, the following distributionally robust PSSE optimization can be arrived at

$$\min_{\boldsymbol{\omega}} \inf_{\gamma \geq 0} \bar{\mathbb{E}}_{(\boldsymbol{z}, \boldsymbol{v}^*) \sim \widehat{P}_0^{(T)}} \Big[\sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \ell(\boldsymbol{\pi}(\boldsymbol{\zeta}; \boldsymbol{\omega}), \boldsymbol{v}^*) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta}))\Big]. \tag{5.39}$$

Finding the optimal solution $(\boldsymbol{\omega}^*, \gamma^*)$ of (5.39) is in general challenging, because it requires the supremum to be solved separately per observed measurements $\boldsymbol{z}$, that cannot readily be handled by existing minimax optimization solvers. A common approach to bypassing this hurdle is to approximate the optimal $\boldsymbol{\omega}^*$ by solving (5.39) with a preselected and fixed $\gamma > 0$ [169]. Indeed, it has been shown in [169] that for any strongly convex transportation cost function, such as $c(\boldsymbol{z}, \boldsymbol{z'}) := \|\boldsymbol{z} - \boldsymbol{z}'\|_p^2$ for any $p \geq 1$, a sufficiently large $\gamma > 0$ ensures that the inner maximization is strongly convex, hence efficiently solvable. Note that having a fixed $\gamma$ is tantamount to tuning $\rho$, which in turn *controls* the level of infused *robustness*. Fixing some large enough $\gamma > 0$ in (5.39), our robustified GNU-GNN model can thus be obtained by solving

$$\min_{\boldsymbol{\omega}} \ \bar{\mathbb{E}}_{(\boldsymbol{z}, \boldsymbol{v}^*) \sim \widehat{P}_0^{(T)}} \Big[\sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \psi(\boldsymbol{\omega}, \boldsymbol{\zeta}; \boldsymbol{z}, \boldsymbol{v}^*)\Big] \tag{5.40}$$

where

$$\psi(\boldsymbol{\omega}, \boldsymbol{\zeta}; \boldsymbol{z}, \boldsymbol{v}^*) := \ell(\boldsymbol{\pi}(\boldsymbol{\zeta}; \boldsymbol{\omega}), \boldsymbol{v}^*) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta})). \tag{5.41}$$

Intuitively, (5.40) can be understood as first 'adversarially' perturbing the measurements $\boldsymbol{z}$ into $\boldsymbol{\zeta}^*$ by maximizing $\psi(\cdot)$, and then seeking a model that minimizes the empirical loss with respect to even such perturbed inputs. Therefore, the robustness of the sought model is achieved to future data that may be contaminated by adversaries. Initialized with some $\boldsymbol{\omega}^0$, and given a

datum $(\boldsymbol{z}^t, \boldsymbol{v}^*)$, we form $\psi(\cdot)$ (c.f. (5.41)), and implement a single gradient ascent step for the inner maximization as follows

$$\boldsymbol{\zeta}^t = \boldsymbol{z}^t + \eta^t \nabla_{\boldsymbol{\zeta}} \psi(\boldsymbol{\omega}^t, \boldsymbol{\zeta}; \boldsymbol{z}^t, \boldsymbol{v}^{*t})\big|_{\boldsymbol{\zeta}=\boldsymbol{z}^t} \tag{5.42}$$

where $\eta^t > 0$ is the step size. Upon evaluating (5.42), the perturbed data $\boldsymbol{\zeta}^t$ will be taken as input (replacing the 'healthy' data $\boldsymbol{z}^t$) fed into Algorithm 10. Having the loss $\ell(\boldsymbol{\pi}(\boldsymbol{\zeta}^t; \boldsymbol{\omega}), \boldsymbol{v}^{*t})$ as solely a function of the GNU-GNN weights $\boldsymbol{\omega}$, the current iterate $\boldsymbol{\omega}^t$ can be updated again by backpropagation.

## 5.12 Numerical Tests

This section tests the estimation performance as well as robustness of our proposed methods.

### 5.12.1 Simulation Setup

The simulations were carried out on an NVIDIA Titan X GPU with a 12GB RAM. For numerical tests, we used real load consumption data from the 2012 Global Energy Forecasting Competition (GEFC) [1]. Using this dataset, training and testing collections were prepared by solving the AC power flow equations using the MATPOWER toolbox. To match the scale of power demands, we normalized the load data, and fed it into MATPOWER to generate $1,000$ pairs of measurements and ground-truth voltages, $80\%$ of which were used for training while the remaining $20\%$ were employed for testing. Measurements include all sending-end active power flows, as well as voltage magnitudes, corrupted by additive white Gaussian noise. Standard deviations of the noise added to power flows and voltage magnitudes were set to $0.02$ and $0.01$ [180], respectively.

A reasonable question to ponder is whether explicitly incorporating the power network topology through a trainable regularizer offers improved performance over competing alternatives. In addition, it is of interest to study how a distributionally robust training method enhances PSSE performance in the presence of bad data and even adversaries. To this aim, four baseline PSSE methods were numerically tested, including one optimization-based method Wirtinger-Flow Gauss-Newton algorithm in [49], and three data-driven methods: i) the prox-linear network in [214]; ii) a 6-layer vanilla feed-forward (F)NN; and iii) an 8-layer FNN. The weights of these NNs were trained using the Adam optimizer to minimize the Hüber loss. The learning rate was

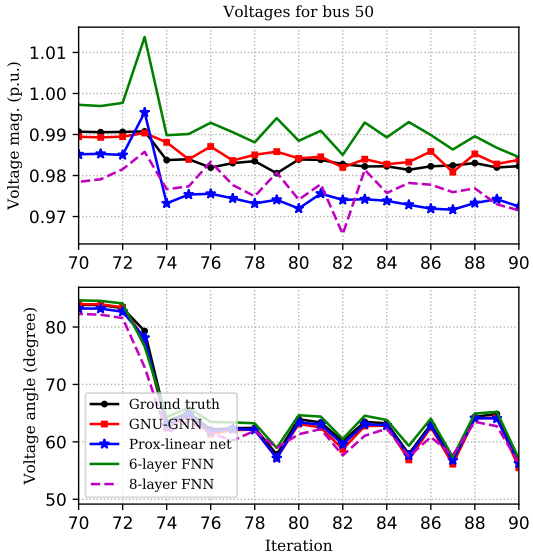



Figure 5.16: The estimated voltage magnitudes and angles by the four schemes at bus $50$ from slots $70$ to $90$.

fixed to $10^{-3}$ throughout $500$ epochs, and the batch size was set to $32$. Furthermore, the average estimation accuracy of each algorithm is defined as follows

$$\nu = \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{v}_n - \boldsymbol{v}_n^*\|_2^2 \tag{5.43}$$

where $\boldsymbol{v}_n$ is the estimated voltage profile from the noisy measurements generated using $\boldsymbol{v}_n^*$.

### 5.12.2 GNU-GNN for regularized PSSE

In the first experiment, we implemented GNU-GNN by unrolling $I = 6$ iterations of the proposed alternating minimizing solver, respectively. A GNN with $K = 2$ hops, and $D = 8$ hidden units with ReLU activations per unrolled iteration was used for the deep prior of GNU-GNN. The GNU-GNN architecture was designed to have total number of weight parameters roughly the same as that of the prox-linear network. The Gauss-Newton algorithm is initialized using the flat voltage profile. Table I tabulates the average performance of the proposed GNU-GNN approach, the Gauss-Newton method, the prox-linear network, $6$-layer FNN and $8$-layer FNN over $200$ testing samples. We report the accuracy of estimation on the IEEE $118$-bus feeder, and the

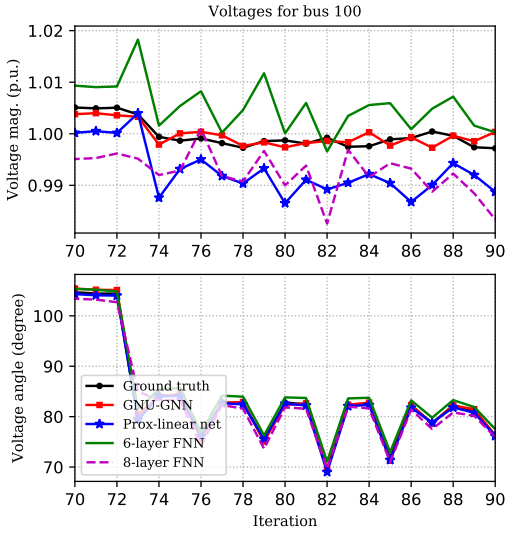



Figure 5.17: The estimated voltage magnitudes and angles by the four schemes at bus 100 from slot 70 to 90.

running time (s) on IEEE 118-bus feeder, as well as IEEE 300-bus feeder. Clearly, the proposed GNU-GNN approach achieves superior performance where the accuracy of estimation is an order of magnitude better than the state-of-the-art Gauss-Newton approach on 118-bus feeder. In addition, since the GNU-GNN approach alleviates almost all the computational burden at the estimation time by shifting it to the training time, the running time of the proposed approach is three orders of magnitude less than the optimization-based approach on both IEEE 118-bus feeder and IEEE 300-bus feeder.

In order to show the quality of estimates provided by the proposed GNU-GNN method, we present the estimate of the voltage magnitudes and angles on the IEEE 118-bus feeder. As is shown in Table I, the estimation performance of the Gauss-Newton method is too bad to depict in the same figure with other alternatives. Therefore, we did not include the Gauss-Newton in this set of results. Figs. 5.16 and 5.17 show the estimated voltage profiles obtained at buses 50 and 100 from test slots 70 to 90, respectively. The ground-truth and estimated voltages for the first 20 buses on the test slot 80 are presented in Fig. 5.18. These results corroborate the improved performance of our GNU-GNN relative to the simulated PSSE solvers.

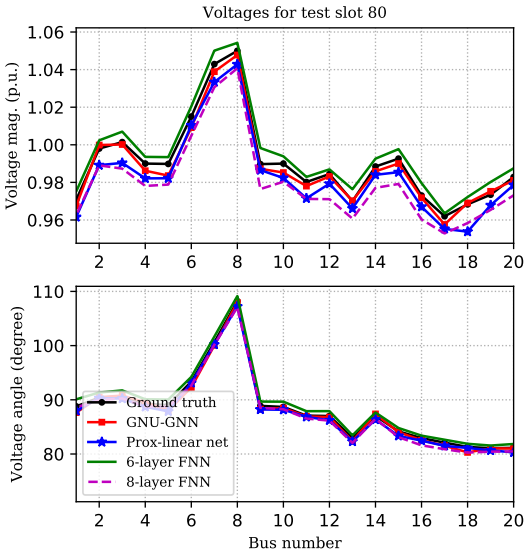



Figure 5.18: The estimated voltages magnitudes and angles by the four schemes for the first 20 buses at slot 80.

### 5.12.3 Robust PSSE

Despite their remarkable performance in standard PSSE, DNNs may fail to yield reliable and accurate estimates in practice when bad data are present. Evidently, this challenges their application in safety-critical power networks. In the experiment of this subsection we examine the robustness of our GNU-GNN trained with the described adversarial learning method on the IEEE 118-bus feeder. To this aim, a distributionally robust learning scheme was implemented to manipulate the input of GNU-GNN, prox-linear net, 6-layer FNN, and 8-layer FNN models. Specifically, under distributional attacks, an ambiguity set $\mathcal{P}$ comprising distributions centered at the nominal data-generating $P_0$ was postulated. Although the training samples were generated according to $P_0$, testing samples were obtained by drawing samples from a distribution $P \in \mathcal{P}$ that yields the worst empirical loss. To this end we preprocessed test samples using (5.42) to generate adversarially perturbed samples. Figs. 5.19 and 5.20 demonstrate the estimated voltage profiles under a distributional attack with a fixed $\gamma = 0.13$ (c.f. (5.40) and (5.41)) and $\ell_2$ transportation cost. As the plots showcase, our proposed robust training method enjoys guarantees against distributional uncertainties, especially relative to competing alternatives.

