Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Automatic Control Laboratory

Master thesis

Learning opponents' behavior in receding horizon games

Leonardo Barcotto
October 30, 2023

Advisors
Dr. Dario Paccagnan, Prof. Dr. Florian Dörfler

Abstract

Receding horizon games are MPC-inspired games in which the strategies of the players are sequences of actions over a time horizon, and are implemented employing a receding-horizon scheme.
In this thesis, we explore the field of receding horizon games with unknown opponents. We consider potential games where the decision making problems of the agents are parameterized by different sets of behavior parameters, representing the preferences of each agent. We study non-cooperative scenarios and take the perspective of the individual agents, whose goal is to learn the behaviors of their opponents in order to play optimally.
We approach the problem assuming that agents maintain a local estimate of the underlying game, on which they base the choice of their strategies. After an iteration of the game is played, agents communicate by exchanging various forms of feedback, and use these information to update their knowledge of the game. In order to do so, we propose two online, distributed algorithms based on techniques from inverse optimization and learning in games.
We show and numerically analyse our results on an application inspired by the electricity market.

Contents

List of Figures

Notation

Acronyms

IOL inverse online learning

KKT Karush-Kuhn-Tucker

LP linear program

MISOCP mixed integer second order conic program

MPC model predictive control

NE Nash equilibrium

QP quadratic program

RHG receding horizon game

Symbols

$:=$ equal by definition

$[a, b]$ interval of real numbers $x \in \mathbb{R}$ with $a \leq x \leq b$

$[x^i]_{i=1}^m$ stacked vector $[x^i]_{i=1}^m := [(x^1)^\top, \ldots, (x^m)^\top]^\top = [x^1; \ldots; x^m], x^i \in \mathbb{R}^{n \times 1}$

$\mathbb{N}, \mathbb{N}_0$ set of natural numbers, set of natural numbers including zero

$\mathbb{R}, \mathbb{R}_{>0}, \mathbb{R}_{\geq 0}$ set of real, positive real, non-negative real numbers

$\mathbf{I}_n$ identity matrix in $\mathbb{R}^{n \times n}$

$\mathbb{1}_n$ vector of unit entries in $\mathbb{R}^n$

$| S |$ cardinality of the (finite) set S

$\nabla_x f(x)$ Jacobian of differentiable $f : \mathbb{R}^n \to \mathbb{R}^m$, i.e., $[\nabla_x f(x)]_{ij} := \frac{\partial g_j(x)}{\partial x^i}$, $\nabla_x f(x) \in \mathbb{R}^{n \times m}$

$\| x \|_2$ 2-norm of $x \in \mathbb{R}^n$

$\Pi_{\mathcal{X}}(x)$ metric projection of $x \in \mathbb{R}^n$ onto $\mathcal{X} \subseteq \mathbb{R}^n$

$A \succ 0 (\succeq 0)$ positive definite (-semi) $A \in \mathbb{R}^{n \times n}$, i.e., $x^\top A x > 0 (\geq 0), \forall x \neq 0$

A_{ij} element in position (i, j) of the matrix A

Reserved symbols

N number of players

$\mathcal{N}$ set of players, with $\mid \mathcal{N} \mid = N$

T length of the time horizon

$\mathcal{T}$ time horizon of length T: $\mathcal{T} = \{0, \dots, T-1\}$

t time step of the time horizon, $t \in \mathcal{T}$

k discrete time step of the time axis, $k \in \mathbb{N}_0$

$\mathcal{G}$ game

$\mathcal{K}^i$ feedback received by player $i \in \mathcal{N}$

$\mathcal{L}$ Lagrangian dual function

$\mathcal{R}$ regularization function

$x^i \in \mathbb{R}^T$ strategy vector of player $i \in \mathcal{N}$

$x^{-i} \in \mathbb{R}^{(N-1)T}$ strategy vector of all players except $i \in \mathcal{N}$

$x_t \in \mathbb{R}^N$ strategy vector of all players at time $t \in \mathcal{T}$

$x = [x_t]_{t \in \mathcal{T}} \in \mathbb{R}^{NT}$ strategy vector of all players

$x^\star$ Nash equilibrium strategy

$\mathcal{X}^i$ strategy or action set of player $i \in \mathcal{N}$

$\theta^i \in \Theta^i$ behavior parameter vector of player $i \in \mathcal{N}$

$\phi(x)$ potential function, $\phi : \mathbb{R}^{NT} \to \mathbb{R}$

$J^i(x)$ cost function of player $i \in \mathcal{N}$, $J^i : \mathbb{R}^{NT} \to \mathbb{R}$

Chapter 1

Introduction

1.1 Problem and outline

In this thesis, we address the problem of learning the behavior of agents in the framework of receding horizon games. In particular, we consider competitive scenarios and take the point of view of a strategic agent repeatedly playing a game against a number of other agents.
One of the motivations behind our work is that sometimes, in the existing literature on non-cooperative game theory, it is assumed that players can communicate with each other in order to choose their actions and play at an equilibrium of the game [13], which we think is highly unrealistic especially when considering competing agents, because it doesn't capture faithfully their self-interested nature. Hence, to let the agents communicate the least possible and in the most realistic way is a criterion we follow in this work: particularly, we will assume that the only possible communication between agents happens after an iteration of the game is played, in the form of observed actions or exchanged feedback. In order to select which actions to play, each agent maintains an estimate of the optimization problems of her opponents, thus being able to locally determine an estimated game and to compute an equilibrium of it. Hence, at every repetition of the game, the agents solve each a potentially different estimated game, which also differs (in general) from the true underlying game that is being played, and plays what they think are the optimal actions given their current knowledge of the game. After that, they exchange feedback and use the information received to update their understanding of the game, employing techniques from inverse optimization and learning in games.
Our objective is twofold. First, we wish to delineate a versatile framework for the problem of learning parameters in receding horizon games, able to handle different forms of uncertainty and capture various types of feedback. Second, we aim at the design of distributed algorithms that agents can utilize in order to correctly anticipate their opponents' actions.

The rest of this thesis is organized as follows. In the remainder of this chapter we provide an overview of the related works, and position our work in the broad literature. Chapter 2 briefly reviews mathematical tools that will be needed throughout this thesis. In Chapter 3 we formalize the problem, building the framework and presenting different kinds of feedback. Chapter 4 illustrates the solutions we propose to address it, and we apply the methods on a relevant example and analyze the results in Chapter 5. Chapter 6 concludes the thesis with a look at possible future extensions.

1.2 Related works

The problem of playing a game with unknown opponents has already been discussed extensively in the literature. In this section, we examine the state-of-the-art on various related topics, to

comprehend in a deeper way under which points of view has the problem been studied and what solutions have been developed to solve it. This review will also help to understand why our work lies at the intersection of the fields of inverse game theory and learning in games, by highlighting their main differences and points in common.

1.2.1 Inverse optimization

Inverse optimization is a field of optimization theory that tackles the challenge of estimating the underlying objective function and/or constraints of an optimization problem based on a given set of optimal (or nearly optimal) solutions, and it is closely related to our problem as, if we look individually at the two-player interactions that give life to a game, these can be seen as attempts of the two players to estimate each other's optimization problems. This problem holds significant practical importance across various fields such as medicine, transportation planning, robotics, and economics. For instance, in economy it is commonly assumed that consumers' buying decisions are guided by a utility function, which quantifies the level of satisfaction derived from purchasing a particular product. Clearly, in practice, access to a consumer's utility function is unavailable; however, by observing how consumer purchases respond to changes in prices, economists can employ inverse optimization techniques to derive and retrieve these utility functions. Essentially, these techniques enable economists to glean valuable insights from consumers' purchasing behaviors.
In the literature, inverse optimization methods can be distinguished based on the amount of data available (*single observation* or *multiple observations* methods) and on how data are collected (*batch* or *online* settings). Dong et al. [11] propose a framework for inverse optimization based on online data collection that is suitable for real-time applications and that can estimate cost function and (generalized) constraint parameters. In [19], the authors consider optimization problems with parametric, convex utility functions, and present a method for imputing these parameters, using prior knowledge about the objective. Nielsen and Jensen [25] address the problem of modeling a decision maker's utility function from (possibly) inconsistent behavior, interpreted as random deviations from an underlying true utility function. They propose two algorithms, one facilitating batch learning, and the other more suited for online, changing behavior. A problem closely related to inverse optimization is the one of inverse reinforcement learning [1, 24], which assumes that some information about the optimal policy is available, and the goal is to learn the reward function.

1.2.2 Learning in control

The learning in control literature specifically addresses scenarios where models for the controlled systems are not available, aiming at computing control actions where traditional model-based controller design principles can not be employed. Many approaches fall into this category, most notably data-driven control, reinforcement learning and system identification. Techniques from this branch of the literature can be useful for our case as our problem can be seen to some extent as a control problem where agents need to select inputs to apply to an unknown system, which is, the underlying game.
Data-driven methods leverage data obtained from a system in order to generate control inputs to apply to it. In [7], Dörfler et al. consider the problem of optimal trajectory tracking for unknown, deterministic LTI systems, and introduce an MPC-inspired approach called DeePC which uses input/output data samples from the unknown system in order to learn its behavior, rather then a model for it. The approach is extended to unknown stochastic LTI systems in [9], where uncertainties in the input/output data are taken into account and handled using distributionally robust stochastic optimization methods, which lead to a regularized DeePC algorithm. In [8], constrained unknown systems are controlled using the same method.

Reinforcement learning approaches utilize a reward function in order to select control actions for an unknown system. In [20], the authors consider an online, model-free method based on a Q-learning algorithm to solve the infinite-horizon linear quadratic tracker for unknown discrete-time systems. The authors of [26] address the Linear Quadratic control problem for linear stochastic systems, both static and time-varying, with unknown parameters using a Thompson sampling-based learning algorithm, which assumes a prior distribution on the parameters. All these methods require substantial amounts of data samples to achieve satisfactory performance. Additionally, they exhibit several drawbacks, such as high sensitivity to hyper-parameters, and safety issues, due to the fact that generally they are not able to take into account safety constraints. Another approach consists on performing sequential system identification and control. When the system model is not available, system identification techniques can be employed in order to estimate it and produce an approximate model [3, 31].

1.2.3 Inverse game theory

The field of inverse game theory deals with the problem of inferring the underlying parameters of an unknown game driving the observed equilibrium behaviors of agents, and captures scenarios, often present in the real world, that constitute the exact opposite of game-theoretical situations where instead we seek to determine the equilibrium of a known game. Imagine for example a group of autonomous vehicles operating in a traffic network, each aiming to optimize its own travel time and fuel efficiency while complying with traffic regulations. By collecting data on the vehicles' observed behaviors, such as their chosen routes, speeds, and lane-changing patterns, researchers can apply inverse game theory techniques to deduce the underlying parameters of the traffic game (e.g., preferences of the vehicles regarding travel time, fuel consumption, and other relevant factors).
Kuleshov and Schrijvers introduced in [23] the concept of inverse game theory, and showed that, in general, the problem of computing the agents' utilities from a set of correlated equilibria is NP-hard, unless the game is known to have special structures. The authors of [27] formulate inverse equilibrium problems by leveraging relaxed stationarity conditions from the KKT conditions, and illustrate the methodology on a Nash-Cournot game. In [33], the authors consider inverse games where the agents are known to have bounded rationality, and provide theoretical guarantees of payoff recovery in general Stackelberg games. Chen et al. [5] address the inverse game theory problem applying inverse optimization for estimating parameters of payoff functions in the context of electricity markets, using a data-driven batch approach based on past data. In [6], potential games are considered, and an algorithm to learn utilities and constraints parameters that explain the behavior of non-cooperative agents is proposed. Variational inequalities are used to solve the inverse game theory problem in [2].
What we do in this thesis shares a close connection with the field of inverse game theory. Our objective is identical, which is, to learn the rationale behind other players' actions. However, there is at least one key distinction between our problem and the one addressed by inverse game theory: while inverse game theory approaches assume that the observed players' actions constitute an (near-)equilibrium of the underlying game, we do not make this assumption.
In conclusion, our approach shares the same objective as inverse game theory methods, but differs in its initial premises.