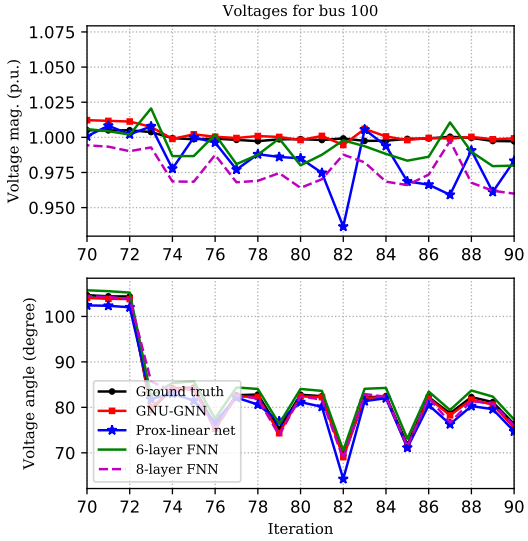



Figure 5.19: The estimated voltage magnitudes and angles by the four schemes under distributional attacks at bus 100 from slots 70 to 90.

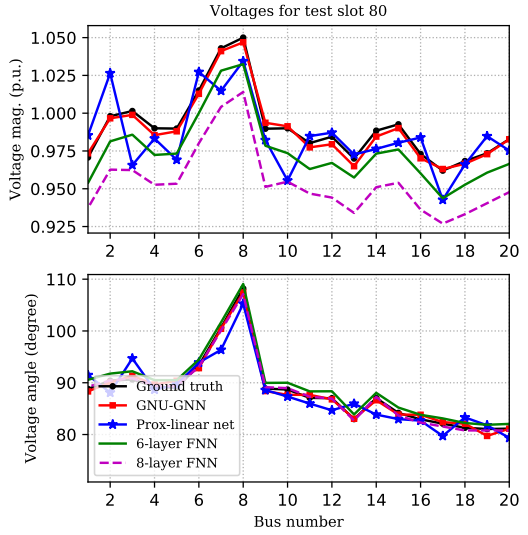



Figure 5.20: The estimated voltage magnitudes and angles by the four schemes under distributional attacks for the first 20 buses at slot 80.

## 5.13 Conclusions

This section introduced topology-aware DNN-based regularizers to deal with the ill-posed and nonconvex characteristics of standard PSSE approaches. An alternating minimization solver was developed to approach the solution of the regularized PSSE objective function, which is further unrolled to construct a DNN model. For real-time monitoring of large-scale networks, the resulting DNN was trained using historical or simulated measured and ground-truth voltages. A basic building block of our GNU-GNN consists of a Gauss-Newton iteration followed by a proximal step to deal with the regularization term. Numerical tests showcased the competitive performance of our proposed GNU-GNN relative to several existing ones. Further, a distributionally robust training method was presented to endow the GNU-GNN with resilience to bad data that even come from adversarial attacks.

# Chapter 6

# Summary and Future Directions

Leveraging recent advances in machine learning, deep learning models in conjunction with statistical signal processing, this thesis pioneered robust, deep, reinforced learning algorithms with applications in management and control of cyber-physical systems. In this final chapter, we provide a summary of the main results discussed in this thesis, and also point out a few possible directions for future research.

## 6.1 Thesis Summary

Chapter 2 dealt with distributionally robust learning, where the data distribution was considered unknown. A framework to robustify parametric machine learning models against distributional uncertainties was put forth, where the so-called Wasserstin distance metric was used to quantify the distance between training and testing data generating data distributions.

Chapter 3 explored robust semi-supervised learning over graphs. To account for uncertainties associated with data distributions, or adversarially manipulated input data, a principled robust learning framework was developed. Using the parametric models of graph neural networks (GNNs), we were able to reconstruct the unobserved nodal values. Experiments corroborated the outstanding performance of the novel method when the input data are corrupted.

Chapter 4 targeted a network resource allocation problem, namely the caching. The idea of caching was to device some entities in a wireed and wireless network with storage capacity. These devices are to store reusable information during off-the peak instances, and then reuse them during on-peak demand periods. By smartly storing popular contents, these devices efficiently

help the network to decrease the operational costs and increase user satisfaction. Especially, we designed a generic setup where a caching unit makes sequential fetch-cache decisions based on dynamic content popularities in local section as well as global network. Critical practical constraints were identified, the aggregated cost across files and time instants was formed, and the optimal adaptive caching was then formulated as: i) classical reinforcement learning algorithm; ii) Deep reinforcement learning problem; and, iii) a stochastic optimization problem. To address the inherent functional estimation problem that arises in each type of considered problems, while leveraging the underlying problem structure, several computationally efficient algorithms were developed.

Finally, Chapter 5 developed a suite of methods to efficiently monitor, and manage the smart grid. Especially, we started with voltage regulation problem using joint control of traditional utility-owned equipment and contemporary smart inverters to inject reactive power. To account for the different response times of those assets, a two-timescale approach to minimizing bus voltage deviations from their nominal values was put forth, by combining physics- and data-driven stochastic optimization. Load consumption and active power generation dynamics were modeled as MDPs. On a fast timescale, the setpoints of smart inverters were found by minimizing the instantaneous bus voltage deviations, while on a slower timescale, the capacitor banks were configured to minimize the long-term expected voltage deviations using a deep reinforcement learning algorithm. Then we considered the second problem of monitoring the smart grid. In particular, topology-aware DNN-based regularizers were developed to deal with the ill-posed and nonconvex characteristics of standard power system state estimation approaches (PSSE). An alternating minimization solver was developed to approach the solution of the regularized PSSE objective function, which was further unrolled to construct a DNN model.[1]

## 6.2 Future Research

The contributions in this thesis opens up a broad range of interesting directions to explore and new problems to solve. Some of such possible research directions are briefly discussed next.

[1]Due to space limitations, a few works of this PhD thesis have not been reported here, including [204].

### 6.2.1 Multi-agent, distributionally robust decentralized RL

We will investigate consensus-based decentralized optimization for our scalable and distributionally robust RL framework, with the ultimate goal of developing safe, multi-agent RL algorithms operating in complex real-world environments. Autonomous driving for instance, is naturally a multi-agent collaborative setting, where the host vehicle (a.k.a. the planner) must apply sophisticated negotiation skills with other road users (agents), when overtaking, giving way, merging, taking left/right turns, or when pushing ahead in unstructured urban roadways. We will also broaden the scope of our multi-step Lyapunov tool, to obtain non-asymptotic performance guarantees of the proposed multi-agent, distributionally robust RL algorithms. We will also corroborate our analytical results extensive experiments.

### 6.2.2 Robust learning approach to fairness in machine learning.

Fairness-aware machine learning algorithms seek methods under which the predicted outcome of a classifier is fair or non-discriminatory based on sensitive attributes such as race, sex, religion, etc. Broadly, fairness-aware machine learning algorithms have been categorized as those *pre-processing* techniques designed to modify the input data so that the outcome of any machine learning algorithm applied to that data will be fair [145]. Preprocessing algorithms considers training data as the cause of the discrimination. This simply is because the training data itself captures historical discrimination or since there are more subtle patterns in the data. Feature modification, data set massaging, and learning unbiased data transformation are examples for this class of methods [54, 29, 81]. The *algorithm modification* techniques on the other hand modify an existing algorithm or create a new one that will be fair under any inputs. Algorithm modification target specific learning algorithms, e.g., by imposing additional constraints. These methods have been by far the most common methods to promote fairness. Among popular techniques in this class are the regularization techniques, convex relaxation of fairness promoting constraints, and training separate models for each value of a sensitive attribute [83, 209, 28]. Combined preprocessing and algorithm modification methods are among effective methods at classification [212]. Finally, the *post-processing* techniques enforce the output of any model to be fair. These methods modify the results of a previously trained classifier to achieve the desired results on different groups. For example modifying the labels of leaves in a decision tree to satisfy fairness constraints, or modifying error profiles to name a few [82, 70, 192]. Despite their

success in dealing with some sensitive features, these proposed methods cannot handle setups where the data is coming from unbalanced mixture of distributions where we should protect at least one of classes. For example, consider data is comming from a mixture of distributions, that is $\{\mathbf{x}_n, y_n\}_{n=1}^N \sim P := \alpha Q_0 + (1-\alpha)Q_1$, where $\alpha \in [0, 1)$ is subpopulation portion, and $Q_0$ and $Q_1$ are *unknown* subpopulations. The classical learning approaches do not guarantee to ensure *equitable* performance for data from both $Q_0$ and $Q_1$, especially for small $\alpha$. To offer a fair model, we instead focus on the worst-case risk that finds model parameters $\boldsymbol{\theta}$ by minimizing the loss over the latent subpopulation $Q_0$

$$\underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}}\ \mathbb{E}_{\mathbf{x}\sim Q_0}\mathbb{E}[\ell(\boldsymbol{\theta}; (\mathbf{x}, y))] \tag{6.1}$$

Since $\alpha$ and $Q_0$ are unknown, it is impossible to compute the loss here from observed data. Therefore we postulate a lower bound $\alpha_0 \in (0, 1/2)$ on the subpopulation proportion $\alpha$ and consider a set of potential minority subpopulations $\mathcal{P}_{\alpha_0} := \{Q_0 : P = \alpha Q_0 + (1-\alpha)Q_1 \text{ for some } \alpha \geq \alpha_0\}$, and target to solve the worst case loss, formulated as follows

$$\underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \sup_{Q_0\in\mathcal{P}_{\alpha_0}} \mathbb{E}[\ell(\boldsymbol{\theta}; (\mathbf{x}, y))]. \tag{6.2}$$

Relying on the proposed techniques described in this T2, we will provide tractable approaches to solve this optimization problem.

### 6.2.3 Communication- and computation-efficient robust federated learning

The recently growing need to learn from massive datasets that are distributed across multiple sites, has propelled research to replace a single learner with multiple learners (a.k.a. workers) exchanging information with a server to learn the sought learning model while abiding with the privacy of local data. Albeit appealing for its scalability, to endow this so-termed federated learning with robustness too, we must address the server-workers communication overhead that is known to constitute the bottleneck in this setup. This becomes aggravated in deep learning, where one may have to deal with millions of unknown parameters.

To outline our research outlook in this setting, consider $M$ workers with each worker $m \in \mathcal{M}$ collecting samples $\{\mathbf{z}_t(m)\}_{t=1}^T$, and a globally shared model $\boldsymbol{\theta}$ that is to be updated at the server by aggregating gradients computed locally per worker. Bandwidth and privacy

concerns discourage uploading these distributed data to the server, which necessitates training to be performed by having workers communicating *iteratively* with the server. To endow such a distributed learning approach with robustness, we will consider solving the following optimization problem in a distributed fashion

$$\min_{\boldsymbol{\theta}\in\Theta} \sup_{P\in\mathcal{P}} \mathbb{E}_{\mathbf{z}\sim P}[\ell(\mathbf{z};\boldsymbol{\theta})]\,, \quad \text{s. to.} \quad \mathcal{P} := \Big\{P \Big| \sum_{m=1}^{M} W_c(P, \widehat{P}^{(T)}(m) \leq \rho\Big\} \tag{6.3}$$

where $W_c(P, \widehat{P}^{(T)}(m))$ is the distance between $P$ and the locally available distribution $\widehat{P}^{(T)}(m)$ per learner $m$. A critical task here is to efficiently handle the communication overhead while guaranteeing the desired robustness. To this aim, we will develop a general framework building on the considered distributionally robust approach. Tapping into our expertise in communication-efficient decentralized learning and wireless sensor networks [130, 113, 15, 187], we will develop methods to integrate distributional robustness in a large-scale parallel architecture with communication-friendly learning schemes through *quantization*, and *censoring*. The quantized gradients computed locally at the learners will be transmitted to the server at a controllably low cost; while censoring will save communication costs in learner-server rounds by simply skipping less informative gradients. To maximize the communication efficiency, we will further investigate two-way communication compression, meaning we will compress both the upload and download information to a limited number of bits. It will be interesting to delineate the tradeoffs emerging between robustness, overhead reduction, and convergence rate of the learning iterates. To this end, we will investigate performance both analytically, as well as with thorough numerical tests.

# References


[1] [Online]. Available: https://www.kaggle.com/c/global-energy-forecasting-competition-2012- load-forecasting/data.

[2] A. Abur and A. G. Exposito, *Power System State Estimation: Theory and Implementation*. New York, USA: CRC Press, 2004.

[3] H. K. Aggarwal, M. P. Mani, and M. Jacob, "MoDL: Model-based deep learning architecture for inverse problems," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 394–405, Aug. 2018.

[4] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[5] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed, "Femtocells: Past, present, and future," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 497–508, Apr. 2012.

[6] C. Bandi and D. Bertsimas, "Robust option pricing," *Eur. J. Oper. Res.*, vol. 239, no. 3, pp. 842–853, Dec. 2014.

[7] M. Baran and F. F. Wu, "Optimal sizing of capacitors placed on a radial distribution system," *IEEE Trans. Power Del.*, vol. 4, no. 1, pp. 735–743, Jan. 1989.

[8] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Trans. Power Del.*, vol. 4, no. 2, pp. 1401–1407, Apr. 1989.

[9] ——, "Optimal capacitor placement on radial distribution systems," *IEEE Trans. Power Del.*, vol. 4, no. 1, pp. 725–734, Jan. 1989.

[10] P. P. Barbeiro, J. Krstulovic, H. Teixeira, J. Pereira, F. J. Soares, and J. P. Iria, "State estimation in distribution smart grids using autoencoders," in *IEEE Intl. Power Eng. and Opt. Conf.*, 2014, pp. 358–363.

[11] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," *SustKDD*, vol. 111, no. 112, p. 108, Aug. 2012.

[12] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Intl. Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, Mumbai, India, May 2015, pp. 161–166.

[13] M. Bazrafshan, N. Gatsis, and H. Zhu, "Optimal tap selection of step-voltage regulators in Multi-phase distribution networks," in *Proc. of Power Syst. Comput. Conf.*, Dublin, Irelands, Jun. 11-15 2018.

[14] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[15] D. Berberidis, V. Kekatos, and G. B. Giannakis, "Online censoring for large-scale regressions with application to streaming big data," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 3854–3867, Aug. 2016.

[16] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "AdaDIF: Adaptive diffusions for efficient semi-supervised learning over graphs," *Intl. Conf. on Big Data*, pp. 92–99, 2018.

[17] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[18] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, Apr. 2016.

[19] J. Blanchet, Y. Kang, F. Zhang, and K. Murthy, "Data-driven optimal transport cost selection for distributionally robust optimization," *Stat.*, vol. 1050, pp. 1527–1554, 2006.

[20] J. Blanchet and K. Murthy, "Quantifying distributional model risk via optimal transport," *Math. Oper. Res.*, vol. 44, pp. 565–600, 2019.

[21] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *IEEE Intl. Conf. on Commun.*, Sydney, Australia, June 10-14, 2014, pp. 1897–1903.

[22] P. Blasco and D. Gündüz, "Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds," *arXiv preprint arXiv:1407.6154*, 2014.

[23] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *Intl. Conf. on Communications*, London, UK, June 2015, pp. 3358–3363.

[24] V. S. Borkar and S. P. Meyn, "The ODE method for convergence of stochastic approximation and reinforcement learning," *SIAM J. Control Optm.*, vol. 38, no. 2, pp. 447–469, 2000.

[25] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Intl. Conf. Comput. Commun.*, San Diego, CA, USA, Mar. 15-19, 2010, pp. 1–9.

[26] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. Intl. Conf. Comput. Commun.*, New York, USA, March 1999, pp. 126–134.

[28] T. Calders and S. Verwer, "Three naive bayes approaches for discrimination-free classification," *Data Mining Knwl. Discov.*, vol. 21, no. 2, pp. 277–292, 2010.

[29] F. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney, "Optimized pre-processing for discrimination prevention," in *Adv. Neural Info. Process. Syst.*, 2017, pp. 3992–4001.

[30] J. A. Carta, P. Ramirez, and S. Velazquez, "A review of wind speed probability distributions used in wind energy analysis: Case studies in the Canary Islands," *Renew. Sust. Energ. Rev.*, vol. 13, no. 5, pp. 933–955, Jun. 2009.

[31] P. M. Carvalho, P. F. Correia, and L. A. Ferreira, "Distributed reactive power generation control for voltage rise mitigation in distribution networks," *IEEE Trans. Power Syst.*, vol. 23, no. 2, pp. 766–772, May 2008.

[32] D. U. Case, "Analysis of the cyber attack on the Ukrainian power grid," 2016.

[33] ——, "Analysis of the cyber attack on the Ukrainian power grid," *E-ISAC*, vol. 388, Mar. 2016.

[34] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv:1810.00069*, 2018.

[35] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *IEEE Trans. Neural Netw.*, vol. 3, p. 542, 2009.

[36] B. Chen, C. Yang, and Z. Xiong, "Optimal caching and scheduling for cache-enabled D2D communications," *IEEE Communications Letters*, vol. 21, no. 5, pp. 1155–1158, May 2017.

[37] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *arXiv:1909.07972*, 2019.

[38] T. Chen, A. G. Marques, and G. B. Giannakis, "DGLB: Distributed stochastic geographical load balancing over cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1866–1880, July 2017.

[39] B. J. Claessens, P. Vrancx, and F. Ruelens, "Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3259–3269, July 2018.

[40] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for Mobile Broadband*. Academic press, 2013.

[41] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *Intl. Conf. Comput. Commun.*, Orlando, FL, USA, Mar. 25-30, 2012, pp. 2444–2452.

[42] J. M. Danskin, "The theory of max-min, with applications," *SIAM J. Appl. Math.*, vol. 14, no. 4, pp. 641–664, 1966.

[43] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1635–1648, June 2017.

[44] E. Delage and Y. Ye, "Distributionally robust optimization under moment uncertainty with application to data-driven problems," *Operations research*, vol. 58, no. 3, pp. 595–612, 2010.

[45] R. Diao, Z. Wang, D. Shi, Q. Chang, J. Duan, and X. Zhang, "Autonomous voltage control for grid operation using deep reinforcement learning," in *Proc. of PESGM*, Atlanta, GA, Aug. 4-8, 2019, pp. 1–5.

[46] Y. Dong, M. Z. Hassan, J. Cheng, M. J. Hossain, and V. C. Leung, "An edge computing empowered radio access network with UAV-mounted FSO fronthaul and backhaul: Key challenges and approaches," *IEEE Wirel. Commun.*, vol. 25, no. 3, pp. 154–160, Jul. 2018.

[47] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hugl, "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, Dec. 2009.

[48] J. Duan, H. Xu, and W. Liu, "Q-learning-based damping control of wide-area power systems under cyber uncertainties," *IEEE Trans. Smart Grid*, vol. 9, no. 6, pp. 6408–6418, Nov 2018.

[49] I. Dzafic, R. A. Jabr, and T. Hrnjic, "Hybrid state estimation in complex variables," *IEEE Trans. Power Systems*, vol. PP, no. 99, pp. 1–1, 2018.

[50] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*. Berlin, HR: SSBM, 1996, vol. 375.

[51] D. Ernst, M. Glavic, and L. Wehenkel, "Power systems stability control: Reinforcement learning framework," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 427–435, Feb. 2004.

[52] P. Fairley, "Cybersecurity at U.S. utilities due for an upgrade: Tech to detect intrusions into industrial control systems will be mandatory," *IEEE Spectr.*, vol. 53, no. 5, pp. 11–13, May 2016.

[53] M. Farivar, C. R. Clarke, S. H. Low, and K. M. Chandy, "Inverter VAR control for distribution systems with renewables," in *Proc. IEEE SmartGridComm.*, Brussels, Belgium, Oct. 2011, pp. 457–462.

[54] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *Proc. ACM SIGKDD*. ACM, 2015, pp. 259–268.

[55] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, Mar. 2019.

[56] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, pp. 1034–1049, 2019.

[57] L. Gan, N. Li, U. Topcu, and S. H. Low, "Exact convex relaxation of optimal power flow in radial networks," *IEEE Trans. on Autom. Control*, vol. 60, no. 1, pp. 72–87, Jan. 2015.

[58] J. Gao and R. Jamidar, "Machine learning applications for data center optimization," *Google White Paper*, Oct. 27, 2014.

[59] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006. [Online]. Available: http://dx.doi.org/10.1561/1300000001

[60] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. P. How, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.

[61] M. Göl and A. Abur, "Lav based robust state estimation for systems measured by PMUs," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1808–1814, 2014.

[62] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[63] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT Press, 2016, vol. 1.

[64] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2014.

[65] S. Gopalakrishnan, Z. Marzi, U. Madhow, and R. Pedarsani, "Combating adversarial attacks using sparse representations," *arXiv:1803.03880*, 2018.

[66] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," 2014.

[67] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv:1412.5068*, 2014.

[68] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," *arXiv:1711.00117*, 2017.

[69] Z. Guo, W. Chen, Y.-F. Liu, Y. Xu, and Z.-L. Zhang, "Joint switch upgrade and controller deployment in hybrid software-defined networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1012–1028, Mar. 2019.

[70] M. Hardt, E. Price, N. Srebro *et al.*, "Equality of opportunity in supervised learning," in *Adv. Neural info. Process. Syst.*, 2016, pp. 3315–3323.

[71] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[72] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference

alignment wireless networks," *IEEE Trans. Vehicular Tech.*, vol. 66, no. 11, pp. 10 433–10 445, Nov. 2017.

[73] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[74] X. Hu, H. Hu, S. Verma, and Z.-L. Zhang, "Physics-guided deep neural networks for power flow analysis," *arXiv:2002.00097*, 2020.

[75] Z. Hu and L. J. Hong, "Kullback-leibler divergence constrained distributionally robust optimization," *Available at Optimization Online*, 2013.

[76] A. Ipakchi and F. Albuyeh, "Grid of the future," *IEEE Power Energy Mag.*, vol. 7, no. 2, pp. 52–62, Feb. 2009.

[77] R. Jabr and B. Pal, "Iteratively re-weighted least absolute value method for state estimation," *IET Gener., Transmiss., Distrib.*, vol. 150, no. 4, pp. 385–391, Jul. 2003.

[78] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[79] ——, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 1, pp. 176–189, Jan. 2016.

[80] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study," *arXiv:2003.00653*, 2020.

[81] F. Kamiran and T. Calders, "Classifying without discriminating," in *Intl. Conf. Comput. Cont. Commun.* IEEE, 2009, pp. 1–6.

[82] F. Kamiran, T. Calders, and M. Pechenizkiy, "Discrimination aware decision tree learning," in *IEEE Intl. Conf. Data Mining*. IEEE, 2010, pp. 869–874.

[83] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Fairness-aware classifier with prejudice remover regularizer," in *Joint European Conf. Machine Learn. Knwl. Discov. Db.* Springer, 2012, pp. 35–50.

[84] V. Kekatos, L. Zhang, G. B. Giannakis, and R. Baldick, "Voltage regulation algorithms for multiphase power distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3913–3923, Sep. 2016.

[85] V. Kekatos, G. Wang, A. J. Conejo, and G. B. Giannakis, "Stochastic reactive power management in microgrids with renewables," *IEEE Trans. Power Syst.*, vol. 30, no. 6, pp. 3386–3395, Dec. 2015.

[86] V. Kekatos, L. Zhang, G. B. Giannakis, and R. Baldick, "Voltage regulation algorithms for multiphase power distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3913–3923, Sep. 2016.

[87] H. Kim, J. Park, M. Bennis, S.-L. Kim, and M. Debbah, "Ultra-dense edge caching under spatio-temporal demand and network dynamics," *arXiv preprint arXiv:1703.01038*, 2017.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Intl. Conf. Learn. Rep.*, May 2015.

[89] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Intl. Conf. Lear. Rep.*, 2016.

[90] E. D. Kolaczyk and G. Csárdi, *Statistical analysis of network data with R*. Springer, 2014, vol. 65.

[91] N. Konstantinov and C. Lampert, "Robust learning from untrusted sources," *Intl. Conf. Mach. Learn.*, June 2019.

[92] P. Kundur, N. J. Balu, and M. G. Lauby, *Power System Stability and Control*. Duisburg, Germany: McGraw-hill New York, May 1994.

[93] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *Intl. Conf. Learn. Rep.*, Vancouver, BC, Canada, Apr.

[94] ——, "Adversarial machine learning at scale," *Intl. Conf. Learn. Rep.*, Apr. 2017.