1.2.4 Learning in games

Numerous practical challenges revolving around the interactions among multiple learning, competing agents, such as managing traffic flow, making market predictions, and understanding social network behavior, can be described as repeated games (see Section 2.1), in which the goal of every agent is to maximize her cumulative reward. Typically, the agents are unaware

of the underlying game dynamics, and the only way to learn about it is by repeatedly playing and observing the resulting game outcomes. The field of learning in games aims to tackle such scenarios by devising effective strategies for agents to effectively interact with their opponents.
To make an example, consider a ride-sharing platform. The drivers have the freedom to choose their routes and fares, and are competing with each other to attract more passengers and maximize their earnings. This situation can be modeled as a game, where each driver's actions affect their own rewards as well as the rewards of other drivers. By employing learning methods, drivers can adapt and improve their strategies over time based on the outcomes of their interactions. For example, they can experiment with different pricing strategies or route choices to increase their chances of getting passengers, and learn from their successes and failures, gradually developing effective tactics to maximize their cumulative rewards.
The majority of the learning in games literature focuses on the problem of learning optimal strategies from the perspective of the agents [10, 14, 21, 22]. In [30], the authors address the challenge by considering noisy bandit feedback paired with observations of the agents' actions, and consider a simultaneous game where the reward function of the learning player is unknown. The same authors, in [29], consider a sequential game between two players where the learner aims at inferring the response function of the opponent, and consider an adversarial sequence of opponents.
While the methods of the learning in games literature also operate under the assumption that agents' observations are not equilibria of the game, their objective of learning how to play a game fundamentally differs from our aim of learning the game's structure. Understanding how a game is structured has at least one significant advantage over just learning how to play a game optimally: it helps to comprehend the reasoning behind each player's decisions, thus enabling the learning agent to play optimally and adapt to changes in the game's initial conditions. In the case of learning in games, altering the initial conditions of a game implies restarting the process of learning how to play.
In conclusion, our approach shares the same initial premises as learning in games methods, but diverges in terms of its ultimate goal.

Chapter 2

Mathematical preliminaries

2.1 Game theory

Game theory is a set of mathematical tools used to model and analyse situations involving multi-agent interaction and decision making [15]. The agents involved are also called *players* or *decision makers*, and each of their possible decisions leads to a different individual outcome, usually called *cost* or *utility*, which, in a situation of collective decision making, can be determined also by the decisions of other players. A game is *noncooperative* if each agent involved pursues his or her own interests, which are partly conflicting with others' interests. It is *cooperative* if, instead, it models situations where agents have the same overall goal, i.e., they cooperate. In the so called *normal-form*, a game can be represented using three elements:

Definition 1. *(Normal-form game) A game in normal-form (or in strategic-form) is an ordered triple* $\mathcal{G} = (\mathcal{N}, [\mathcal{X}^i]_{i\in\mathcal{N}}, [J^i]_{i\in\mathcal{N}})$ *where:*

- $\mathcal{N} = \{1, 2, \dots, N\}$ *is a finite set of players. Each player is identified with an index* $i \in \mathcal{N}$*;*
- $\mathcal{X}^i \subseteq \mathbb{R}^n$ *is the strategy or action set of player i, for every player* $i \in \mathcal{N}$*. We use* x^i *to denote the action of player i, and* x^{-i} *to indicate the vector of actions of players* $j \in \mathcal{N}\backslash\{i\}$*. The joint action set is defined by* $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^N$*;*
- $J^i : \mathcal{X} \to \mathbb{R}$ *is a function, called utility or cost, associating each vector of strategies* $x = [x^i]_{i\in\mathcal{N}}$ *with a real number representing the outcome* $J^i(x)$ *of player i, for every player* $i \in \mathcal{N}$*.*

One more element needed to fully characterize a game is a solution concept, which in game-theoretic terms is called *equilibrium*. Although different notions of it exists, perhaps the most famous one, and the one we will use throughout this work, is the notion of *Nash equilibrium*:

Definition 2. *(Nash equilibrium) A vector of strategies* $x^\star = [x^{i\star}]_{i\in\mathcal{N}}$ *is a Nash equilibrium of the game G if* $x^\star \in \mathcal{X}$ *and*

$$J^i(x^{i\star}, x^{-i\star}) \leq J^i(x^i, x^{-i\star}) \tag{2.1}$$

for all $i \in \mathcal{N}$ *and all* $x^i \in \mathcal{X}^i$*.*

Situations which involve more than one interaction between players are modelled through repeated games.

Definition 3. *(Repeated game) A game* $\mathcal{G}^{(T)}$ *is a repeated game if it consists of the same base game* $\mathcal{G}$ *repeated T times.*

In this thesis, we mostly focus on a class of games called convex games.

Definition 4. *(Convex game) An N-player game with continuous action spaces $\mathcal{X}^i$ is called convex if:*

- *the action spaces $\mathcal{X}^i \subset \mathbb{R}^n$ are compact and convex;*
- *J^i are continuous in $x \in \mathcal{X}$;*
- *J^i are convex in x^i for fixed x^{-i}.*

2.1.1 Potential games

In game theory, there exist some types of strategic interaction models that have additional structure w.r.t. the normal-form games, which confers certain mathematical properties. One example of these can be seen in *potential games*. In a potential game there exists a function of the strategies of the players, called *potential*, which is such that, for each player i, the difference between the potential values of any two vectors of strategies only differing in the strategy of player i is equal to the difference in player i's individual outcome. In other words, when a player deviates from one strategy to another, the change in the potential function value reflects the change in their own outcome. More formally:

Definition 5. *A game $\mathcal{G}$ is said to be an (exact) potential game if there exists a function $\phi(x)$ such that*

$$J^i(x^i, x^{-i}) - J^i(\tilde{x}^i, x^{-i}) = \phi(x^i, x^{-i}) - \phi(\tilde{x}^i, x^{-i}), \quad \forall x^i, \tilde{x}^i \in \mathcal{X}^i, x^{-i} \in \mathcal{X}^{-i}, i \in \mathcal{N}, \tag{2.2}$$

and ϕ is called an (exact) potential for the game.

It's important to notice that the potential function of a game is not uniquely defined. Trivially, if $\phi(\cdot)$ is a potential for $\mathcal{G}$, then, for every constant c, $\phi(\cdot) + c$ is also a potential for the same game.
The interesting property induced by this additional structure is that, in potential games, directionally-local minima of the potential ϕ are Nash equilibria:

Proposition 1. *Consider a potential game $\mathcal{G}$ with potential ϕ. An N-tuple of strategies $x^\star := (x^{1\star}, ..., x^{N\star})^\top \in \mathcal{X}$ is a Nash equilibrium of $\mathcal{G}$ if and only if it is a directionally-local minimum of ϕ.*

In this work, we consider games whose action spaces are intervals in the real line. For games of this kind, it is straightforward to determine whether they are potential or not:

Lemma 1 (Potential games with continuous action spaces)**.** *For a given game $\mathcal{G}$, suppose that every action space $\mathcal{X}^i$ is a closed interval in $\mathbb{R}$ and every outcome J^i is twice continuously differentiable. Then, the following three statements are equivalent:*

1. *$\mathcal{G}$ is an exact potential game;*
2. *There exists a twice differentiable function $\phi(x)$ such that*

$$\frac{\partial J^i(x)}{\partial x^i} = \frac{\partial \phi(x)}{\partial x^i}, \quad \forall x \in \mathcal{X}, i \in \mathcal{N} \tag{2.3}$$

and when these equalities hold, ϕ is an exact potential for the game;

3. The outcome satisfies

$$\frac{\partial^2 J^i(x)}{\partial x^i x^j} = \frac{\partial^2 \phi(x)}{\partial x^i x^j}, \quad \forall x \in \mathcal{X}, i, j \in \mathcal{N} \tag{2.4}$$

and when these equalities hold, we can construct an exact potential using

$$\phi(x) = \sum_{k=1}^{N} \int_0^1 \frac{\partial J^k(\zeta + \tau(x-\zeta))}{\partial x^k}(x^k - \zeta^k)\, d\tau, \quad \forall x \in \mathcal{X}, \tag{2.5}$$

where ζ can be any element of $\mathcal{X}$.

2.1.2 Receding horizon games

The main focus of this thesis is on *receding horizon games* [13]. With this name we refer to an MPC-inspired game-theoretic framework that takes a fundamental element of Model Predictive Control, the receding-horizon implementation, and applies it to multi-stage games. The result is a closed-loop, multi-stage game where agents implement their actions in a receding-horizon fashion. This new class of games is motivated in [13] as a way to bridge the gap between fully-cooperative MPC schemes and the need to find an alternative to the inefficient repeated open-loop control employed in many situations, such as in most existing game-theoretic demand-side management (DSM) mechanisms.

To formalize, consider a prediction horizon $\mathcal{T} = \{0, \dots, T-1\}$. Each agent $i \in \mathcal{N}$ aims to select actions $x^i = [x_t^i]_{t\in\mathcal{T}}$, $x_t^i \in \mathcal{X}_t^i$, where $\mathcal{X}_t^i \subset \mathbb{R}^n$ is the action space of agent i at time t, and $x^i \in \mathcal{X}^i = \mathcal{X}_0^i \times \cdots \times \mathcal{X}_{T-1}^i \subset \mathbb{R}^{nT}$, minimizing the sum of their stage costs over the horizon $\mathcal{T}$, by solving the following optimal control problem:

$$\begin{aligned} \min_{x^i} \quad & \sum_{t\in\mathcal{T}} J_t^i(x) \\ \text{s.t.} \quad & g^i(x) \le 0, \\ & x_t^i \in \mathcal{X}_t^i, \quad \forall t \in \mathcal{T} \\ & x_0^i = \bar{x}_0^i, \end{aligned} \tag{2.6}$$

where $x_t = [x_t^i]_{i\in\mathcal{N}}$, $x = [x_t]_{t\in\mathcal{T}}$, $\bar{x}_0^i$ is the initial state of agent i and $g^i(x) \in \mathbb{R}^m$ is a set of constraints.

The collection of these N inter-dependent optimal control problem constitutes a (generalized) game. By defining the cumulative cost of agent i as the sum of the stage costs

$$J^i(x) = \sum_{t\in\mathcal{T}} J_t^i(x), \tag{2.7}$$

and the global action set as

$$\mathcal{Z}(x) = \{\forall i \in \mathcal{N}, \quad x^i \mid g^i(x) \le 0, x_t^i \in \mathcal{X}_t^i, \forall t \in \mathcal{T}, x_0^i = \bar{x}_0^i\}, \tag{2.8}$$

we can recast the game more compactly in the following *standard form* for generalized games:

$$\forall i \in \mathcal{N}: \quad \begin{cases} \min\limits_{x^i} & J^i(x) \\ \text{s.t.} & (x^i, x^{-i}) \in \mathcal{Z}(x). \end{cases} \tag{2.9}$$

A suitable solution concept for this game is the (generalized) Nash equilibrium (GNE) introduced in Definition 2, which we reformulate here coherently with 2.9: a vector of strategies $x^\star = [x^{i\star}]_{i\in\mathcal{N}}$

is a Nash equilibrium for the generalized game 2.9 if no agent can reduce its cost by unilaterally changing its action, i.e.:

$$J^i(x^{i\star}, x^{-i\star}) \leq J^i(x^i, x^{-i\star}), \quad \forall i \in \mathcal{N}, \forall (x^i, x^{-i\star}) \in \mathcal{Z}(x). \tag{2.10}$$

In this thesis we will compute a GNE of 2.9 distributedly, but, in general, other solutions are available for centralized or semi-decentralized communication structures.

At this point, we are ready to introduce the part inspired by MPC, which is, the receding-horizon implementation of the players' actions. Essentially, at every repetition of the game happening at time step $k \in \mathbb{N}_0$, the players compute a GNE $x^\star \in \mathcal{X}$ of 2.9, and each player i applies the first $a \in \{1, \ldots, T\}$ elements $x_0^{i\star}, \ldots, x_{a-1}^{i\star,}$ of their predicted control trajectory. Then, computation of the GNE and application of the control actions is repeated after $k + a$ time steps.

2.2 Optimization theory

2.2.1 Inverse optimization

In traditional mathematical optimization, an agent aims at generating an optimal decision given an objective function and a set of constraints. In inverse optimization, instead, decisions are given as inputs and the goal of the problem solver is to determine parameters of an optimization model (i.e. an objective function and a set of constraints) that render those decisions approximately or exactly optimal with respect to that model.
Traditional optimization problems are usually represented using the so called *forward optimization model* [4]:

$$\begin{aligned} \min_x \quad & f(x, \theta) \\ \text{s.t.} \quad & x \in \mathcal{X}(\theta), \end{aligned} \tag{2.11}$$

where x is the decision variable, $\mathcal{X}(\theta) \subseteq \mathbb{R}^n$ is the feasible set and $\theta \in \Theta \subseteq \mathbb{R}^p$ is a parameter vector that influences the objective $f(x, \theta) \in \mathbb{R}$, the feasible set $\mathcal{X}(\theta)$, or both. Common choices for modelling the objective function which will also be relevant for this thesis are:

- linear objective: $f(x, \theta) = \theta^\top x$, where $\Theta \subseteq \mathbb{R}^n$;
- quadratic objective: $f(x, \theta) = \frac{1}{2} x^\top Q x + q^\top x + c$, where $Q \in \mathbb{R}^{n \times n}, q \in \mathbb{R}^n, c \in \mathbb{R}$, and $\theta := (Q, q, c) \in \Theta \subseteq \mathbb{R}^{n \times n} \times \mathbb{R}^n \times \mathbb{R}$.