[95] J. Kwak, G. Paschos, and G. Iosifidis, "Dynamic cache rental and content caching in elastic wireless CDNs," in *Proc. Intl. Symp. Modeling Opt. Mobile, Ad Hoc, Wireless Netw.*, Shanghai, China, May 2018, pp. 1–8.

[96] Y. Lan, H. Zhu, and X. Guan, "Fast nonconvex SDP solvers for large-scale power system state estimation," *IEEE Trans. Power Syst.*, 2020.

[97] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Intl. Conf. on Computer Communications*, San Francisco, USA, Apr. 2016, pp. 1–9.

[98] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.

[99] S. Li, A. Pandey, and L. Pileggi, "A WLAV-based robust hybrid state estimation using circuit-theoretic approach," *arXiv:2011.06021*, 2020.

[100] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[101] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv:1812.06127*, 2018.

[102] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tut.*, Apr. 8 2020.

[103] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *arXiv:2006.07242*, 2020.

[104] T. Lin, C. Jin, and M. I. Jordan, "On gradient descent ascent for nonconvex-concave minimax problems," *arXiv:1906.00331*, 2019.

[105] W. Lin, R. Thomas, and E. Bitar, "Real-time voltage regulation in distribution systems via decentralized PV inverter control," in *Proc. Annual Hawaii Intl. Conf. System Sciences*, Waikoloa Village, Hawaii, Jan. 2-6, 2018.

[106] S. G. Lingala and M. Jacob, "Blind compressive sensing dynamic MRI," *IEEE trans. Med. Imag.*, vol. 32, no. 6, pp. 1132–1145, Mar. 2013.

[107] J. Liu, B. Bai, J. Zhang, and K. B. Letaief, "Content caching at the wireless network edge: A distributed algorithm via belief propagation," in *Intl. Conf. on Communications*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[108] L. M. Lopez-Ramos, A. G. Marques, and J. Ramos, "Jointly optimal sensing and resource allocation for multiuser interweave cognitive radios," *IEEE Trans. Wireless Commun.*, vol. 13, no. 11, pp. 5954–5967, Nov. 2014.

[109] S. H. Low, "Convex relaxation of optimal power flow—Part II: Exactness," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 177–189, May 2014.

[110] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 446–454.

[111] Q. Lu, V. N. Ioannidis, and G. B. Giannakis, "Semi-supervised learning of processes over multi-relational graphs," in *IEEE Intl. Conf. Acoustics, Speech Signal Proces.*, 2020, pp. 5560–5564.

[112] R. Lu, S. H. Hong, and M. Yu, "Demand response for home energy management using reinforcement learning and artificial neural network," *IEEE Trans. Smart Grid*, Apr. 2019.

[113] X. Luo and G. B. Giannakis, "Energy-constrained optimal quantization for wireless sensor networks," *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 73:1–73:12, Jan. 2008.

[114] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surv. Tutor.*, pp. 1–1, to appear 2019.

[115] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[116] ——, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[117] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015.

[118] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv:1706.06083*, 2017.

[119] S. Magnússon, G. Qu, C. Fischione, and N. Li, "Voltage control using limited communication," *IEEE Trans. Control Netw. Syst.*, vol. 6, no. 3, pp. 993–1003, Sep. 2019.

[120] S. Mahadevan, "Learning representation and control in markov decision processes: New frontiers," *Foundations and Trends in Machine Learning*, vol. 1, no. 4, pp. 403–565, 2009.

[121] E. Manitsas, R. Singh, B. C. Pal, and G. Strbac, "Distribution system state estimation using an artificial neural network approach for pseudo measurement modeling," *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 1888–1896, Nov. 2012.

[122] A. G. Marques, L. M. Lopez-Ramos, G. B. Giannakis, J. Ramos, and A. J. Caamaño, "Optimal cross-layer resource allocation in cellular networks using channel- and queue-state information," *IEEE Trans. Veh. Technol.*, vol. 61, no. 6, pp. 2789–2807, July 2012.

[123] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficientlearning of deep networks from decentralized data," in *Proc. Intl. Conf. Artif. Intell. Stat.*, vol. 54, Fort Lauderdale, FL, USA, 20–22 Apr. 2017, pp. 1273–1282.

[124] H. M. Merrill and F. C. Schweppe, "Bad data suppression in power system static state estimation," *IEEE Trans. Power App. Syst.*, vol. PAS-90, no. 6, pp. 2718–2725, Nov. 1971.

[125] K. R. Mestav, J. Luengo-Rozas, and L. Tong, "Bayesian state estimation for unobservable distribution systems via deep learning," *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4910–4920, May 2019.

[126] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks," *Proc. IEEE*, pp. 1–32, 2020 (to appear).

[127] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[128] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[129] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[130] E. J. Msechu, S. I. Roumeliotis, A. Ribeiro, and G. B. Giannakis, "Decentralized quantized kalman filtering with scalable communication cost," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3727–3741, 2008.

[131] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[132] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks," in *Global Commun. Conf.*, Singapore, Dec. 4-8, 2017, pp. 1–7.

[133] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance Internet applications," *ACM SIGOPS Operating Syst. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[134] J. Ostrometzky, K. Berestizshevsky, A. Bernstein, and G. Zussman, "Physics-informed deep neural network method for limited observability state estimation," *arXiv:1910.06401*, 2019.

[135] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[136] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Math. Oper. Res.*, vol. 12, no. 3, pp. 441–450, 1987.

[137] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. Conf. Comput. Commun. Sec.*, 2017, pp. 506–519.

[138] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, Aug. 2016.

[139] G. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in *Proc. of INFOCOM Conf.*, Paris, France, April 2019, pp. 545–549.

[140] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 836–845, Apr. 2016.

[141] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Intl. Conf. on Computer Communications*, Toronto, Canada, Apr. 2014, pp. 1078–1086.

[142] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug 2018.

[143] B. A. Robbins, H. Zhu, and A. D. Domínguez-García, "Optimal tap setting of voltage regulation transformers in unbalanced distribution systems," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 256–267, Feb. 2016.

[144] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*. Springer Science & Business Media, 2009, vol. 317.

[145] A. Romei and S. Ruggieri, "A multidisciplinary survey on discrimination analysis," *The Knowledge Engineering Review*, vol. 29, no. 5, pp. 582–638, 2014.

[146] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[147] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Upper Saddle River, NJ, USA,: Prentice-Hall, 2010.

[148] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ, USA, 2010.

[149] A. Sadeghi, M. Ma, B. Li, and G. Giannakis, "Distributionally robust semi-supervised learning over graphs," in *Proc. of Intl. Conf. on Learning Represenations, Workshop on Responsible AI*, 2021.

[150] A. Sadeghi, A. G. Marques, and G. B. Giannakis, "Distributed network caching via dynamic programming," in *Intl. Conf. on Acoustics, Speech, and Signal Process.*, Brighton, UK, 12–17 May 2019, pp. 4574–4578.

[151] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.

[152] A. Sadeghi, G. B. Giannakis, G. Wang, and F. Sheikholeslami, "Reinforcement learning for caching with space-time popularity dynamics," *IET book "Edge Caching for Mobile Networks" edited by W. Chen and V. Poor*, 2021.

[153] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal dynamic caching via reinforcement learning," in *Signal Process. Adv. in Wireless Commun.*, 2018.

[154] A. Sadeghi, F. Sheikholeslami, A. G. Marques, and G. B. Giannakis, "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE J. Sel. Areas in Commun.*, vol. 37, no. 10, pp. 2267–2281, 2019.

[155] A. Sadeghi, F. Sheikholeslami, A. G. Matrques, and G. B. Giannakis, "Reinforcement learning for 5G caching with dynamic cost," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Process*, 2018, pp. 6653–6657.

[156] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1024–1033, 2019.

[157] ——, "Hierarchical caching via deep reinforcement learning," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Process*, 2020, pp. 3532–3536.

[158] ——, "Learning while respecting privacy and robustness to distributional uncertainties and adversarial data," *arXiv:2007.03724*, 2020.

[159] M. Salem, L. Talat, and H. Soliman, "Voltage control by tap-changing transformers for a radial distribution network," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 144, no. 6, pp. 517–520, Nov. 1997.

[160] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for dynamic MR image reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 2, pp. 491–503, Oct. 2017.

[161] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Advances in Neural Information Processing Systems*, 2018, pp. 5014–5026.

[162] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. Intl. Symp. on Wireless Communications Systems*, Barcelona, Spain, Aug. 2014, pp. 917–921.

[163] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *ACM SIGSAC Conf. on Comput. Commun. Security*, 2016, pp. 1528–1540.

[164] F. Sheikholeslami, S. Jain, and G. B. Giannakis, "Minimum uncertainty based detection of adversaries in deep neural networks," *arXiv:1904.02841*, 2019.

[165] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *arXiv:2006.03262*, 2020.

[166] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Proactive retention-aware caching with multi-path routing for wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1286–1299, Jun. 2018.

[167] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Proces. Mag.*, vol. 30, pp. 83–98, 2013.

[168] A. Sinha, H. Namkoong, and R. V. J. Duchi, "Certifying some distributional robustness with principled adversarial training," *Intl. Conf. Learn. Rep.*, 2017.

[169] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *Intl. Conf. Learn. Rep.*, Vancouver, BC, Canada, May 2018.

[170] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, June 2018.

[171] W. Su, J. Wang, and J. Roh, "Stochastic energy scheduling in microgrids with intermittent renewable energy resources," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1876–1883, July 2014.

[172] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1960–1971, Apr. 2019.

[173] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press, 2016.

[174] ——, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT press, 2018.

[175] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conf. Robot Learn.*, 2020, pp. 671–682.

[176] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE Intl. Conf. on Comput. Commun.* IEEE, 2016, pp. 1–9.

[177] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, Nov. 2013.

[178] D. A. Tziouvaras, P. McLaren, G. Alexander, D. Dawson, J. Esztergalyos, C. Fromen, M. Glinkowski, I. Hasenwinkle, M. Kezunovic, L. Kojovic *et al.*, "Mathematical models for current, voltage, and coupling capacitor voltage transformers," *IEEE Trans. Power Del.*, vol. 15, no. 1, pp. 62–72, Jan. 2000.

[179] C. Villani, *Optimal Transport: Old and New*. Berlin, HR: SSBM, 2008, vol. 338.

[180] G. Wang, G. B. Giannakis, and J. Chen, "Robust and scalable power system state estimation via composite optimization," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6137–6147, Nov. 2019.

[181] G. Wang, V. Kekatos, A. J. Conejo, and G. B. Giannakis, "Ergodic energy management leveraging resource variability in distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4765–4775, Nov. 2016.

[182] G. Wang, G. B. Giannakis, J. Chen, and J. Sun, "Distribution system state estimation: An overview of recent developments," *Front. Inform. Technol. Electron. Eng.*, vol. 20, no. 1, pp. 4–17, Jan. 2019.

[183] G. Wang, V. Kekatos, A. J. Conejo, and G. B. Giannakis, "Ergodic energy management leveraging resource variability in distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4765–4775, Nov. 2016.

[184] W. Wang, D. Niyato, P. Wang, and A. Leshem, "Decentralized caching for content delivery based on blockchain: A game theoretic perspective," *arXiv:1801.07604*, 2018.

[185] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[186] Z. Wang, L. Li, Y. Xu, H. Tian, and S. Cui, "Handover control in wireless systems via asynchronous multiuser deep reinforcement learning," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4296–4307, Dec. 2018.

[187] Z. Wang, Z. Yu, Q. Ling, D. Berberidis, and G. B. Giannakis, "Decentralized rls with data-adaptive censoring for regressions over large-scale networks," *IEEE Trans. Signal Process.*, vol. 66, no. 6, pp. 1634–1648, Mar. 2018.

[188] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[189] C. Watkins and P. Dayan, "Q-learning," *Mach. learn.*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[190] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor, "Performance analysis and optimization in privacy-preserving federated learning," *arXiv:2003.00229*, 2020.

[191] W. Wiesemann, D. Kuhn, and M. Sim, "Distributionally robust convex optimization," *Operations Research*, vol. 62, no. 6, pp. 1358–1376, 2014.

[192] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro, "Learning non-discriminatory predictors," *arXiv:1702.06081*, 2017.

[193] G. Wu, J. Sun, and L. Xiong, "Optimal switching attacks and countermeasures in cyber-physical systems," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 5, pp. 1–10, Jun. 2020.

[194] G. Wu, G. Wang, J. Sun, and J. Chen, "Optimal partial feedback attacks in cyber-physical power systems," *IEEE Trans. Autom. Control*, pp. 1–8, 2020, to be published; DOI: 10.1109/TAC.2020.2981915.

[195] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv:1609.08144*, 2016.

[196] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2020.

[197] H. Xu, A. D. Domínguez-García, and P. W. Sauer, "Optimal tap setting of voltage regulation transformers using batch reinforcement learning," *arXiv:1807.10997v2*, 2018.

[198] Y. Xu, R. Jin, and T. Yang, "Non-asymptotic analysis of stochastic methods for non-smooth non-convex regularized problems," in *Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 2626–2636.

[199] Z. Yan and Y. Xu, "Data-driven load frequency control for stochastic power systems: A deep reinforcement learning method with continuous action search," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1653–1656, Nov. 2018.

[200] Q. Yang, A. Sadeghi, G. Wang, G. B. Giannakis, and J. Sun, "A statistical learning approach to reactive power control in distribution systems," *arXiv:1910.13938*, 2019.

[201] ——, "Deep policy gradient for reactive power control in distribution systems," in *IEEE Intl. Conf. on Commun., Control, and Comput. Tech. for Smart Grids (SmartGridComm)*, 2020, pp. 1–6.

[202] ——, "Power system state estimation using Gauss-Newton unrolled neural networks with trainable priors," in *IEEE Intl. Conf. on Commun., Control, and Comput. Tech. for Smart Grids (SmartGridComm)*, 2020, pp. 1–6.

[203] ——, "Robust PSSE using graph neural networks for data-driven and topology-aware priors." 2020.

[204] Q. Yang, A. Sadeghi, G. Wang, and J. Sun, "Learning two-layer ReLU networks is nearly as easy as learning linear classifiers on separable data," *IEEE Trans. Signal Process.*, vol. 69, pp. 4416–4427, 2021.

[205] Q. Yang, G. Wang, A. Sadeghi, G. B. Giannakis, and J. Sun, "Two-timescale voltage control in distribution grids using deep reinforcement learning," *IEEE Trans. Smart Grid*, pp. 1–11, 2019.

[206] ——, "Two-timescale voltage regulation in distribution grids using deep reinforcement learning," in *Proc. of SmartGridComm*, Beijing, CN, Oct. 21-23, 2019, pp. 1–6.

[207] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64 533–64 539, Oct. 2018.

[208] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," in *Intl. Conf. on Comm.*, Kansas City, USA, May 20 - 24, 2018, pp. 1–7.

[209] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," *arXiv preprint arXiv:1507.05259*, 2015.

[210] A. S. Zamzam and N. D. Sidiropoulos, "Physics-aware neural networks for distribution system state estimation," *arXiv:1903.09669*, 2019.

[211] A. S. Zamzam, B. Yang, and N. D. Sidiropoulos, "Energy storage management via deep Q-networks," in *Proc. of PESGM*, Atlanta, GA, Aug. 4-8, 2019, pp. 1–7.

[212] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," in *Intl. Conf. Mach. Learn.*, 2013, pp. 325–333.

[213] B. Zhang, A. Dominguez-Garcia, and D. Tse, "A local control approach to voltage regulation in distribution networks," in *Proc. North American Power Symposium*, Manhattan, KS, 2013.

[214] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 4069–4077, Aug. 2019.

[215] Y. Zhang, M. Hong, E. Dall'Anese, S. V. Dhople, and Z. Xu, "Distributed controllers seeking AC optimal power flow solutions using ADMM," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4525–4537, Sept. 2018.

[216] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Conf. on Info. Sciences and Syst.*, Princeton, NJ, March 21–23, 2018, pp. 1–6.

[217] F. Zhou, Q. Yang, T. Zhong, D. Chen, and N. Zhang, "Variational graph neural networks for road traffic prediction in intelligent transportation systems," *IEEE Trans. Ind. Inf.*, pp. 2802–2812, 2020.

[218] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv:1812.08434*, 2018.

[219] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov. 2018.

[220] H. Zhu and G. B. Giannakis, "Power system nonlinear state estimation using distributed semidefinite programming," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 6, pp. 1039–1050, Jun. 2014.

[221] H. Zhu and H. J. Liu, "Fast local voltage control under limited reactive power: Optimality and stability analysis," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3794–3803, Dec. 2016.

[222] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power syst.*, vol. 26, pp. 12–19, 2010.

[223] D. Zügner, O. Borchert, A. Akbarnejad, and S. Guennemann, "Adversarial attacks on graph neural networks: Perturbations and their patterns," *ACM Trans. Knwl. Discov. from Data*, vol. 14, pp. 1–31, 2020.

# Appendix A

# Proofs for Chapter 2

### A.0.1 Proof of Lemma 1

Since function $\boldsymbol{\zeta} \mapsto \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z})$ is $\lambda$-strongly concave, then $\boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}) = \sup_{\boldsymbol{\zeta} \in \mathcal{Z}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z})$ is unique. In addition, the first-order optimality condition gives $\langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}); \boldsymbol{z}), \boldsymbol{\zeta} - \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}) \rangle \leq 0$. Let us define $\boldsymbol{\zeta}_*^1 = \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}_1)$, $\boldsymbol{\zeta}_*^2 = \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}_2)$, and use the strong concavity for any $\bar{\boldsymbol{\theta}}_1$ and $\bar{\boldsymbol{\theta}}_2$, to write

$$\begin{aligned} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^2; \boldsymbol{z}) \leq \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) + \langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}), \boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1 \rangle \\ - \frac{\lambda}{2} \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\|^2 \end{aligned} \tag{A.1}$$

and

$$\begin{aligned} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) &\leq \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^2; \boldsymbol{z}) + \langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^2; \boldsymbol{z}), \boldsymbol{\zeta}_*^1 - \boldsymbol{\zeta}_*^2 \rangle \\ &\quad - \frac{\lambda}{2} \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\|^2 \\ &\leq \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^2; \boldsymbol{z}) - \frac{\lambda}{2} \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\|^2 \end{aligned} \tag{A.2}$$

where the last inequality is a consequence of the first-order optimality condition. Summing (A.1) and (A.2), we find that

$$\begin{aligned} \lambda \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\|^2 &\leq \langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}), \boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1 \rangle \\ &\leq \langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}), \boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1 \rangle \\ &\quad - \langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z}), \boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1 \rangle \end{aligned} \tag{A.3}$$

$$= \langle \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) - \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z}), \boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1 \rangle. \tag{A.4}$$

And using Hölder's inequality, we obtain that

$$\begin{aligned} \lambda \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\|^2 \leq \|\nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) - \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z})\|_\star \\ \times \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\| \end{aligned} \tag{A.5}$$

from which we deduce

$$\|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\| \leq \frac{1}{\lambda} \|\nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) - \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z})\|_\star. \tag{A.6}$$

Using $\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z}) := \ell(\boldsymbol{\theta}; \boldsymbol{\zeta}) + \gamma(\rho - c(\boldsymbol{z}, \boldsymbol{\zeta}))$, we have that

$$\begin{aligned} &\|\nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) - \nabla_{\boldsymbol{\zeta}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z})\|_\star \\ &= \|\nabla_{\boldsymbol{\zeta}} \ell(\boldsymbol{\theta}_2; \boldsymbol{\zeta}_*^1) - \nabla_{\boldsymbol{\zeta}} \ell(\boldsymbol{\theta}_1; \boldsymbol{\zeta}_*^1) \qquad \text{(A.7)} \\ &\quad + \gamma_1 \nabla_{\boldsymbol{\zeta}} c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1) - \gamma_2 \nabla_{\boldsymbol{\zeta}} c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1)\|_\star \\ &\leq \|\nabla_{\boldsymbol{\zeta}} \ell(\boldsymbol{\theta}_2; \boldsymbol{\zeta}_*^1) - \nabla_{\boldsymbol{\zeta}} \ell(\boldsymbol{\theta}_1; \boldsymbol{\zeta}_*^1)\|_\star \qquad \text{(A.8)} \\ &\quad + \|\gamma_1 \nabla_{\boldsymbol{\zeta}} c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1) - \gamma_2 \nabla_{\boldsymbol{\zeta}} c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1)\|_\star \\ &\leq L_{\boldsymbol{z}\boldsymbol{\theta}} \|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\| + \|\nabla_{\boldsymbol{\zeta}} c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1)\|_\star \|\gamma_2 - \gamma_1\|. \qquad \text{(A.9)} \end{aligned}$$

Substituting (A.9) into (A.6), yields

$$\begin{aligned} \|\boldsymbol{\zeta}_*^2 - \boldsymbol{\zeta}_*^1\| &\leq \frac{L_{\boldsymbol{z}\boldsymbol{\theta}}}{\lambda} \|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\| + \frac{1}{\lambda} \|\nabla_{\boldsymbol{\zeta}} c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1)\|_\star \|\gamma_2 - \gamma_1\| \\ &\leq \frac{L_{\boldsymbol{z}\boldsymbol{\theta}}}{\lambda} \|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\| + \frac{L_c}{\lambda} \|\gamma_2 - \gamma_1\| \end{aligned} \tag{A.10}$$

where the last inequality holds because $\boldsymbol{\zeta} \mapsto c(\boldsymbol{z}, \boldsymbol{\zeta})$ is $L_c$-Lipschitz as per Assumption 3.

To obtain (2.10c), we first suppose without loss of generality that only a single datum $\boldsymbol{z}$ is given, and in order to prove existence of the gradient of $\bar{\psi}(\bar{\boldsymbol{\theta}}, \boldsymbol{z})$ with respect to $\bar{\boldsymbol{\theta}}$, we resort to the Danskin's theorem as follows.

*Danskin's Theorem [42].* Consider the following minimax optimization problem

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\zeta} \in \mathcal{X}} f(\boldsymbol{\theta}, \boldsymbol{\zeta}) \tag{A.11}$$

where $\mathcal{X}$ is a nonempty compact set, and $f : \Theta \times \mathcal{X} \to [0, \infty)$ is such that $f(\cdot, \boldsymbol{\zeta})$ is differentiable for any $\boldsymbol{\zeta} \in \mathcal{X}$, and $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\zeta})$ is continuous on $\Theta \times \mathcal{X}$. With $\mathcal{S}(\boldsymbol{\theta}) := \{\boldsymbol{\zeta}_* | \boldsymbol{\zeta}_* = \arg\max_{\boldsymbol{\zeta}} f(\boldsymbol{\theta}, \boldsymbol{\zeta})\}$, the function

$$\bar{f}(\boldsymbol{\theta}) := \max_{\boldsymbol{\zeta} \in \mathcal{Z}} f(\boldsymbol{\theta}, \boldsymbol{\zeta})$$

is locally Lipschitz and directionally differentiable, where the directional derivatives satisfy

$$\bar{f}(\boldsymbol{\theta}, \boldsymbol{d}) = \sup_{\boldsymbol{\zeta} \in \mathcal{S}(\boldsymbol{\theta})} \langle \boldsymbol{d}, \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\zeta}) \rangle. \tag{A.12}$$

For a given $\boldsymbol{\theta}$, if the set $\mathcal{S}(\boldsymbol{\theta})$ is a singleton, then the function $\bar{f}(\boldsymbol{\theta})$ is differentiable at $\boldsymbol{\theta}$ with gradient

$$\nabla_{\boldsymbol{\theta}} \bar{f}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\zeta}_*(\boldsymbol{\theta})). \tag{A.13}$$

Given $\boldsymbol{\theta}$, and the $\mu$-strongly convex $c(\boldsymbol{z}, \cdot)$, function $\psi(\bar{\boldsymbol{\theta}}, \cdot; \boldsymbol{z})$ is concave if $L_{\boldsymbol{zz}} - \gamma\mu < 0$, which holds true for $\gamma_0 > L_{\boldsymbol{zz}}/\mu$. Replacing $\bar{f}(\boldsymbol{\theta}, \boldsymbol{\zeta})$ with $\psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z})$, and given the concavity of $\boldsymbol{\zeta} \mapsto \psi(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}; \boldsymbol{z})$, we have that $\bar{\psi}(\bar{\boldsymbol{\theta}}; \boldsymbol{z})$ is a continuous function with gradient

$$\nabla_{\bar{\boldsymbol{\theta}}} \bar{\psi}(\bar{\boldsymbol{\theta}}; \boldsymbol{z}) = \nabla_{\bar{\boldsymbol{\theta}}} \bar{\psi}(\bar{\boldsymbol{\theta}}, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}; \boldsymbol{z}); \boldsymbol{z}). \tag{A.14}$$