The feasible region is usually defined as

$$\mathcal{X}(\theta) := \{x \in \mathbb{R}^n \mid g(x, \theta) \leq 0\}, \tag{2.12}$$

where $g : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$ is the vector-valued set of equality and inequality constraints, and we denote with $\mathcal{S}(\theta)$ the optimal solution set, which is the set of points in $\mathcal{X}(\theta)$ that are optimal under θ:

$$\mathcal{S}(\theta) := \arg\min_x \{f(x, \theta) \mid x \in \mathcal{X}(\theta)\}. \tag{2.13}$$

The forward optimization model is also what inverse optimization tries to estimate. In particular, given a dataset of dimension D of solutions of an unknown forward optimization model, $\{\bar{x}_d\}_{d=1}^D$, inverse optimization seeks to find a parameter vector $\theta^\star$ such that the corresponding forward optimization model it induces explains the dataset well.
Based on features such as the dimension of the forward problem and dataset, the way of collecting data (offline or online), and the precision needed for the estimation, various formulations can

be considered that lead to different design of the *inverse optimization problem*. Relevant for the work we do in this thesis are the *classical* and the *learning-based* formulations.
Classical inverse optimization aims to find a parameter vector θ that solves the following bilevel program (see Section 2.2.2):

$$\min_{\theta} \left\{ h(\theta) \mid \theta \in \Theta_d^{inv}(\bar{x}_d), \quad \forall d \in \{1, \dots, D\}, \theta \in \Theta \right\}, \tag{2.14}$$

where $h(\theta)$ is an application-specific objective and $\Theta_d^{inv}(\bar{x}_d)$ is the set of all θ that induce a forward problem for which $\bar{x}_d$ is an optimal solution, and it is defined as:

$$\Theta_d^{inv}(\bar{x}_d) := \{\theta \mid \bar{x}_d \in \mathcal{S}_d(\theta)\}. \tag{2.15}$$

Inverse optimization can also be addressed using a number of learning-based paradigms, most notably Inverse Online Learning, an approach able to handle online input data collection and which will be discussed later in this work. The main idea behind IOL is that an estimate for θ is obtained iteratively by generating estimates θ_t from the refinition of existing ones θ_{t-1} at each "round" $t \in \{1, 2, \dots, T\}$, when the observation of a new decision $\bar{x}_t \in \mathcal{X}_t(\theta)$ becomes available. In order to do this, a suitable *update rule* is employed.

2.2.2 Bilevel optimization

Problem 2.14 is an example of a bilevel program. Note the hierarchical structure of the minimization problem: in order to solve it, one must first determine the inverse-feasible set $\Theta_d^{inv}(\bar{x}_d)$, whose computation requires solving another optimization problem.

Usual optimization problems as 2.11 are single-level problems, with one objective function, one vector of decision variables and one set of constraints, and they model situations where the decisions are all taken by a single decision maker. Bilevel problems, instead, are useful when it is necessary to model scenarios where a decision is taken while anticipating the rational reaction of another decision maker, whose decision depends on the first decision. This hierarchical structure is common in many areas spanning from economics, logistics, homeland security, computer science and many more.

Definition 6 (Bilevel optimization problem)**.** *A bilevel optimization problem reads*

$$\begin{aligned} \min_{x,y} \quad & F(x, y) \\ s.t. \quad & G(x, y) \leq 0, \\ & y \in S(x), \end{aligned} \tag{2.16}$$

where $S(x)$ is the set of optimal solutions of the x-parameterized single-level problem

$$\begin{aligned} \min_{y} \quad & f(x, y) \\ s.t. \quad & g(x, y) \leq 0. \end{aligned} \tag{2.17}$$

Problem 2.16 is called *upper-level problem* and Problem 2.17 is the *lower-level problem*. The variables $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ are the upper-level decision variables and $y \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ are the lower-level decision variables, where the feasible regions $\mathcal{X}$ and $\mathcal{Y}$ are defined accordingly to 2.12. The objective functions are indicated by $F, f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$ and the constraint functions by $G : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^m$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^l$.

Most frequently in practice, the solution methods for bilevel problems imply using a *single-level reformulation* in order to bring them into their single-level form, which can then be solved using

an appropriate solver for the resulting problem. Although many single-level reformulations exist, each with distinct properties and suitable for different classes of problems, in this thesis we will mainly make use of the *KKT reformulation*. This type of reformulation is only applicable to those cases where the lower-level problem has convexity and satisfies certain constraint qualifications, as it is based on the idea of exploiting optimality conditions of the lower-level problem.
Let's consider the bilevel program in Definition 6 and assume that the lower-level problem is convex, which is, $f(x,\cdot)$ and $g(x,\cdot)$ are convex functions for all $x \in \mathcal{X}$, and $\mathcal{Y}$ is a convex set. Moreover, assume that Slater's constraint qualification holds for the lower-level program:

Definition 7. *(Slater's constraint qualification) For a given upper-level feasible point x of the bilevel problem in Definition 6, we say that Slater's constraint qualification holds for the lower-level problem if there exists a point $\hat{y}(x)$ with $g_i(x,\hat{y}(x)) < 0$ for all $i = 1,\dots,l$.*

Under these assumptions, the bilevel problem can be reformulated into its single-level form by writing the lower-level program as a set of KKT conditions:

$$\begin{aligned}
\min_{x,y,\lambda} \quad & F(x,y) \\
\text{s.t.} \quad & G(x,y) \leq 0, \\
& \nabla_y f(x,y) + \lambda^\top \nabla_y g(x,y) = 0, && (stationarity) \\
& \lambda^\top g(x,y) = 0, && (complementarity) \\
& g(x,y) \leq 0, && (primal feasibility) \\
& \lambda \geq 0, && (dual feasibility)
\end{aligned} \tag{2.18}$$

where we now optimize over an extended space of variables since we additionally have to include $\lambda \in \mathbb{R}^l_{\geq 0}$, the dual variables of the lower-level problem. Note that the complementarity constraints of the KKT conditions of the lower-level problem render the problem nonconvex.

Chapter 3

Problem formulation

3.1 Problem setting

3.1.1 Multi-agent decision making problem

Consider a non-cooperative and simultaneous game among N agents, played over a finite-time horizon of length T: $\mathcal{T} = \{0, \dots, T-1\}$. Each agent $i \in \mathcal{N} = \{1, \dots, N\}$ is interested in selecting actions $x^i = [x_0^{i\top}, \dots, x_{T-1}^{i\top}]^\top = [x_t^i]_{t\in\mathcal{T}}$ such that a cumulative objective function J^i is minimized, while taking into account an action set $\mathcal{X}^i = \mathcal{X}_0^i \times \cdots \times \mathcal{X}_{T-1}^i$, $\mathcal{X}_t^i \subset \mathbb{R}^n$ for all $t \in \mathcal{T}$, and shared constraints that limit the choice.
Player i's choice of actions follows the rationale captured by the following parameterized decision-making problem:

$$\begin{aligned} \min_{x^i \in \mathcal{X}^i} \quad & J^i(x, \theta^i) \\ \text{s.t.} \quad & g^i(x, \theta^i) \leq 0, \end{aligned} \tag{3.1}$$

where the strategy profile made of the decisions of all players, x, the strategy set of player i, $\mathcal{X}^i$, and the cumulative cost over the time horizon, $J^i : \mathbb{R}^{nTN} \times \mathbb{R}^p \to \mathbb{R}$ are all defined as in Section 2.1.2, $g^i : \mathbb{R}^{nTN} \times \mathbb{R}^p \to \mathbb{R}^m$ is vector valued, and $\theta^i \in \Theta^i \subset \mathbb{R}^p$ is the parameter vector representing the individual preferences of agent i.
We make a few assumptions on the optimization problems of the N players.

Assumption 1. *The sets Θ^i are convex and compact. There exists $D^i > 0$ such that $\|\theta^i\|_2 \leq D^i$ for all $\theta^i \in \Theta^i$.*

Assumption 2. *For each $\theta^i \in \Theta^i$, both $J^i(x, \theta^i)$ and $g^i(x, \theta^i)$ are convex in x.*

Hence, the cost functions of all the players have the same convex structure, and only differ for a set of parameters. As we have seen in Section 2.1.2, since these N optimization problems are coupled in the cost functions and constraints, they form a generalized game. We assume that the game is *potential* as in Definition 5, hence, a solution can be found by solving the following related minimization problem [6]:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & \phi(x, \theta) \\ \text{s.t.} \quad & g^i(x, \theta^i) \leq 0, \quad \forall i \in \mathcal{N} \end{aligned} \tag{3.2}$$

where $\theta = [\theta^i]_{i\in\mathcal{N}} \in \Theta \subset \mathbb{R}^{pN}$ is the vector of parameters of all the agents, and $\Theta = \Theta^1 \times \cdots \times \Theta^N$. We denote $\mathcal{X}_\theta = \{x \in \mathcal{X} \mid g^i(x, \theta^i) \leq 0, \quad \forall i \in \mathcal{N}\}$ the feasible region of Problem 3.2, and $S_\theta = \arg\min\{\phi(x, \theta) \mid x \in \mathcal{X}_\theta\}$ the optimal solution set of Problem 3.2. We indicate this parameterized game with $\mathcal{G}(\theta)$.

Assumption 3. *The potential function $\phi(x, \theta)$ is convex in x for every $\theta \in \Theta$.*

We recall that, since the Cartesian product of convex and compact sets is a convex and compact set as well, Assumption 1 holds for Θ.

As we pointed out before, the objective of each player is to select the strategy x^i, among the available ones, that is the best in terms of minimization of the cost function $J^i(x, \theta^i)$ over the horizon $\mathcal{T}$. However, they do not know exactly the underlying game that is being played, so they are not able to predict the actions of their opponents which influence their payoff as well as their actions. We model this scenario through the parameters θ, assuming that player i only has perfect knowledge of θ^i, while uncertainty surrounds all the parameters θ^{-i} defining the optimization problems of the other players. In order to reach their goals, players are motivated to learn these parameters.

To summarize the outlined situation, we are considering a number of decision makers interacting with each other in a competitive setting, trying to learn others' preferences, with the aim of using this knowledge to make decisions that best satisfy their selfish nature. It's not difficult to imagine such circumstances happening in a real-life scenario.

3.1.2 Playing the game

We assume that each player has an initial knowledge of the parameter of other players (coming, for example, from past data), which we represent through an uncertainty set that is known to contain the parameter, and which remains constant in the learning process. We call Θ^{ij} the uncertainty set of agent i with respect to player j's parameter, and assume that the parameters in the set have uniform probability to be the correct one. We use $\hat{\theta}^{ij}_k$ to indicate player i's estimate of player j's parameters at time k, for all $i \in \mathcal{N}, j \in \mathcal{N} \setminus \{i\}$, and the vector $\theta^i_k = [\hat{\theta}^{i1\top}_k, \dots, \theta^{i\top}, \dots, \hat{\theta}^{iN\top}_k]^\top$ to represent the knowledge that player i has of the game at time k: this allows each player to locally build, at each time step k, the estimated game $\mathcal{G}(\theta^i_k)$ and to solve it using the conventional Nash equilibrium concept. Each realization of the vector of estimates θ^i_k induces an equilibrium

$$\hat{x}(\theta^i_k) = [\hat{x}^1(\theta^i_k), \dots, \hat{x}^N(\theta^i_k)]^\top \in \mathcal{X}, \quad \forall i \in \mathcal{N} \tag{3.3}$$

which we denote with a *hat* to stress that this is an equilibrium of the estimated game $\mathcal{G}(\theta^i_k)$, but not (necessarily) of the true underlying game $\mathcal{G}(\theta)$.

At this point, if an external entity could see what the interacting agents have in mind, it would see a collection of N Nash equilibria $[\hat{x}(\theta^i_k)]_{i \in \mathcal{N}}$.