We can then obtain the second inequality, as

$$\begin{aligned}
&\|\nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) - \nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^2; \boldsymbol{z})\| \\
&\leq \|\nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^1; \boldsymbol{z}) - \nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^2; \boldsymbol{z})\| \\
&\qquad + \|\nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}_1, \boldsymbol{\zeta}_*^2; \boldsymbol{z}) - \nabla_{\bar{\boldsymbol{\theta}}} \psi(\bar{\boldsymbol{\theta}}_2, \boldsymbol{\zeta}_*^2; \boldsymbol{z})\| \qquad \text{(A.15)} \\
&\leq \left\| \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_1, \boldsymbol{\zeta}_*^1) - \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_1, \boldsymbol{\zeta}_*^2) \\ c(\boldsymbol{z}, \boldsymbol{\zeta}_*^2) - c(\boldsymbol{z}, \boldsymbol{\zeta}_*^1) \end{bmatrix} \right\| \\
&\qquad + \left\| \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_1, \boldsymbol{\zeta}_*^2) - \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_2, \boldsymbol{\zeta}_*^2) \\ 0 \end{bmatrix} \right\| \qquad \text{(A.16)} \\
&\leq L_{\boldsymbol{\theta z}} \|\boldsymbol{\zeta}_*^1 - \boldsymbol{\zeta}_*^2\| + L_c \|\boldsymbol{\zeta}_*^1 - \boldsymbol{\zeta}_*^2\| + L_{\boldsymbol{\theta\theta}} \|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\| \\
&\leq \Big( L_{\boldsymbol{\theta\theta}} + \frac{L_{\boldsymbol{\theta}z} L_{\boldsymbol{z\theta}} + L_c L_{\boldsymbol{z\theta}}}{\lambda} \Big) \|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\|
\end{aligned}$$

$$+ \frac{L_{\boldsymbol{\theta}z} L_c + L_c^2}{\lambda} \|\gamma_2 - \gamma_1\| \tag{A.17}$$

where we again used inequality (A.10). As a technical note, if the considered model is a neural network with a non-smooth activation function, the loss will not be continuously differentiable. However, we will not encounter this challenge often in practice.

### A.0.2 Proof of Theorem 1

With slight abuse of notation, define for convenience $F(\boldsymbol{\theta}, \gamma) := f(\boldsymbol{\theta}, \gamma) + r(\boldsymbol{\theta}) + h(\gamma)$, where $h(\gamma)$ is the indicator function

$$h(\gamma) = \begin{cases} 0, & \text{if } \gamma \in \Gamma \\ \infty, & \text{if } \gamma \notin \Gamma \end{cases} \tag{A.18}$$

with $\Gamma := \{\gamma | \gamma \geq \gamma_0\}$, and for ease of representation we use $\bar{r}(\bar{\boldsymbol{\theta}}) := r(\boldsymbol{\theta}) + h(\gamma)$. Having an $L_f$–smooth function $f$, yields

$$f(\bar{\boldsymbol{\theta}}^{t+1}) \leq f(\bar{\boldsymbol{\theta}}^t) + \left\langle \nabla f(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \right\rangle + \frac{L_f}{2} \|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2. \tag{A.19}$$

For a given $\boldsymbol{z}^t$, the gradients are

$$\begin{aligned} \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t) &:= \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \\ \partial_\gamma \psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \end{bmatrix} \\ &= \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \\ \rho - c(\boldsymbol{z}^t, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t) \end{bmatrix}. \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t) &:= \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \\ \partial_\gamma \psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \end{bmatrix} \\ &= \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \\ \rho - c(\boldsymbol{z}^t, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t) \end{bmatrix} \end{aligned}$$

obtained by an oracle at the optimal $\boldsymbol{\zeta}_*$ and the $\epsilon$-optimal $\boldsymbol{\zeta}_\epsilon$ solvers, respectively. Now, we define the error vector $\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t) := \nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)$, and replace this into (A.19), to obtain

$$f(\bar{\boldsymbol{\theta}}^{t+1}) \leq f(\bar{\boldsymbol{\theta}}^t) + \big\langle \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t) + \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle + \frac{L_f}{2}\big\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\big\|^2. \tag{A.20}$$

The following properties hold equivalently for the proximal operator, and for any $\boldsymbol{x}, \boldsymbol{y}$

$$\boldsymbol{u} = \text{prox}_{\alpha r}(\boldsymbol{x}) \iff \big\langle \boldsymbol{x} - \boldsymbol{u}, \boldsymbol{y} - \boldsymbol{u} \big\rangle \leq \alpha r(\boldsymbol{y}) - \alpha r(\boldsymbol{u}). \tag{A.21}$$

With $\boldsymbol{u} = \bar{\boldsymbol{\theta}}^{t+1}$ and $\boldsymbol{x} = \bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)$ in (A.21), it holds that

$$\big\langle \bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t) - \bar{\boldsymbol{\theta}}^{t+1}, \bar{\boldsymbol{\theta}}^t - \bar{\boldsymbol{\theta}}^{t+1} \big\rangle \leq \alpha_t \bar{r}(\bar{\boldsymbol{\theta}}^t) - \alpha_t \bar{r}(\bar{\boldsymbol{\theta}}^{t+1})$$

and upon rearranging, we obtain

$$\big\langle \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle \leq \bar{r}(\bar{\boldsymbol{\theta}}^t) - \bar{r}(\bar{\boldsymbol{\theta}}^{t+1}) - \frac{1}{\alpha_t}\big\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\big\|^2. \tag{A.22}$$

Adding inequalities in (A.22) and (A.20) gives

$$f(\bar{\boldsymbol{\theta}}^{t+1}) \leq f(\bar{\boldsymbol{\theta}}^t) + \big\langle \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle + \frac{L_f}{2}\big\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\big\|^2 + \bar{r}(\bar{\boldsymbol{\theta}}^t) - \bar{r}(\bar{\boldsymbol{\theta}}^{t+1}) - \frac{1}{\alpha_t}\big\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\big\|^2$$

and with $F(\bar{\boldsymbol{\theta}}) := f(\bar{\boldsymbol{\theta}}) + \bar{r}(\bar{\boldsymbol{\theta}})$, we can write

$$F(\bar{\boldsymbol{\theta}}^{t+1}) - F(\bar{\boldsymbol{\theta}}^t) \leq \big\langle \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle + \Big(\frac{L_f}{2} - \frac{1}{\alpha_t}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\big\|^2. \tag{A.23}$$

Using Young's inequality for any $\eta > 0$ gives $\big\langle \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle \leq \frac{\eta}{2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2 + \frac{1}{2\eta}\|\delta(\bar{\boldsymbol{\theta}}^t)\|^2$, and hence

$$F(\bar{\boldsymbol{\theta}}^{t+1}) - F(\bar{\boldsymbol{\theta}}^t) \leq \Big(\frac{L_f + \eta}{2} - \frac{1}{\alpha_t}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\big\|^2 + \frac{\big\|\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t)\big\|^2}{2\eta}. \tag{A.24}$$

Next, we will bound $\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t) := \nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)$. By adding and subtracting $\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)$ to the right hand side, we find

$$\|\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t)\|^2 \leq 2\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\|^2 + 2\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)\|^2. \tag{A.25}$$

The Lipschitz smoothness of the gradient, implies that

$$\begin{aligned}
&\left\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)\right\|^2 \qquad (A.26)\\
&= \left\| \begin{bmatrix} \nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \\ \rho - c(\boldsymbol{z}^t, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t) \end{bmatrix} - \begin{bmatrix} \nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) \\ \rho - c(\boldsymbol{z}^t, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t) \end{bmatrix} \right\|^2 \\
&= \left\|\nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_*(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t) - \nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}^t, \gamma, \boldsymbol{\zeta}_\epsilon(\bar{\boldsymbol{\theta}}^t; \boldsymbol{z}^t); \boldsymbol{z}^t)\right\|^2 \\
&\quad + \left\|c(\boldsymbol{z}^t, \boldsymbol{\zeta}_*^t) - c(\boldsymbol{z}^t, \boldsymbol{\zeta}_\epsilon^t)\right\|^2 \\
&\overset{(a)}{\leq} \Big(\frac{L_{\boldsymbol{\theta z}}^2}{\lambda^t} + L_c\Big)\|\boldsymbol{\zeta}_*^t - \boldsymbol{\zeta}_\epsilon^t\|^2 \\
&\overset{(b)}{\leq} \Big(\frac{L_{\boldsymbol{\theta z}}^2}{\lambda^t} + L_c\Big)\epsilon \\
&\leq \Big(\frac{L_{\boldsymbol{\theta z}}^2}{\lambda_0} + L_c\Big)\epsilon \qquad (A.27)
\end{aligned}$$

where (a) uses the $\lambda^t = \mu\gamma^t - L_{\boldsymbol{zz}}$ strong-concavity of $\boldsymbol{\zeta} \mapsto \psi(\bar{\boldsymbol{\theta}}, \gamma, \boldsymbol{\zeta}; \boldsymbol{z})$, and the second term is bounded by $L_c\|\boldsymbol{\zeta}_*^t - \boldsymbol{\zeta}_\epsilon^t\|^2$ according to Assumption 3. The last inequality holds for $\lambda_0 := \mu\gamma_0 - L_{\boldsymbol{zz}}$, where we used (A.18) to bound $\gamma^t \geq \gamma_0 > L_{\boldsymbol{zz}}$. So far, we have established that

$$\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)\|^2 \leq \frac{L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2\epsilon}{\lambda_0} \tag{A.28}$$

where for notational convenience we let $L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2 := L_{\boldsymbol{\theta z}}^2 + \lambda_0 L_c$. Substituting (A.28) into (A.25), the error can be bounded as

$$\|\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t)\|^2 \leq 2\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\|^2 + \frac{2L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2\epsilon}{\lambda_0}. \tag{A.29}$$

Combining (A.24) and (A.27) yields

$$F(\bar{\boldsymbol{\theta}}^{t+1}) - F(\bar{\boldsymbol{\theta}}^t) \leq \Big(\frac{L_f + \eta}{2} - \frac{1}{\alpha_t}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2 \tag{A.30}$$

$$+\frac{1}{\eta}\big\|\nabla f(\bar{\boldsymbol{\theta}}^t)-\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\big\|^2+\frac{L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2\epsilon}{\eta\lambda_0}.$$

Considering a constant step size $\alpha$ and summing these inequalities over $t=1,\dots,T$ yields

$$\begin{aligned}\Big(\frac{1}{\alpha}-\frac{L_f+\eta}{2}\Big)\sum_{t=0}^{T}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\big\|^2 &\le F(\bar{\boldsymbol{\theta}}^0)-F(\bar{\boldsymbol{\theta}}^T)\\ &+\frac{1}{\eta}\sum_{t=0}^{T}\big\|\nabla f(\bar{\boldsymbol{\theta}}^t)-\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\big\|^2+\frac{(T+1)L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2\epsilon}{\lambda_0}.\end{aligned}\tag{A.31}$$

From the proximal gradient update

$$\bar{\boldsymbol{\theta}}^{t+1}=\arg\min_{\boldsymbol{\theta}}\ \alpha\bar{r}(\boldsymbol{\theta})+\alpha\big\langle\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}^t,\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)\big\rangle+\frac{1}{2}\big\|\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}^t\big\|^2\tag{A.32}$$

the optimality of $\bar{\boldsymbol{\theta}}^{t+1}$ in (A.32), implies that

$$\bar{r}(\bar{\boldsymbol{\theta}}^{t+1})+\big\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t,\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)\big\rangle+\frac{1}{2\alpha}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\big\|^2\le\bar{r}(\bar{\boldsymbol{\theta}}^t)$$

which combined with the smoothness of $f$ (c.f. (A.19)) yields

$$\begin{aligned}&\big\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t,\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\big\rangle+\Big(\frac{1}{2\alpha}-\frac{L_f}{2}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\big\|^2\\ &\quad\le F(\bar{\boldsymbol{\theta}}^t)-F(\bar{\boldsymbol{\theta}}^{t+1})\end{aligned}$$

Subtracting $\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t,\nabla f(\bar{\boldsymbol{\theta}}^{t+1})\rangle$ from both sides gives

$$\begin{aligned}&\big\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t,\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})\big\rangle+\Big(\frac{1}{2\alpha}-\frac{L_f}{2}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\big\|^2\\ &\quad\le F(\bar{\boldsymbol{\theta}}^t)-F(\bar{\boldsymbol{\theta}}^{t+1})-\big\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t,\nabla f(\bar{\boldsymbol{\theta}}^{t+1})-\nabla f(\bar{\boldsymbol{\theta}}^t)\big\rangle.\end{aligned}$$

Considering $\big\|\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t)\big\|^2$ on the left hand side, and adding relevant terms to the right hand side, we arrive at

$$\begin{aligned}&\Big\|\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t)\Big\|^2\\ &\le\big\|\boldsymbol{g}^\epsilon(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})\big\|^2+\frac{1}{\alpha^2}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\big\|^2\end{aligned}\tag{A.33}$$

$$
\begin{aligned}
&+\Big(\frac{L_f}{\alpha}-\frac{1}{\alpha^2}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2+\frac{2}{\alpha}\big(F(\bar{\boldsymbol{\theta}}^{t})-F(\bar{\boldsymbol{\theta}}^{t+1})\big)\\
&-\frac{2}{\alpha}\big\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t},\nabla f(\bar{\boldsymbol{\theta}}^{t+1})-\nabla f(\bar{\boldsymbol{\theta}}^{t})\big\rangle\\
\leq&\big\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^{t})-\nabla f(\bar{\boldsymbol{\theta}}^{t})\|^2+\frac{1}{\alpha^2}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2 \qquad \text{(A.34)}\\
&+\Big(\frac{L_f}{\alpha}-\frac{1}{\alpha^2}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2+\frac{2}{\alpha}\big(F(\bar{\boldsymbol{\theta}}^{t})-F(\bar{\boldsymbol{\theta}}^{t+1})\big)\\
&-\frac{2}{\alpha}\big\langle\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}),\nabla f(\bar{\boldsymbol{\theta}}^{t+1})-\nabla f(\bar{\boldsymbol{\theta}}^{t})\big\rangle\\
\leq&\big\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^{t})-\nabla f(\bar{\boldsymbol{\theta}}^{t})\|^2+\frac{1}{\alpha^2}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2 \qquad \text{(A.35)}\\
&+\Big(\frac{L_f}{\alpha}-\frac{1}{\alpha^2}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2+\frac{2}{\alpha}\big(F(\bar{\boldsymbol{\theta}}^{t})-F(\bar{\boldsymbol{\theta}}^{t+1})\big)\\
&+\frac{\eta}{\alpha}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2+\frac{L_f^2}{\eta}\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2
\end{aligned}
$$

where the last inequality is obtained by applying Young's inequality, and then using the $L_f$-Lipschitz continuity of $f(\cdot)$. By simplifying the last inequality, we obtain

$$
\begin{aligned}
&\Big\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^{t})-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t})\Big\|^2\leq\big\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^{t})-\nabla f(\bar{\boldsymbol{\theta}}^{t})\big\|^2\\
&+\frac{2}{\alpha}\big(F(\bar{\boldsymbol{\theta}}^{t})-F(\bar{\boldsymbol{\theta}}^{t+1})\big)+\Big(\frac{L_f^2}{\eta}+\frac{L_f+\eta}{\alpha}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2.
\end{aligned}
$$

The first term in the right hand side can be bounded by adding and subtracting $\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)$ and using (A.28), to arrive at

$$
\begin{aligned}
&\Big\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^{t})-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t})\Big\|^2\\
&\leq 2\big\|\nabla f(\bar{\boldsymbol{\theta}}^{t})-\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^{t})\big\|^2\frac{2L_{\bar{\boldsymbol{\theta}}}^2\epsilon}{\lambda_0}+\frac{2}{\alpha}\big(F(\bar{\boldsymbol{\theta}}^{t})-F(\bar{\boldsymbol{\theta}}^{t+1})\big)\\
&\quad+\Big(\frac{L_f^2}{\eta}+\frac{L_f+\eta}{\alpha}\Big)\big\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t}\big\|^2. \qquad \text{(A.36)}
\end{aligned}
$$

Summing these inequalities over $t=1,\ldots,T$, we find

$$
\sum_{t=0}^{T}\Big\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^{t})-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^{t})\Big\|^2
$$

$$\leq \sum_{t=0}^{T} \left\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\right\|^2 + \frac{2(T+1)L_{\bar{\boldsymbol{\theta}}}^2\epsilon}{\lambda_0} \tag{A.37}$$
$$+ \frac{2}{\alpha}\big[F(\bar{\boldsymbol{\theta}}^0) - F(\bar{\boldsymbol{\theta}}^T)\big] + \Big(\frac{L_f^2}{\eta} + \frac{L_f+\eta}{\alpha}\Big)\sum_{t=0}^{T}\left\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\right\|^2.$$

Using (A.31) to bound the last term yields

$$\sum_{t=0}^{T}\left\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^t) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) + \frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t)\right\|^2$$
$$\leq 2\sum_{t=0}^{T}\left\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\right\|^2 + \frac{2(T+1)L_{\bar{\boldsymbol{\theta}}}^2\epsilon}{\lambda_0} + \frac{2}{\alpha}\Delta_F + \beta\Delta_F$$
$$+ \frac{\beta}{\eta}\sum_{t=0}^{T}\left\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\right\|^2 + \frac{\beta(T+1)L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2\epsilon}{\lambda_0} \tag{A.38}$$

where $\beta = (\frac{L_f^2}{\eta} + \frac{L_f+\eta}{\alpha})\frac{2\alpha}{2-(L_f+\eta)\alpha}$. By taking expectation of both sides of this inequality, we obtain

$$\frac{1}{T+1}\mathbb{E}\Big[\sum_{t=0}^{T}\left\|\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^t) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) + \frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t)\right\|^2\Big]$$
$$\leq \Big(\frac{2}{\alpha} + \beta\Big)\frac{\Delta_F}{T+1} + \Big(\frac{\beta}{\eta} + 2\Big)\sigma^2 + \frac{(\beta+2)L_{\bar{\boldsymbol{\theta}}}^2\epsilon}{\lambda_0} \tag{A.39}$$

where we have used $\mathbb{E}[\|\nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\|_2^2] \leq \sigma^2$, which holds according to Assumption 5. By [144, Theorem 10] and [198], we know that

$$-\boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^t) - \frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t) \in \partial\bar{r}(\bar{\boldsymbol{\theta}}^{t+1}) \tag{A.40}$$

which gives

$$\nabla f(\bar{\boldsymbol{\theta}}^{t+1}) - \boldsymbol{g}^{\epsilon}(\bar{\boldsymbol{\theta}}^t) - \frac{1}{\alpha}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t) \in \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) + \partial\bar{r}(\bar{\boldsymbol{\theta}}^{t+1})$$
$$:= \partial F(\bar{\boldsymbol{\theta}}^{t+1}).$$

Upon replacing the latter in the left hand side of (A.39), and recalling the definition of distance,

we deduce that

$$\mathbb{E}\big[\text{dist}(\mathbf{0}, \partial \hat{F}(\bar{\boldsymbol{\theta}}^{t'}))\big] \leq \big(\frac{2}{\alpha}+\beta\big)\frac{\Delta_F}{T}+\big(\frac{\beta}{\eta}+2\big)\sigma^2+\frac{(\beta+2)L_{\bar{\boldsymbol{\theta}}_z}^2\epsilon}{\lambda_0}$$

where $t'$ is randomly drawn from $t' \in \{1, 2, \ldots, T+1\}$, which concludes the proof.

### A.0.3 Proof of Theorem 2

Instead of resorting to an oracle to obtain an $\epsilon$-optimal solver for the surrogate loss, here we utilize a single step stochastic gradient ascent with mini-batch size $M$ to solve the maximization step. Consequently, the updates become

$$\bar{\boldsymbol{\theta}}^{t+1} = \text{prox}_{\alpha_t r}\big(\bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)\big) \tag{A.41}$$

where $\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) := \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{g}(\bar{\boldsymbol{\theta}}^t, \boldsymbol{\zeta}_m^t; \boldsymbol{z}_m)$. Letting $\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t) := \nabla f(\bar{\boldsymbol{\theta}}^t) - \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)$, and using the $L_f$-smoothness of $f(\bar{\boldsymbol{\theta}})$, we obtain

$$\begin{aligned} f(\bar{\boldsymbol{\theta}}^{t+1}) &\leq f(\bar{\boldsymbol{\theta}}^t) + \big\langle \nabla f(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle + \frac{L_f}{2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2 \\ &\leq f(\bar{\boldsymbol{\theta}}^t) + \big\langle \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) + \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle + \frac{L_f}{2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2. \end{aligned} \tag{A.42}$$

Next, we substitute $\bar{\boldsymbol{\theta}}^{t+1} \to \boldsymbol{u}$, $\boldsymbol{\theta}^t \to \boldsymbol{y}$, and $\bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) \to \boldsymbol{x}$ in (A.21), to arrive at

$$\big\langle \bar{\boldsymbol{\theta}}^t - \alpha_t \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) - \bar{\boldsymbol{\theta}}^{t+1}, \bar{\boldsymbol{\theta}}^t - \bar{\boldsymbol{\theta}}^{t+1} \big\rangle \leq \alpha_t \bar{r}(\bar{\boldsymbol{\theta}}^t) - \alpha_t \bar{r}(\bar{\boldsymbol{\theta}}^{t+1})$$

which leads to

$$\big\langle \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle \leq \bar{r}(\bar{\boldsymbol{\theta}}^t) - \bar{r}(\bar{\boldsymbol{\theta}}^{t+1}) - \frac{1}{\alpha_t}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2.$$

Substituting the latter into (A.42), gives

$$\begin{aligned} f(\bar{\boldsymbol{\theta}}^{t+1}) \leq f(\bar{\boldsymbol{\theta}}^t) + \big\langle \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t \big\rangle + \frac{L_f}{2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2 \\ + \bar{r}(\bar{\boldsymbol{\theta}}^t) - \bar{r}(\bar{\boldsymbol{\theta}}^{t+1}) - \frac{1}{\alpha_t}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t\|^2 \end{aligned}$$

and with $F(\boldsymbol{\theta}) := f(\boldsymbol{\theta}) + \bar{r}(\boldsymbol{\theta})$, we have

$$F(\bar{\boldsymbol{\theta}}^{t+1}) - F(\bar{\boldsymbol{\theta}}^{t}) \leq \big\langle \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^{t}), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t} \big\rangle + \Big(\frac{L_f}{2} - \frac{1}{\alpha_t}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2. \tag{A.43}$$

Using Young's inequality $\big\langle \boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^{t}), \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t} \big\rangle \leq \frac{1}{2}\|\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^{t})\|^2 + \frac{1}{2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2$ implies that

$$F(\bar{\boldsymbol{\theta}}^{t+1}) - F(\bar{\boldsymbol{\theta}}^{t}) \leq \Big(\frac{L_f + 1}{2} - \frac{1}{\alpha_t}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 + \frac{\|\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^{t})\|^2}{2} \tag{A.44}$$

and after adding the term $\big\langle \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}, \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) \big\rangle$ to both sides in (A.44), and simplifying terms, yields

$$\begin{aligned}
&\big\langle \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}, \boldsymbol{g}^{t}(\bar{\boldsymbol{\theta}}^{t}) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) \big\rangle \\
&\leq -\Big(\frac{1}{2\alpha_t} - \frac{L_f}{2}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 + F(\bar{\boldsymbol{\theta}}^{t}) - F(\bar{\boldsymbol{\theta}}^{t+1}) \\
&\quad - \big\langle \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}, \boldsymbol{\nabla} f(\bar{\boldsymbol{\theta}}^{t+1}) - \nabla f(\bar{\boldsymbol{\theta}}^{t}) \big\rangle.
\end{aligned} \tag{A.45}$$