Using the receding-horizon implementation introduced in Section 2.1.2, each agent plays the first a actions of their corresponding strategy of the Nash equilibrium they computed, which is, player i plays $x^i_a = [\hat{x}^i_0(\theta^i_k), \dots, \hat{x}^i_{a-1}(\theta^i_k)]^\top$ from the strategy vector $\hat{x}^i(\theta^i_k)$ from the Nash equilibrium $\hat{x}(\theta^i_k)$, and obtains a cumulative payoff given by $\sum_{t=0}^{a-1} J^i_t(x_a, \theta^i)$, where $x_a = [x^i_a]_{i \in \mathcal{N}}$ is the vector of all actions played (note that we dropped the hat in the notation to indicate actions that are actually played and not only planned). After this, the agents exchange information in the form of a *feedback*. In the following we will consider various kinds of feedback, differing in terms of the quantity of information exchanged and the potential presence of noise and/or other disturbances. We assume that players share information in a symmetric fashion: player i equally informs all players $j \in \mathcal{N} \setminus \{i\}$ (sending the same type and "amount" of information), and all players accept to exchange information (the exchange is bidirectional). We indicate with x^j_f the information broadcast by player j, for all $j \in \mathcal{N}$, and with $\mathcal{K}^i = [x^j_f]_{j \in \mathcal{N} \setminus \{i\}}$ the feedback received by player i from all the other agents, for all $i \in \mathcal{N}$.

Remark 1. *This passage apparently goes against two key principles we follow in this work: the will of reducing communication to a minimum and the competitive nature of the players.*

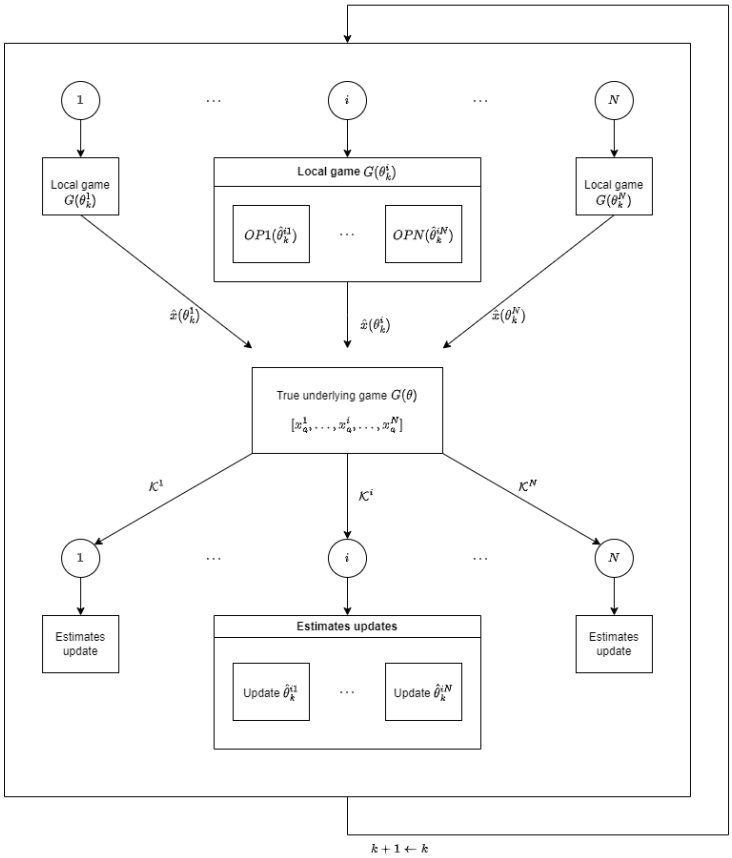


Figure 3.1: Block diagram depicting how the game is played

Regarding the former, our claim is that communication between players after *the game is played, which is, in the form of feedback, is less impactful and more realistic than letting competing agents communicate* in order to *play the game optimally; for the latter, we agree that the exchange of information is a form of collaboration, however, we could say that in some situations competing agents are open to collaborate to some extent if they can benefit from it.*

The next step in the approach concerns using the received information in order to update one's belief of the opponents' parameter, and progress in the knowledge of the game. We will propose two algorithms to tackle this *update step*, both relying on the concept of *loss function* to capture the mismatch between predictions and observations. The algorithms will be introduced in Chapter 4, while in the remaining of this Chapter we present different loss functions typically used in the literature, as well as the various forms of information feedback that we consider in this work.

3.2 Loss functions and feedback

In the previous section we presented the general, fixed framework of the problem we tackle in this work. Here, instead, we look at two blocks whose usage can be considered interchangeable, in the sense that we can exploit different formulations in order to adapt to the situation and/or to get more favourable properties. Moreover, these are fundamental parts of the learning process, as they are related to what kind of information we receive and how we use it in order to advance our learning.

3.2.1 Loss functions

A loss function is a mathematical tool that quantifies the discrepancy between the predicted output of a model and the actual output. Essentially, it measures how well or poorly the model is performing on a given task.
Consider now an agent i aiming to learn the parameter θ^j of an opponent j, using the relevant information contained in the feedback $\mathcal{K}^i$. When using a loss function, the objective of the agent is captured by

$$\min_{\theta \in \Theta} l_\theta(x_f^j), \tag{3.4}$$

where $l_\theta(x_f^j) \in \mathbb{R}$ denotes the loss experienced by the learning agent when it models agent j's preferences with θ, and receives x_f^j from agent j.
In this section we introduce choices of loss functions that are common in the literature, and in the next Chapter we discuss how they apply to our case.

Predictability loss. The *predictability loss* of model θ is defined as

$$l_\theta^p(x) := \min_{y \in \mathcal{S}_\theta} \|x - y\|_2^2. \tag{3.5}$$

When an agent receives x as a feedback, $l_\theta^p(x)$ quantifies the goodness of model θ by measuring how close x is from the optimal set S_θ predicted by θ, in the (squared) Euclidean distance sense. Clearly, this loss is always non-negative and evaluates to zero if and only if $x \in \mathcal{S}_\theta$. Among the losses we present here, this one best captures the agent's objective to predict the opponent's decisions. However, as we will see, it also renders the problem an NP-hard bilevel optimization problem.

Suboptimality loss. The *suboptimality loss* of model θ is defined as

$$l_\theta^s(x) := \phi(x, \theta) - \min_{y \in \mathcal{X}_\theta} \phi(y, \theta). \tag{3.6}$$

and was first introduced in [12]. When an agent receives x as a feedback, $l_\theta^s(x)$ quantifies the goodness of model θ by measuring the extent to which the solution x is suboptimal with respect to the hypothetical utility function $\phi(\cdot, \theta)$. Note that, as it happens with the predictability loss, the suboptimality loss is always non-negative and evaluates to zero if and only if $x \in \mathcal{S}_\theta$.

First-order loss. The *first-order loss* of model θ is defined as

$$l_\theta^f(x) := \max_{y \in \mathcal{X}_\theta} \nabla_x \phi(x, \theta)^\top (x - y), \tag{3.7}$$

and clearly it is defined only if $\phi(\cdot, \theta)$ is differentiable with respect to x. When an agent receives x as a feedback, $l_\theta^f(x)$ quantifies the goodness of model θ by measuring the extent to which x violates the first-order optimality condition of the optimization problem 3.2, where ϕ is evaluated with θ. Note that, as it happens with the predictability and suboptimality losses, the first-order loss is always non-negative and evaluates to zero if and only if $x \in \mathcal{S}_\theta$.

3.2.2 Feedback models

Let us now expand on the topic of feedback. As we explained above, in order to step forward in their understanding of the underlying game, agents need to make use of information, shared by the opponents in the form of feedback. Many feedback models can be utilized to this end

that have been extensively considered in the literature, mainly differing in the "amount" and type of information exchanged (e.g., utility function values, played actions, ...) and in the presence of *imperfections*. Here, we account for one particular type of feedback, which is, actions played or planned by the agents, but value variety in terms of quantity and possible imperfections.

Perfect feedback model. With perfect feedback model, we intend a form of information feedback which is *full information* and *noiseless*: each agent exchanges the complete Nash equilibrium over the entire time horizon computed based on its knowledge of the game, and we assume that this information arrives uncorrupted by noise to the other agents. Hence, we have that $x_f^j = \hat{x}(\theta_k^j)$, and the feedback received by player i after iteration k of the game is:

$$\mathcal{K}^i = \{\hat{x}(\theta_k^j)\}_{j \in \mathcal{N} \setminus \{i\}}. \tag{3.8}$$

Clearly, such a model is unrealistic and only useful for theoretical purposes.

Imperfect feedback models. We consider different forms of imperfect feedback:

(i) **Noisy feedback.** We relax the assumption that agents receive precise, clear information from others. As a matter of fact, in practice, the observations of agents usually are corrupted by noise. In the presence of *measurement noise*, the shared feedback x_f^j contains randomly perturbed information that we denote using a *tilde*. Therefore, if $\hat{x}(\theta_k^j)$ is the exact equilibrium computed and shared by agent j, in the presence of measurement noise it reaches agent i as $\tilde{x}(\theta_k^j)$. Essentially, this represents a suboptimal solution to the game estimated by player j, maybe even infeasible. Note that this type of feedback can also be used to model other kinds of uncertainty, such as *bounded rationality* of the agents;

(ii) **Partial feedback.** As a form of imperfection, we also account for the case where the players only share part of their local solution. In particular, we will consider two types of partial feedback: (1) the agents agree on sharing only the equilibrium actions corresponding to the first $t \in \mathcal{T}$ steps of the horizon: $x_f^j = [\hat{x}_t^1(\theta_k^j), \ldots, \hat{x}_t^N(\theta_k^j))]^\top$; (2) the agents agree on sharing only the equilibrium information related to their actions $x_f^j = \hat{x}^j(\theta_k^j)$.

Chapter 4

Proposed solutions

In this chapter, we focus on the update step and present ways in which agents can add new information to existing estimates and improve their knowledge of others' preferences. More precisely, we propose two algorithms that help us in this direction, both capable of handling data that arrive sequentially in an online manner: the first one formulates the learning problem as an inverse optimization problem, the second is based on the differentiation of the loss function.

4.1 Multi-agent implicit online learning algorithm

The first algorithm we present is adapted from [11], where the authors consider the scenario of a learner aiming to learn the parameterized objective function or constraints of a single decision maker. We adapt the algorithm to the case of a N-player game, where each agent is a learner, trying to learn $N-1$ parameterized objective functions or constraints, but also a decision maker.

Algorithm 1 Multi-agent implicit online learning

Input θ_0^i for all $i \in \mathcal{N}$

for $k = 0$ to $K-1$ **do**
  for all players $i \in \mathcal{N}$ **do**
    for all players $j \in \mathcal{N} \setminus \{i\}$ **do**
      receive x_f^j
      suffer loss $l_{\theta_k^i}(x_f^j)$
      if $l_{\theta_k^i}(x_f^j) = 0$ **then**
        $\hat{\theta}_{k+1}^{ij} \leftarrow \hat{\theta}_k^{ij}$
      else
        set learning rate $\eta_k \propto 1/\sqrt{k}$
        update $\hat{\theta}_{k+1}^{ij} \leftarrow \arg\min_{\theta \in \Theta} \frac{1}{2}\mathcal{R}(\theta^j) + \eta_k l_\theta(x_f^j)$
      end if
    end for
  end for
end for

Let us consider the point of view of a generic player i. At each round k, player i receives the feedback $\mathcal{K}^i$ containing x_f^j for all $j \in \mathcal{N} \setminus \{i\}$. Then, he/she measures how good the current estimate θ_k^i is by evaluating the loss function $l_{\theta_k^i}(x_f^j)$, for all $N-1$ players $j \in \mathcal{N} \setminus \{i\}$. Whenever $l_{\theta_k^i}(x_f^j) = 0$, it means that the estimate $\theta_k^i = [\hat{\theta}_k^{i1\top}, \dots, \theta^{i\top}, \dots, \hat{\theta}_k^{iN\top}]^\top$ of player i at round k is good enough to explain the behavior of player j. In such a case, the estimate $\hat{\theta}_k^{ij}$ corresponding

to player j does not get updated. Conversely, when $l_{\theta_k^i}(x_f^j) \neq 0$ the vector θ_k^i can't predict the behavior of player j satisfyingly enough, and an update is necessary. When this happens, player i updates the parameter of the vector corresponding to player j.
There are some key aspects to notice:

1. When evaluating the loss function, player i utilizes the full vector of estimates θ_k^i, but when he/she decides to (not) update an estimate, it is only the parameter of an agent j that gets (not) updated;

2. The term $\mathcal{R} : \mathbb{R}^p \to \mathbb{R}$ in the update rule is called *regularizer* or *regularization function*, and represents how conservative we want to be in maintaining the current estimation. Note that, since each θ_k^{ij} gets updated independently from the other entries of the vector θ_k^i, the regularizer only takes into account the parameter corresponding to player j. In the following, we will use $\mathcal{R}(\theta^j) = \|\theta^j - \hat{\theta}_k^{ij}\|_2^2$, but other functions are possible;

3. The update rule looks for a vector $\theta \in \Theta$, however, only one parameter among the estimated ones is actually used for the update.

Essentially, at each round k, player i estimates an $N \times N \times p$ matrix of parameters

$$\begin{bmatrix} \theta_k^{11} & \dots & \theta_k^{1N} \\ \vdots & & \vdots \\ \theta_k^{i1} & \ddots & \theta_k^{iN} \\ \vdots & & \vdots \\ \theta_k^{N1} & \dots & \theta_k^{NN} \end{bmatrix}, \tag{4.1}$$

where the i-th row corresponds to his/hers knowledge vector θ_k^i, and the jz-th entry is player i's estimate of $\hat{\theta}_k^{jz} \in \mathbb{R}^p$, which is, the estimate of player j's estimate of player z parameter, for $j, z \in \mathcal{N}, j \neq i$. Of all these estimates, only the diagonal entries are used to update the Np elements of θ_k^i.
The heart of Algorithm 1 resides in the update rule

$$\hat{\theta}_{k+1}^{ij} \leftarrow \arg\min_{\theta \in \Theta} \frac{1}{2}\mathcal{R}(\theta^j) + \eta_k l_\theta(x_f^j). \tag{4.2}$$

This rule says that, starting from an initial estimate and making use of the information x_f^j, a new, updated estimate $\hat{\theta}_{k+1}^{ij}$ can be derived by solving an optimization problem. Thus, the entire online learning process is broken down to a sequence of single observation inverse optimization problems. Note that an important role in this decomposition is played by the regularization function, since, by enforcing closeness to the current parameter $\hat{\theta}_k^{ij}$, it encapsulates the information coming from past observations.

Let's now see how the loss functions we introduced in Chapter 3 influence this update step.
Again, we take the perspective of an agent $i \in \mathcal{N}$ updating its estimate of the parameter of agent $j \in \mathcal{N} \setminus \{i\}$.

Predictability loss. When using the predictability loss, the update rule reads

$$\begin{aligned} \hat{\theta}_{k+1}^{ij} = \arg\min_{\theta \in \Theta} \quad & \frac{1}{2}\mathcal{R}(\theta^j) + \eta_k \|x_f^j - x\|_2^2 \\ \text{s.t.} \quad & x \in \arg\min_{x \in \mathcal{X}_\theta} \phi(x, \theta). \end{aligned} \tag{4.3}$$

Clearly, as hinted in Section 3.2.1, this is a bilevel optimization problem in which the lower-level problem makes sure that the x variable appearing in the upper-level cost function is a Nash equilibrium of the game induced by θ, the parameter fixed by the upper-level problem. Hence, by using the predictability loss we search for a parameter that induces an equilibrium whose actions are as close as possible to the actions shared by player j through the feedback, in the sense of the (squared) Euclidean norm. In order to untie this hierarchical structure and write the problem as a single-level optimization problem, we reformulate the lower-level problem using the KKT conditions as in Section 2.2.2. Defining $g(x,\theta) = [g^i(x,\theta^i)]_{i\in\mathcal{N}}^\top \in \mathbb{R}^{mN}$ and recalling that $\mathcal{X}_\theta = \{x \in \mathcal{X} : g^i(x,\theta^i) \leq 0, \quad \forall i \in \mathcal{N}\}$, we can write:

$$\begin{aligned} \hat{\theta}_{k+1}^{ij} = \arg\min_{\theta\in\Theta,x,\lambda} \quad & \frac{1}{2}\mathcal{R}(\theta^j) + \eta_k \|x_f^j - x\|_2^2 \\ \text{s.t.} \quad & \nabla_x \phi(x,\theta) + \lambda^\top \nabla_x g(x,\theta) = 0, \\ & \lambda^\top g(x,\theta) = 0, \\ & x \in \mathcal{X}_\theta, \\ & \lambda \geq 0, \end{aligned} \tag{4.4}$$

where $\lambda \in \mathbb{R}_{\geq 0}^{mN}$ is the vector of dual variables of the lower-level problem. As pointed out in Section 2.2.2, the complementarity constraints of the KKT conditions render the problem nonconvex. One way to obtain an equivalent, linearized formulation of the complementarity constraints is by introducing a vector of binary variables called *active-set* $Z \in \{0,1\}^{mN}$. We obtain:

$$\begin{aligned} \hat{\theta}_{k+1}^{ij} = \arg\min_{\theta\in\Theta,x,\lambda,Z} \quad & \frac{1}{2}\mathcal{R}(\theta^j) + \eta_k \|x_f^j - x\|_2^2 \\ \text{s.t.} \quad & \nabla_x \phi(x,\theta) + \lambda^\top \nabla_x g(x,\theta) = 0, \\ & g(x,\theta) \geq M(Z-1), \\ & \lambda \leq MZ, \\ & \lambda \geq 0, \\ & x \in \mathcal{X}_\theta, \quad Z \in \{0,1\}^{mN}, \end{aligned} \tag{4.5}$$

where $M \in \mathbb{R}$ is an appropriately big number. This is a mixed integer second order conic program (MISOCP). In general, solving such a problem is an NP-hard task, and it is left to solvers such as Gurobi. However, there exist some (purely theoretic) cases where this problem can be recast as a simple convex problem with continuous variables. In our work, this happens when we consider a *perfect feedback model*. Essentially, from such a feedback we can directly compute the active-set Z, which ceases to be an optimization variable and allows us to resolve the complementarity in the constraints. By solving the resulting optimization problem, we look for a parameter θ that induces a Nash equilibrium x that is as close as possible to the Nash equilibrium computed by the opponent, in the (squared) 2-norm sense, and has the same active-set.
A further simplification made possible by a noiseless feedback consists in dropping the x optimization variable, since we already know the equilibrium we are looking for. This also implies removing the loss part from the utility function:

$$\begin{aligned} \hat{\theta}_{k+1}^{ij} = \arg\min_{\theta\in\Theta,\lambda} \quad & \frac{1}{2}\mathcal{R}(\theta^j) \\ \text{s.t.} \quad & \nabla_x \phi(x_f^j,\theta) + \lambda^\top \nabla_x g(x_f^j,\theta) = 0, \\ & g(x_f^j,\theta) \geq M(Z_f-1), \\ & \lambda \leq MZ_f, \\ & \lambda \geq 0, \end{aligned} \tag{4.6}$$

where we use Z_f to indicate that the active-set is computed from the feedback and is no longer subject to optimization.
To conclude, the predictability loss allows to estimate uncertain parameters in the objective functions or in the constraints, however, it is hardly applicable in practice.

Before going on, we need to restrict our attention to a subclass of the wider class of convex potential games in Problem 3.2, which is the one of strictly convex potential games with quadratic potential ϕ. Moreover, we only consider situations where the uncertainty is in the objective functions:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & \frac{1}{2} x^\top Q x + q^\top x \\ \text{s.t.} \quad & g(x) \leq 0, \end{aligned} \tag{4.7}$$

where $\theta = (Q, q)$ and $Q \succ 0$ and q are matrices of appropriate dimensions. In Appendix A we briefly discuss the case of linear potential.

Suboptimality loss. First of all, we notice that when focusing on quadratic functions the suboptimality loss in Equation 3.6 reduces to

$$\begin{aligned} l^s_\theta(x) &= \phi(x, \theta) - \min_{y \in \mathcal{X}} \phi(y, \theta) \\ &= \max_{y \in \mathcal{X}} \phi(x, \theta) - \phi(y, \theta), \end{aligned} \tag{4.8}$$

where $\mathcal{X}$ is no longer dependent on θ. Moreover, the uncertainty set Θ accounting for prior information should exclude $\theta = (0, 0)$, as this choice of the parameter would render every action in $\mathcal{X}$ optimal.
When using the suboptimality loss, the update rule reads

$$\hat{\theta}^{ij}_{k+1} = \arg\min_{\theta \in \Theta} \frac{1}{2} \mathcal{R}(\theta^j) + \eta_k \max_{y \in \mathcal{X}} \phi(x^j_f, \theta) - \phi(y, \theta). \tag{4.9}$$

Note that, in this program, a hierarchical structure is still present. We can formulate the program in single-level form by exploiting *Lagrange duality*, obtaining

$$\hat{\theta}^{ij}_{k+1} = \arg\min_{\theta \in \Theta, \lambda \geq 0} \frac{1}{2} \mathcal{R}(\theta^j) + \eta_k (\phi(x^j_f, \theta) - \mathcal{L}(\lambda)), \tag{4.10}$$

where $\mathcal{L}(\lambda)$ is the *Lagrangian dual function*, which is defined as the minimum over y of the Lagrangian $L(y, \lambda)$ of the loss function. For more details on this derivation, we refer to Appendix A. Since $\mathcal{L}$ is the minimum of a linear function w.r.t. λ, it is concave, and the problem we obtain is convex. Hence, unlike what happens with the predictability loss, its solution does not require computing an active-set, and it can be solved even for noisy and partial feedback.

First-order loss. When using the first-order loss, the update rule reads

$$\hat{\theta}^{ij}_{k+1} = \arg\min_{\theta \in \Theta} \frac{1}{2} \mathcal{R}(\theta^j) + \eta_k \max_{y \in \mathcal{X}} \nabla_x \phi(x^j_f, \theta)^\top (x^j_f - y), \tag{4.11}$$

where Θ does not include $\theta = (0, 0)$. Again, using Lagrange duality, it can be reformulated into

$$\hat{\theta}^{ij}_{k+1} = \arg\min_{\theta \in \Theta, \lambda \geq 0} \frac{1}{2} \mathcal{R}(\theta^j) + \eta_k (\nabla_x \phi(x^j_f, \theta)^\top x^j_f - \mathcal{L}(\lambda)). \tag{4.12}$$

We leave the details of the derivation to Appendix A. As it happens with the suboptimality loss, the first-order loss can handle imperfect feedback.

Remark 2. *If the constraints $g(x) \leq 0$ have a linear structure, i.e., they can be written as $Ax \leq b$, in order to write the problem as a single-level program we can use* LP duality*. For the first-order loss case, we would obtain*

$$\begin{aligned} \hat{\theta}^{ij}_{k+1} = \arg\min_{\theta\in\Theta,\lambda} \quad & \frac{1}{2}\mathcal{R}(\theta^j) + \eta_k(\nabla_x\phi(x^j_f,\theta)^\top x^j_f - b^\top\lambda) \\ s.t. \quad & A^\top\lambda \geq \nabla_x\phi(x^j_f,\theta)^\top, \\ & \lambda \geq 0. \end{aligned} \tag{4.13}$$

The details of the derivation are left to Appendix A.

4.2 Multi-agent active-set algorithm

As we discussed above, Algorithm 1 presents some limitations as, in practice, it can only be applied to cases where the uncertainties are in the objective functions. We now propose a second algorithm, adapted from [6], which takes a different approach for solving our learning problem and which can handle uncertainty in the constraints, as well as in the objectives. In [6], the authors consider the point of view of an external regulator whose goal is to learn the parameters of agents in a potential game, and their algorithm works in batch setting, as they assume availability of a dataset containing the Nash equilibria of past agents' interactions. We drop all these assumptions: firstly, we look at the problem from the perspective of the agents participating in the game; secondly, our algorithm works in an online setting, with data arriving sequentially at each round; lastly, these data observed by the agents are not necessarily Nash equilibria of the underlying game. As in [6], we restrict our range of applications to situations where each agent optimizes a quadratic objective J^i subject to polyhedral constraints g^i and the resulting potential game has a quadratic potential ϕ. Problems 3.1 and 3.2 become

$$\begin{aligned} \min_{x^i\in\mathcal{X}^i} \quad & J^i(x,\theta^i) \\ \text{s.t.} \quad & A^i(\theta^i)x \leq b^i(\theta^i), \end{aligned} \tag{4.14}$$

and

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \frac{1}{2}x^\top Q(\theta)x + q(\theta)^\top x \\ \text{s.t.} \quad & A^i(\theta^i)x \leq b^i(\theta^i), \quad \forall i \in \mathcal{N} \end{aligned} \tag{4.15}$$

where $A^i : \mathbb{R}^p \to \mathbb{R}^{m\times nTN}$, $b^i : \mathbb{R}^p \to \mathbb{R}^m$, $Q : \mathbb{R}^{pN} \to \mathbb{R}^{nTN\times nTN}$ and $q : \mathbb{R}^{pN} \to \mathbb{R}^{nTN}$.
Before we present the algorithm, let's define some notation that will be needed. First, we define the global constraint matrices $A(\theta)$ and $b(\theta)$ such that

$$A(\theta)x \leq b(\theta) \quad \Leftrightarrow \quad A^i(\theta^i)x \leq b^i(\theta^i), \quad \forall i \in \mathcal{N}. \tag{4.16}$$