Completing the square yields

$$\begin{aligned}
&\big\|\boldsymbol{g}^{t}(\bar{\boldsymbol{\theta}}^{t}) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) + \frac{1}{\alpha_t}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t})\big\|^2 \\
&\leq \|\boldsymbol{g}^{t}(\bar{\boldsymbol{\theta}}^{t}) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1})\|^2 + \frac{1}{\alpha_t^2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 \\
&\quad + \Big(\frac{L_f}{\alpha_t} - \frac{1}{\alpha_t^2}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 + \frac{2(F(\bar{\boldsymbol{\theta}}^{t}) - F(\bar{\boldsymbol{\theta}}^{t+1}))}{\alpha_t} \\
&\quad - \frac{2}{\alpha_t}\big\langle \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}, \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) - \nabla f(\bar{\boldsymbol{\theta}}^{t}) \big\rangle \\
&\leq 2\|\boldsymbol{g}^{t}(\bar{\boldsymbol{\theta}}^{t}) - \nabla f(\bar{\boldsymbol{\theta}}^{t})\|^2 + 2\|\nabla f(\bar{\boldsymbol{\theta}}^{t}) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1})\|^2 \\
&\quad + \frac{1}{\alpha_t^2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 + \Big(\frac{L_f}{\alpha_t} - \frac{1}{\alpha_t^2}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 \\
&\quad + \frac{2(F(\bar{\boldsymbol{\theta}}^{t}) - F(\bar{\boldsymbol{\theta}}^{t+1}))}{\alpha_t} - \frac{2}{\alpha_t}\langle \bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}, \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) - \nabla f(\bar{\boldsymbol{\theta}}^{t}) \rangle \\
&\leq 2\|\boldsymbol{g}^{t}(\bar{\boldsymbol{\theta}}^{t}) - \nabla f(\bar{\boldsymbol{\theta}}^{t})\|^2 + 2L_f^2\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 \\
&\quad + \frac{1}{\alpha_t^2}\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 + \Big(\frac{L_f}{\alpha_t} - \frac{1}{\alpha_t^2}\Big)\|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2
\end{aligned}$$

$$
\begin{aligned}
&+\frac{2(F(\bar{\boldsymbol{\theta}}^t)-F(\bar{\boldsymbol{\theta}}^{t+1}))}{\alpha_t}+\frac{2L_f}{\alpha_t}\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\|^2\\
\leq& 2\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2+\frac{2(F(\bar{\boldsymbol{\theta}}^t)-F(\bar{\boldsymbol{\theta}}^{t+1}))}{\alpha_t}+\\
&\frac{3L_f+2L_f^2\alpha_t}{\alpha_t}\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\|^2. \qquad \text{(A.46)}
\end{aligned}
$$

Recalling that $\boldsymbol{\delta}(\bar{\boldsymbol{\theta}}^t):=\nabla f(\bar{\boldsymbol{\theta}}^t)-\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)$, we can bound the first term as

$$
\begin{aligned}
&\mathbb{E}\Big[\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2\,\big|\boldsymbol{\theta}^t\Big]\\
=&\,\mathbb{E}\Big[\,\big\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)+\boldsymbol{\delta}^t\big\|^2\,\big|\boldsymbol{\theta}^t\Big]\\
=&\,\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2+\|\boldsymbol{\delta}^t\|^2+2\mathbb{E}\Big[\,\big\langle\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t),\boldsymbol{\delta}^t\big\rangle\,\big|\boldsymbol{\theta}^t\Big] \qquad \text{(A.47)}
\end{aligned}
$$

where the third equality is obtained by expanding the square term, and using $\mathbb{E}\big[\langle\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t),\boldsymbol{\delta}^t\rangle\big|\bar{\boldsymbol{\theta}}^t\big]=\mathbf{0}$. We will further bound the right hand side here as follows. Recalling that $\boldsymbol{\delta}^t=\frac{1}{M}\sum_{m=1}^M\boldsymbol{g}(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t;\boldsymbol{z}_m)-\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)$, where $\boldsymbol{g}^*(\boldsymbol{\theta}^t):=\frac{1}{M}\sum_{m=1}^M\nabla_{\bar{\boldsymbol{\theta}}}\psi(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^{*t};\boldsymbol{z}_m)$, it holds that

$$
\begin{aligned}
&\mathbb{E}\Big[\Big\|\frac{1}{M}\sum_{m=1}^M\Big[\boldsymbol{g}(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t;\boldsymbol{z}_m)-\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)\Big]\Big\|^2\Big|\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t\Big]\\
=&\,\frac{1}{M^2}\sum_{m=1}^M\mathbb{E}\Big[\big\|\nabla_{\bar{\boldsymbol{\theta}}}\psi(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t;\boldsymbol{z}_m)-\nabla_{\bar{\boldsymbol{\theta}}}\psi(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^{*t};\boldsymbol{z}_m)\big\|^2\big|\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t\Big]\\
\leq&\,\frac{L_{\boldsymbol{\theta z}}^2}{M^2}\sum_{m=1}^M\big\|\boldsymbol{\zeta}_m^t-\boldsymbol{\zeta}_m^{*t}\big\|^2 \qquad \text{(A.48)}
\end{aligned}
$$

where the second equality is because the samples $\{\boldsymbol{z}_m\}_{m=1}^M$ are i.i.d., and last inequality holds due to the Lipschitz smoothness of $\psi(\cdot)$. Since $\boldsymbol{\zeta}_m^t$ is obtained by a single gradient ascent update over a $\mu$-strongly concave function, we have that

$$
\frac{L_{\boldsymbol{\theta z}}^2}{M^2}\sum_{m=1}^M\big\|\boldsymbol{\zeta}_m^t-\boldsymbol{\zeta}_m^{*t}\big\|^2\leq\frac{L_{\boldsymbol{\theta z}}^2}{M}\Big[(1-\alpha_t\mu)\,D^2+\alpha_t^2B^2\Big] \qquad \text{(A.49)}
$$

where $D$ is the diameter of the feasible set, and $\alpha_t>0$ is the step size. The following holds for

the expected error term

$$\mathbb{E}\Big[\|\boldsymbol{\delta}^t\|^2\big|\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t\Big]\leq\frac{L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2}{M}\Big[(1-\alpha_t\mu)\,D^2+\alpha_t^2B^2\Big] \tag{A.50}$$

and using it in (A.47), we arrive at

$$\mathbb{E}\Big[\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2\big|\boldsymbol{\theta}^t\Big]\leq 2\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2 \tag{A.51}$$

$$+\frac{L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2}{M}\Big[(1-\alpha_t\mu)\,D^2+\alpha_t^2B^2\Big]. \tag{A.52}$$

Substituting the last inequality into (A.46) boils down to

$$\begin{aligned}&\mathbb{E}\Big[\big\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha_t}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t)\big\|^2\big|\bar{\boldsymbol{\theta}}^t\Big]\\&\leq 4\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2+\frac{3L_f+2L_f^2\alpha_t}{\alpha_t}\mathbb{E}\Big[\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\|^2\big|\bar{\boldsymbol{\theta}}^t\Big]\\&\quad+\frac{2F(\bar{\boldsymbol{\theta}}^t)-2\mathbb{E}\big[F(\bar{\boldsymbol{\theta}}^{t+1})\big|\bar{\boldsymbol{\theta}}^t\big]}{\alpha_t}+\frac{L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2}{M}\Big[(1-\alpha_t\mu)\,D^2+\alpha_t^2B^2\Big].\end{aligned} \tag{A.53}$$

Taking again expectation over $\bar{\boldsymbol{\theta}}^t$ on both sides, yields

$$\begin{aligned}&\mathbb{E}\Big\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha_t}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t)\Big\|^2\\&\leq 4\mathbb{E}\Big[\|\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2\Big]+\frac{L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2}{M}\Big[(1-\alpha_t\mu)\,D^2+\alpha_t^2B^2\Big]\\&\quad+\mathbb{E}\Big[\frac{2F(\bar{\boldsymbol{\theta}}^t)-2F(\bar{\boldsymbol{\theta}}^{t+1})}{\alpha_t}+\frac{3L_f+2L_f^2\alpha_t}{\alpha_t}\|\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t\|^2\Big].\end{aligned} \tag{A.54}$$

Recalling that $\mathbb{E}[\|\boldsymbol{\psi}^*(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^t;\boldsymbol{z}_m)-\nabla f(\bar{\boldsymbol{\theta}}^t)\|^2]\leq\sigma^2$, and that $\boldsymbol{g}^*(\bar{\boldsymbol{\theta}}^t)=\frac{1}{M}\sum_{m=1}^M\boldsymbol{\psi}(\bar{\boldsymbol{\theta}}^t,\boldsymbol{\zeta}_m^{*t};\boldsymbol{z}_m)$, the first term on the right hand side can be bounded by $\frac{4\sigma^2}{M}$. For a fixed learning rate $\alpha>0$, summing inequalities (A.54) from $t=0,\ldots,T$, yields

$$\begin{aligned}&\frac{1}{T+1}\mathbb{E}\Big[\sum_{t=0}^T\big\|\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t)-\nabla f(\bar{\boldsymbol{\theta}}^{t+1})+\frac{1}{\alpha_t}(\bar{\boldsymbol{\theta}}^{t+1}-\bar{\boldsymbol{\theta}}^t)\big\|^2\Big]\\&\leq\frac{2}{\alpha(T+1)}\big(F(\boldsymbol{\theta}^{\mathbf{0}})-\mathbb{E}[F(\boldsymbol{\theta}^T)]\big)+\frac{2L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2}{M}[(1-\alpha\mu)D^2+\alpha^2B^2]\end{aligned}$$

$$
\begin{aligned}
&+ \frac{3L_f + 2L_f^2\alpha}{\alpha} \frac{1}{T+1} \mathbb{E}\Big[ \sum_{t=0}^{T} \|\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^{t}\|^2 \Big] + \frac{4\sigma^2}{M} \\
\leq & \frac{1}{T+1} \left\{ \frac{2}{\alpha} + \frac{6L_f + 4L_f^2\alpha}{[2 - \alpha(L_f + \beta)]} \right\} (F(\bar{\boldsymbol{\theta}}^0) - \mathbb{E}[F(\bar{\boldsymbol{\theta}}^T)]) + \frac{4\sigma^2}{M} \\
&+ \frac{2L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2}{M} \left\{ 1 + \frac{3L_f + 2L_f^2\alpha}{2(2 - \alpha(L_f + \beta))} \right\} \Big[ (1 - \alpha\mu) D^2 + \alpha^2 B^2 \Big]. \qquad \text{(A.55)}
\end{aligned}
$$

Consider now replacing $F(\bar{\boldsymbol{\theta}}^0) - F(\bar{\boldsymbol{\theta}}^T)$ with $\Delta_F = F(\bar{\boldsymbol{\theta}}^0) - \inf_{\bar{\boldsymbol{\theta}}} \boldsymbol{F}(\bar{\boldsymbol{\theta}})$, and note that $\boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) + \frac{1}{\alpha_t}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t) \in \partial F(\bar{\boldsymbol{\theta}}^{t+1})$, where $\partial F$ denotes the set of subgradients of $F$. It then becomes clear that

$$
\begin{aligned}
& \mathbb{E}\big[\text{dist}(0, \partial F)^2\big] \\
\leq & \frac{1}{T+1} \mathbb{E}\Big[ \sum_{t=0}^{T} \Big\| \boldsymbol{g}^t(\bar{\boldsymbol{\theta}}^t) - \nabla f(\bar{\boldsymbol{\theta}}^{t+1}) + \frac{1}{\alpha_t}(\bar{\boldsymbol{\theta}}^{t+1} - \bar{\boldsymbol{\theta}}^t) \Big\|^2 \Big] \\
\leq & \frac{\zeta}{T+1} \Delta_F + \frac{2L_{\bar{\boldsymbol{\theta}}\boldsymbol{z}}^2 \nu}{N} \Big[ (1 - \alpha\mu) D^2 + \alpha^2 B^2 \Big] + \frac{4\sigma^2}{M}
\end{aligned}
$$

where $\zeta = \frac{2}{\alpha} + \frac{6L_f + 4L_f^2\alpha}{(2 - \alpha(L_f + \beta))}$ and $\nu = 1 + \frac{3L_f + 2L_f^2\alpha}{2(2 - \alpha(L_f + \beta))}$, which concludes the proof.