Then, we indicate with $Z^{(k)}$ and $Y^{(k)}$ the active-sets at round $k \in \{0,\dots,K-1\}$. Although their meaning is the same as the active-set introduced in the previous section, we define them slightly differently as the sets of indices of tight and loose constraints, respectively: $Z^{(k)} \subseteq \{1,\dots,mN\}$ and $Y^{(k)} \subseteq \{1,\dots,mN\}$. Moreover, as in [6], we use $x^{(k)}_{Z^{(k)}}(\theta)$ to indicate a Nash equilibrium of the game given $Z^{(k)}$. More precisely, $x^{(k)}_{Z^{(k)}}(\theta)$ is obtained *implicitly* as the solution of $F(x^{(k)},\lambda^{(k)}) = 0$, where

$$F(x^{(k)},\lambda^{(k)}) = \begin{bmatrix} Q(\theta)x^{(k)} + q(\theta) + A(\theta)^\top\lambda^{(k)} \\ b(\theta)_{Z^{(k)}} - A(\theta)_{Z^{(k)}}x^{(k)} \\ \lambda_{Y^{(k)}} \end{bmatrix}, \tag{4.17}$$

where the subscripts $Z^{(k)}$ and $Y^{(k)}$ indicate the set of constraints with indices contained in the active-sets. We can now reformulate the predictability loss in Equation 3.5 as a function of the active-set $Z^{(k)}$ as

$$l_\theta^p(Z^{(k)}, x) := \min_y \|x - y_{Z^{(k)}}\|_2^2. \tag{4.18}$$

We are now ready to state the algorithm:

Algorithm 2 Multi-agent active-set learning

Input θ_0^i for all $i \in \mathcal{N}$

for $k = 0$ to $K-1$ **do**
  for all players $i \in \mathcal{N}$ **do**
    for all players $j \in \mathcal{N} \setminus \{i\}$ **do**
      receive x_f^j
      $x^{(k)}(\theta_k^i), \lambda^{(k)}(\theta_k^i) \leftarrow$ primal and dual variables from $\mathcal{G}(\theta_k^i)$
      $Z^{(k)} \leftarrow \{z : b(\theta_k^i) - A(\theta_k^i)x^{(k)}(\theta_k^i) = 0\}$
      $Y^{(k)} \leftarrow \{z : \lambda^{(k)}(\theta_k^i) = 0\}$
      set $l_k(x) \leftarrow l_\theta^p(Z^{(k)}, x_f^j)$
      $\nabla_\theta l_k(x^{(k)}(\theta_k^i)) = \nabla_\theta(x^{(k)}(\theta_k^i))^\top \nabla_x l_k(x^{(k)}(\theta_k^i))$
      update $\hat{\theta}_{k+1}^{ij} \leftarrow \Pi_{\Theta^{ij}}\{\hat{\theta}_k^{ij} - \eta_k \nabla_\theta l_k(x^{(k)}(\theta_k^i))\}$
    end for
  end for
end for

As we have seen above, when an agent i tries to update the estimate of another agent j's parameter using the predictability loss as a discrepancy measure, it gets to the MISOCP in Equation 4.5, which with our newest assumptions and definitions can be equivalently written as:

$$\begin{aligned} \hat{\theta}_{k+1}^{ij} = \arg \min_{\theta \in \Theta, x, \lambda, Z} \quad & \|x_f^j - x\|_2^2 \\ \text{s.t.} \quad & Q(\theta)x + q(\theta) + A(\theta)^\top \lambda = 0, \\ & b(\theta) - A(\theta)x \geq 0, \\ & \lambda \geq 0, \\ & b(\theta)_Z - A(\theta)_Z x = 0, \\ & \lambda_Y = 0, \\ & x \in \mathcal{X}_\theta, \quad Z \subseteq \{1, \ldots, mN\}, \end{aligned} \tag{4.19}$$

where Y is the complement of Z. Algorithm 2 provides a way of solving this program without assuming knowledge of the correct active-set, by exploiting an online sequence of "estimated active-sets" $Z^{(k)}$, for $k = 0, \ldots, K-1$. At each step k, the vector of estimates θ_k^i of player i induces a Nash equilibrium $x^{(k)}(\theta_k^i)$, which is associated to an active-set $Z^{(k)}$. Given this set and the feedback x_f^j, the agent can compute the gradient of the loss function w.r.t. θ and exploit this information to update player j's parameter estimate using gradient descent projected onto the space of possible parameters Θ^{ij}. A key passage to pay attention to regards the computation of the gradient of the implicit function $x^{(k)}(\theta_k^i)$, which can be done by applying the following theorem [6, 28]:

Theorem 1. *(Dini classical implicit function theorem) Let $G(z)$ be implicitly defined by the relation $H(x, z) = 0$ for $H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Assume that (1) H is continuously differentiable in a neighborhood of (x_0, z_0), (2) $H(x_0, z_0) = 0$ and (3) $\nabla_x H(x_0, z_0)$ is nonsingular. Then, G is continuously differentiable in a neighborhood of (x_0, z_0) with Jacobian given by*

$$\nabla_z G(z) = \nabla_x H(x_0, z_0)^{-1} \nabla_z H(x_0, z_0). \tag{4.20}$$

Note that, as for Algorithm 1, the loss function and in particular its gradient are evaluated using the complete estimated vector θ_k^i, but only one parameter at a time gets updated with the update rule. Moreover, as opposed to Algorithm 1, at each round k player i only estimates the Np parameters corresponding to the entries of θ_k^i.

Remark 3. *(Degenerate active-sets) Note that, in Algorithm 2, the active-set $Y^{(k)}$ is not computed as the complement of $Z^{(k)}$, hence it may occur that $Z^{(k)} \cap Y^{(k)} \neq \emptyset$ for some realizations of $(x^{(k)}(\theta_k^i), \lambda^{(k)}(\theta_k^i))$. In this case, we say that the active-sets are* degenerate*. Degenerate active-sets prevent us from computing the gradient of the loss function as the matrix corresponding to H in Equation 4.20 is not invertible. When this happens, we select non-degenerate active-sets by using the following rule, introduced and motivated in [6]: let $W^{(k)} = Z^{(k)} \cap Y^{(k)}$ and partition it randomly and uniformly into two sets, $W_1^{(k)}$ and $W_2^{(k)}$. Then, define $Z_{new}^{(k)} = Z^{(k)} \setminus W_1^{(k)}$ and $Y_{new}^{(k)} = Y^{(k)} \setminus W_2^{(k)}$.*

Chapter 5

Applications

The framework we propose is quite general and many applications can be tailored to it. Inspired by [13], in which the authors apply receding horizon games to the problem of demand-side management, we also address the energy market and, in particular, we take the consumers' perspective and consider the task of planning electricity consumption profiles over T hours of the day, that satisfy their energy needs while taking into account constraints that could come, for example, from the electricity providers, who then are not left completely out of the equation. The data we utilize are based on residential electricity consumption data of the Italian population in 2020 taken from [16–18].

5.1 Electricity market

5.1.1 Modelling

Individual decision making problem

We consider a situation where N electricity consumers want to plan their electricity consumption for the next T hours. Each consumer $i \in \mathcal{N} = \{1, \dots, N\}$ selects actions $x^i = [x_0^i, \dots, x_{T-1}^i]^\top \in \mathcal{X}^i \subseteq \mathbb{R}^T$, where $x_t^i \in \mathcal{X}_t^i \subseteq \mathbb{R}$ represents the planned consumption at time t, for all $t \in \mathcal{T} = \{0, \dots, T-1\}$.

Objective function. We assume that each agent has a desired planned consumption profile $x^{i*} = [x_0^{i*}, \dots, x_{T-1}^{i*}]^\top$. The local stage objective function for each consumers is

$$J_t^i(x_t, \theta_t^i) = w\|x_t^i - x_t^{i*}\|_2^2 + \sigma_t(L_t)x_t^i, \tag{5.1}$$

where $w \in \mathbb{R}$ is a constant weight, $\sigma_t(L_t) = 2s_tL_t$ is the price of electricity, s_t is a positive constant representing the price rate, and $L_t(x_t) = \sum_{j\in\mathcal{N}} x_t^j$ is the total energy to be consumed at time t. By considering the complete time horizon, agent i optimizes

$$\begin{aligned} J^i(x, \theta^i) &= \sum_{t\in\mathcal{T}} J_t^i(x_t, \theta_t^i) \\ &= \sum_{t\in\mathcal{T}} w\|x_t^i - x_t^{i*}\|_2^2 + \sigma_t(L_t)x_t^i. \end{aligned} \tag{5.2}$$

By writing the stage objective as

$$\begin{aligned} J_t^i(x_t, \theta_t^i) &= (w + 2s_t)x_t^{i2} + 2s_t(\sum_{j\in\mathcal{N}\setminus\{i\}} x_t^j)x_t^i - 2wx_t^{i*}x_t^i + wx_t^{i*2} \\ &= (w + 2s_t)x_t^{i2} + [2s_t(\sum_{j\in\mathcal{N}\setminus\{i\}} x_t^j) - 2wx_t^{i*}]x_t^i + wx_t^{i*2}, \end{aligned} \tag{5.3}$$

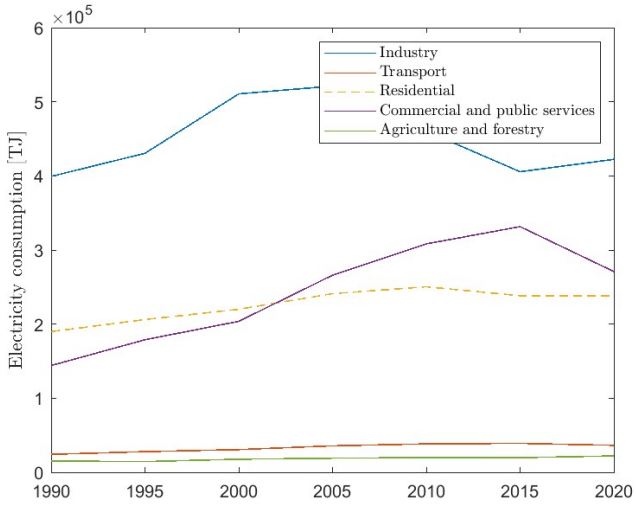


Figure 5.1: Electricity consumption by sector, Italy 1990-2020 [16]

we can write the objective cost of agent i as

$$J^i(x, \theta^i) = \frac{1}{2} x^{i\top} Q^i x^i + q^{i\top} x^i + c^i, \tag{5.4}$$

where

$$Q^i = \begin{bmatrix} w + 2s_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w + 2s_{T-1} \end{bmatrix} \in \mathbb{R}^{T \times T}, \tag{5.5}$$

$$q^i = \begin{bmatrix} 2s_0 L_0(x_0^{-i}) - 2w x_0^{i*} \\ \vdots \\ 2s_{T-1} L_{T-1}(x_{T-1}^{-i}) - 2w x_{T-1}^{i*} \end{bmatrix} \in \mathbb{R}^T, \tag{5.6}$$

$$c^i = w \sum_{t \in \mathcal{T}} x_t^{i*2} \in \mathbb{R}. \tag{5.7}$$

Constraints. We assume that consumers have a baseline hourly consumption representing, e.g., the energy needed for domestic appliances, and a maximum consumption: $\mathcal{X}_t^i = [\underline{x}^i, \overline{x}^i]$, for all $t \in \mathcal{T}$, where $\overline{x}^i > \underline{x}^i \geq 0$ are the hourly consumption bounds. Apart from this, in our analysis, we consider the following set of individual linear constraints, representing the fact that each agent has minimum and maximum bounds on the total electricity that can be consumed in the horizon:

$$\underline{l}^i \leq \sum_{t \in \mathcal{T}} x_t^i \leq \overline{l}^i, \tag{5.8}$$

where $\overline{l}^i > \underline{l}^i \geq 0$. Besides this set of constraints, we will also consider global coupling constraints, but only at a stage of the game where consumers have applied the learning algorithms and understood others' preferences. In particular, we enforce that the aggregate load at each time step, in the long term, is constrained by

$$\underline{L}_t \leq \sum_{i \in \mathcal{N}} x_t^i \leq \overline{L}_t, \quad \forall t \in \mathcal{T} \tag{5.9}$$

where $\overline{L}_t > \underline{L}_t \geq 0$ for all $t \in \mathcal{T}$.
The decision making problem of each agent i when only individual constraints are considered is

$$\begin{aligned} \min_{x^i \in \mathcal{X}^i} \quad & \frac{1}{2} x^{i\top} Q^i x^i + q^{i\top} x^i + c^i \\ \text{s.t.} \quad & A_u^i x^i \leq b_u^i, \end{aligned}, \quad \forall i \in \mathcal{N} \tag{5.10}$$

where

$$A_u^i = \begin{bmatrix} \mathbb{1}_T^\top \\ -\mathbb{1}_T^\top \end{bmatrix} \in \mathbb{R}^{2\times T}, \quad b_u^i = \begin{bmatrix} \overline{l}^i \\ -\underline{l}^i \end{bmatrix} \in \mathbb{R}^2. \tag{5.11}$$

The decision making problem of each agent i when also the coupling constraints are taken into account is

$$\begin{aligned} \min_{x^i \in \mathcal{X}^i} \quad & \frac{1}{2} x^{i\top} Q^i x^i + q^{i\top} x^i + c^i \\ \text{s.t.} \quad & A^i x^i \leq b^i(x^{-i}), \end{aligned}, \quad \forall i \in \mathcal{N} \tag{5.12}$$

where

$$A^i = \begin{bmatrix} A_u^i \\ A_c^i \end{bmatrix} \in \mathbb{R}^{(2+2T)\times T}, \quad b^i(x^{-i}) = \begin{bmatrix} b_u^i \\ b_c^i(x^{-i}) \end{bmatrix} \in \mathbb{R}^{2+2T} \tag{5.13}$$

$$A_c^i = \begin{bmatrix} \mathbf{I}_T \\ -\mathbf{I}_T \end{bmatrix} \in \mathbb{R}^{2T\times T}, \quad b_c^i(x^{-i}) = \begin{bmatrix} \overline{L}_0 - \sum_{j\in\mathcal{N}\setminus\{i\}} x_0^j \\ \vdots \\ \overline{L}_{T-1} - \sum_{j\in\mathcal{N}\setminus\{i\}} x_{T-1}^j \\ -\underline{L}_0 + \sum_{j\in\mathcal{N}\setminus\{i\}} x_0^j \\ \vdots \\ -\underline{L}_{T-1} + \sum_{j\in\mathcal{N}\setminus\{i\}} x_{T-1}^j \end{bmatrix} \in \mathbb{R}^{2T}. \tag{5.14}$$

Note that, in both cases, the optimization problems of the individuals are QPs. Moreover, while the matrices A_u^i and A_c^i are constant and $A_u^i = A_u^j$, $A_c^i = A_c^j$ for all $i, j \in \mathcal{N}$, the vectors b_u^i and b_c^i are in general different for all consumers.

Potential game

The game resulting from the interaction of the N consumers, each optimizing a cost of the form 5.4, is potential. By defining $x_t = [x_t^1, \ldots, x_t^N]^\top$ for all $t \in \mathcal{T}$ and $x = [x_0^\top, \ldots, x_{T-1}^\top]^\top$ we can write the potential function as

$$\phi(x) = \frac{1}{2} x^\top Q x + q^\top x + c, \tag{5.15}$$

where the matrices are defined as follows:

$$Q = 2\begin{bmatrix} W_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_{T-1} \end{bmatrix} \in \mathbb{R}^{NT\times NT}, \tag{5.16}$$

$$W_t = \begin{bmatrix} w+2s_t & \cdots & s_t \\ \vdots & \ddots & \vdots \\ s_t & \cdots & w+2s_t \end{bmatrix} \in \mathbb{R}^{N\times N}, \quad \forall t \in \mathcal{T} \tag{5.17}$$

$$q = -2w\begin{bmatrix} x_0^* \\ \vdots \\ x_{T-1}^* \end{bmatrix} \in \mathbb{R}^{NT}, \tag{5.18}$$

$$c = \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} x_t^{i*2} \in \mathbb{R}. \tag{5.19}$$

The Nash equilibrium of the potential game corresponds to the solution of the following minimization problem:

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \frac{1}{2}x^\top Qx + q^\top x + c \\ \text{s.t.} \quad & Ax \le b, \end{aligned} \tag{5.20}$$

where in the uncoupled case the constraint matrices are defined as

$$A = \begin{bmatrix} \mathbf{I}_{N\times N} & \cdots & \mathbf{I}_{N\times N} \\ -\mathbf{I}_{N\times N} & \cdots & -\mathbf{I}_{N\times N} \end{bmatrix} \in \mathbb{R}^{2N\times NT}, \quad b = \begin{bmatrix} \bar{l}\mathbb{1}_N \\ -\underline{l}\mathbb{1}_N \end{bmatrix} \in \mathbb{R}^{2N}, \tag{5.21}$$

while when taking coupling constraints into account they become

$$A = \begin{bmatrix} \mathbf{I}_{N\times N} & \cdots & \mathbf{I}_{N\times N} \\ -\mathbf{I}_{N\times N} & \cdots & -\mathbf{I}_{N\times N} \\ \mathbb{1}_N^\top & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbb{1}_N^\top \\ -\mathbb{1}_N^\top & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\mathbb{1}_N^\top \end{bmatrix} \in \mathbb{R}^{(2N+2T)\times NT}, \quad b = \begin{bmatrix} \bar{l}\mathbb{1}_N \\ -\underline{l}\mathbb{1}_N \\ \overline{L}_0 \\ \vdots \\ \overline{L}_{T-1} \\ -\underline{L}_0 \\ \vdots \\ -\underline{L}_{T-1} \end{bmatrix} \in \mathbb{R}^{2N+2T}, \tag{5.22}$$

and

$$\bar{l} = \begin{bmatrix} \bar{l}^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \bar{l}^N \end{bmatrix} \in \mathbb{R}^{N\times N}, \quad \underline{l} = \begin{bmatrix} \underline{l}^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \underline{l}^N \end{bmatrix} \in \mathbb{R}^{N\times N}. \tag{5.23}$$

5.1.2 Numerical analysis

We will take into consideration the individual constraints case, where the optimization problems are coupled only in the objective function. We define each consumer's parameter vector $\theta^i := (x^{i*\top}, l^{i\top})^\top$, where $l^i = (\bar{l}^i, \underline{l}^i)^\top$, and the complete parameter vector $\theta := (x^{*\top}, l^\top)$. Moreover, we assume that $s = [s_t]_{t\in\mathcal{T}}$ is an external signal, broadcast to every consumer at the beginning of each iteration $k \in \mathcal{K}$ of the algorithm, and that $s_{t_1} = s_{t_2}$ for all $t \in \mathcal{T}$. Hence, we have:

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \frac{1}{2}x^\top Q(s)x + q(\theta)^\top x + c(\theta) \\ \text{s.t.} \quad & Ax \le b(\theta). \end{aligned} \tag{5.24}$$

We consider a population of $N = 7$ consumers of electricity planning their consumption over an horizon of $T = 2, 5, 10$ hours, and applying $a = 1$ actions of the planned profile before calculating a new one. We assume the algorithm is run once per hour over a period of $K = 72$ hours, and that the price rate s_t varies at each k following a normal distribution with mean $0.015€/kWh$ and variance $0.005€/kWh$. For each experiment, we generate a test dataset $\mathcal{D}_{test}$ containing $D = 30$ datapoints each consisting of a pair $(x_d^\star, s_d)$, for $d = 1, \dots, 30$, and, for each agent i, we evaluate the performance of Algorithms 1 and 2 using the following prediction error, as in [6], quantifying the goodness of the estimated parameters in predicting the correct equilibria

$$\sqrt{(1/D) \sum_{(x_d^\star, s_d) \in \mathcal{D}_{test}} \|\hat{x}(\theta_k^i) - x_d^\star\|_2^2}. \tag{5.25}$$

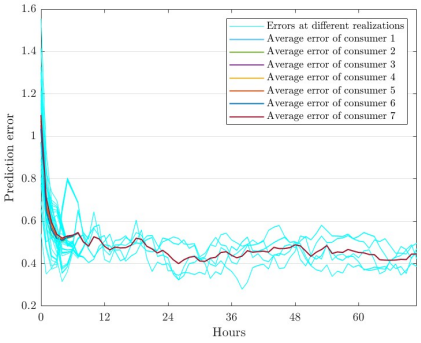
As a second performance assessment criteria, we consider the estimation error after K rounds of the game, calculated as

$$\frac{1}{p} \sum_{z=1}^{p} \|\theta^{j,z} - \hat{\theta}_{K-1}^{ij,z}\|_2^2, \tag{5.26}$$

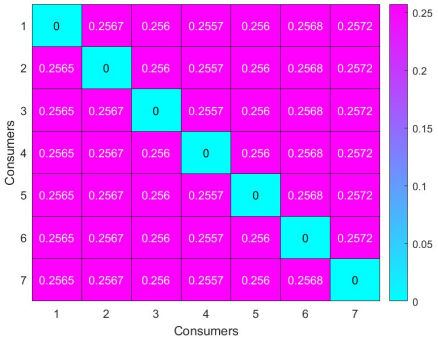
for each pair of consumers $i, j \in \mathcal{N}$, where $\theta^{j,z}$ is the z-th element of $\theta^j \in \mathbb{R}^p$.

Uncertainty in the objective functions

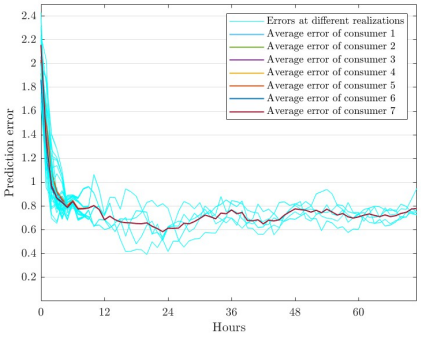
Assume $\theta = (x^{*\top})$. When the uncertainty regards parameters of the utility functions, in general we obtained good results using both our algorithms. All the results we showcase here are averaged over 5 independent realizations of the experiments.
Let's first utilize Algorithm 1. Figure 5.2 shows the performances of the algorithm when perfect feedback is exchanged between agents, and the predictability loss is considered. As we can see, on average, a shorter prediction horizon favours the estimation of sets of parameters which are better in predicting the opponents' behavior, but are less close to the correct ones. The same behavior is evident also in Figure 5.3, where the suboptimality loss is employed. Looking at the perfect feedback case, the suboptimality loss has performances comparable to the predictability loss. However, it is also able to handle noise in the feedback, which we introduced in the form of independent Gaussian noise with zero mean and $\sigma^2 = 0.1$ variance. We notice that, when noise comes into place, the predictive power of the algorithm doesn't change much, being it almost the same in terms of both the error value and the convergence speed, however, the estimation of the parameters is, on average, slightly worse than the noiseless case. Figure 5.4 depicts the results when the first-order loss is utilized. This loss is the one with the worst performances, both in terms of the estimation of the true parameters and of the prediction error, which doesn't reach convergence and keeps oscillating even after 72 iterations of the algorithm.
Figures 5.5 and 5.6 shows the performances of Algorithm 2. In general, we noticed that when the consumers exchange a partial feedback of type 1 (see Section 3.2.2), even if the estimates converge (the gradient of the loss function employed by the algorithm tends to zero), they are not able to reach satisfying predictive performances, for every horizon length T and every number of steps $t \in \mathcal{T}$ of information exchanged. Instead, a partial feedback of type 2 doesn't influence the final result and eventually consumers are able to reach perfect knowledge of the unknown parameters, for every T, which is also true when the feedback is noisy, although in this case the convergence rate is much lower with the learning rates that we utilized. This behavior shown by partial feedback of type 1 and 2 might suggest that the consumers need at least one piece of information regarding the full length of the prediction horizon, which seems more valuable than many more data encompassing only a portion of the horizon. With respect to Algorithm 1, we can see how the performances of this second algorithm are preferable, allowing the consumers to reach much better results at the same conditions, and that the estimation errors are less uniform when using



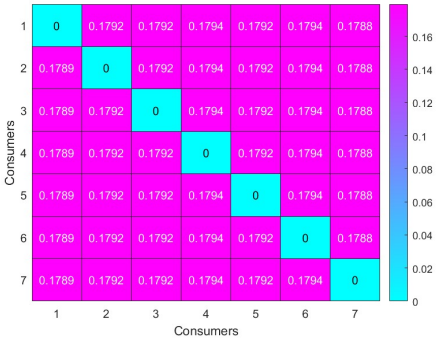
(a) T = 2, perfect feedback



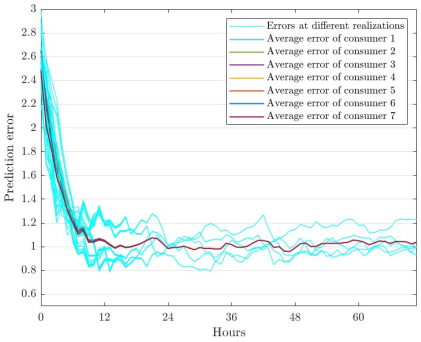
(b) T = 2, perfect feedback



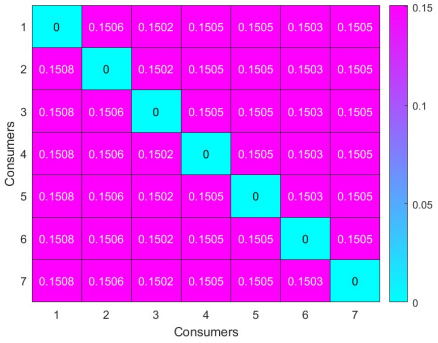
(c) T = 5, perfect feedback



(d) T = 5, perfect feedback

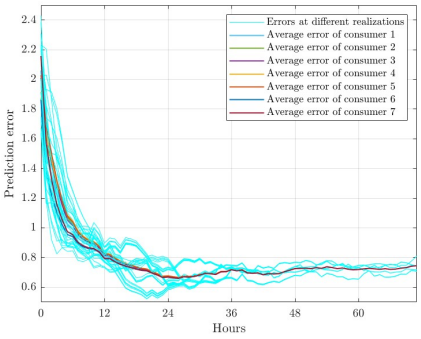


(e) T = 10, perfect feedback

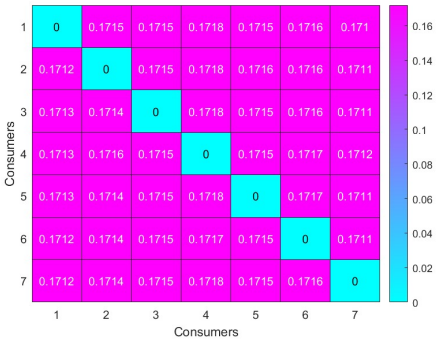


(f) T = 10, perfect feedback

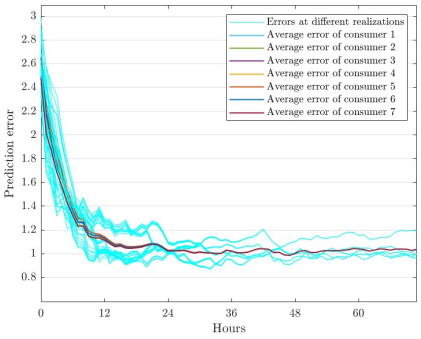
Figure 5.2: Prediction and estimation errors when using Algorithm 1 with the predictability loss



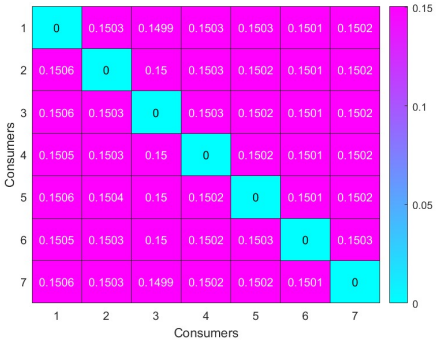
(a) T = 5, perfect feedback



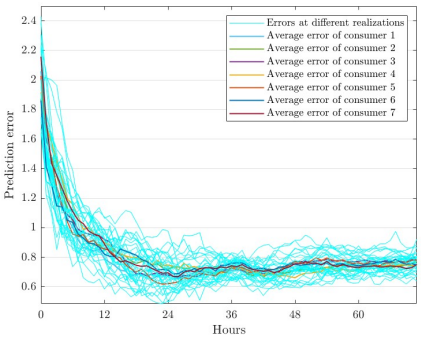
(b) T = 5, perfect feedback



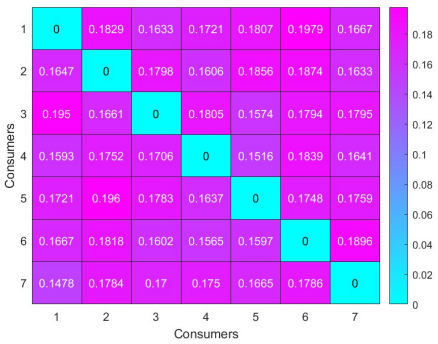
(c) T = 10, perfect feedback



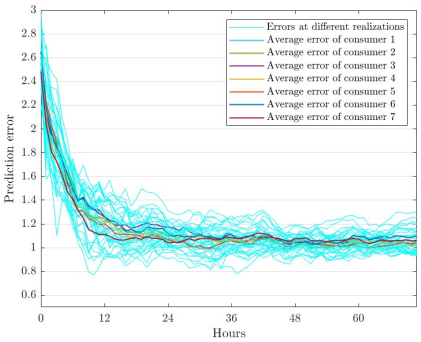
(d) T = 10, perfect feedback



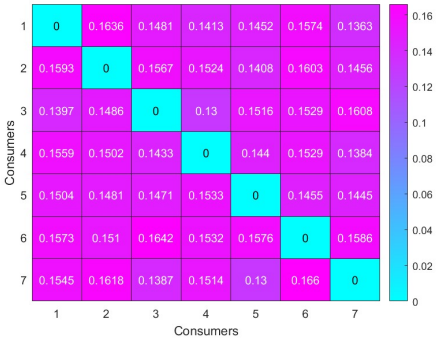
(e) T = 5, noisy feedback



(f) T = 5, noisy feedback

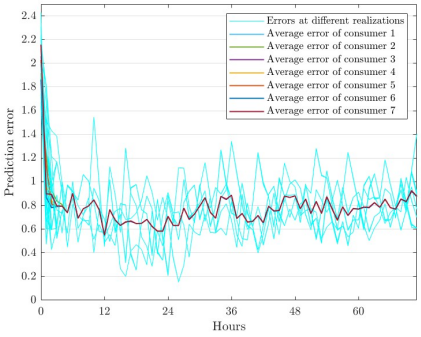


(g) T = 10, noisy feedback

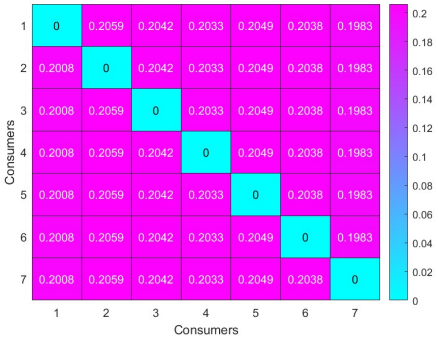


(h) T = 10, noisy feedback

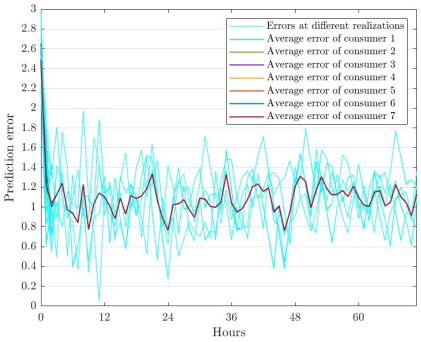
Figure 5.3: Prediction and estimation errors when using Algorithm 1 with the suboptimality loss



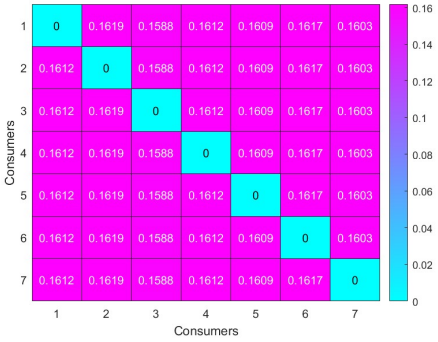
(a) T = 5, perfect feedback



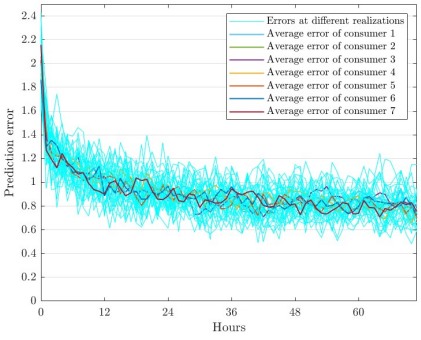
(b) T = 5, perfect feedback



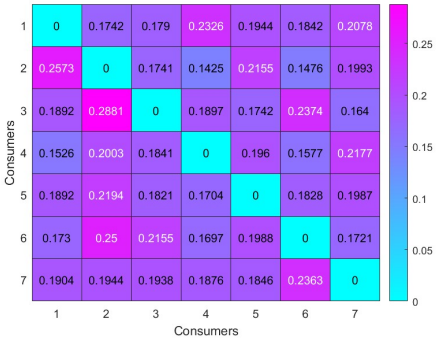
(c) T = 10, perfect feedback



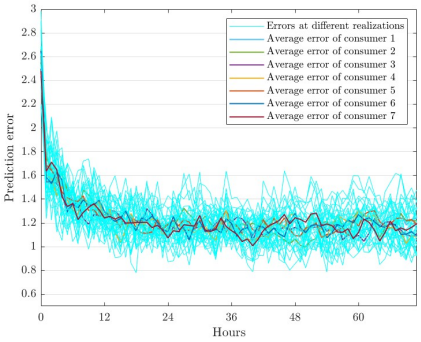
(d) T = 10, perfect feedback



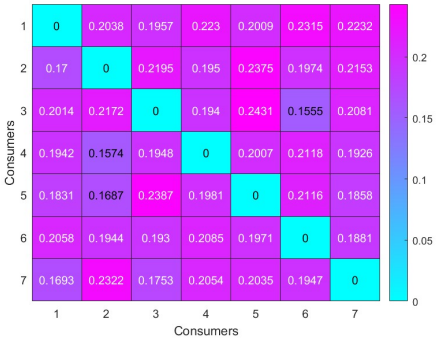
(e) T = 5, noisy feedback



(f) T = 5, noisy feedback



(g) T = 10, noisy feedback



(h) T = 10, noisy feedback

Figure 5.4: Prediction and estimation errors when using Algorithm 1 with the first-order loss

Algorithm 2, though this should not be much informative as in the original algorithm in [6], the goal is not to learn the correct sets of parameters, but to reach good predictive capabilities.

Uncertainty in the constraints

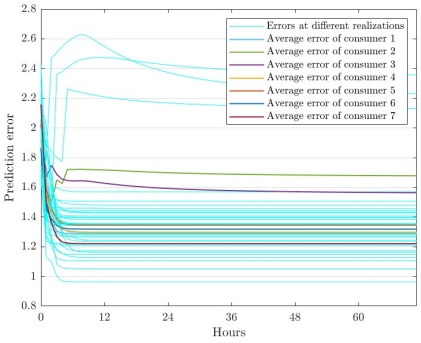
Assume $\theta = (l^\top)$. Each consumer knows the objective functions of the others, and wants to learn more about their constraints. In such cases, our results are negative. Algorithm 2, which was introduced with the purpose of learning constraints parameters, is not able to identify the correct ones, nor to predict the correct Nash equilibrium of the underlying game. For the sake of simplicity, in Figure 5.7 we present the results for a simple case with $N = 2$ consumers, $T = 2$ and a perfect feedback model, over $K = 24$ hours. In all the experiments we run, the prediction error reached convergence after a few steps, and the estimates never departed too much from the initial guesses.
To test further the approach, we also considered a case (less reasonable in practice) where the uncertainty is about parameters in the left hand side of the constraints, which is, in matrix A, obtaining the same negative results.

5.1.3 Unexpected disturbances

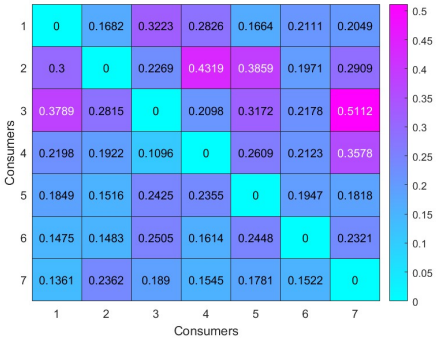
One of the main advantages of a closed-loop approach with respect to an open-loop approach is that it can identify unexpected situations or disturbances and react by adjusting the agents' decisions in real time. We demonstrate this capability by considering two examples of situations that fall into this category.

Change in the agents' preferences. We assume that, at some point k in time, a subset $\mathcal{N}_{change} \subseteq \mathcal{N}$ of the agents change their desired consumption profile. A receding-horizon approach is able to readily react and adjust the decisions to the new situation. Figure 5.8 shows a $N = 3$ situation where Consumer 1 reacts to a change in Consumers 2 and 3 preferences.

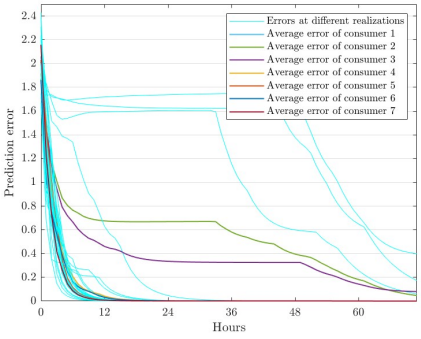
Drop in the aggregate load limit $\overline{L}_t$. For this point, we take into account also the coupling constraints in Equation 5.9. We consider $N = 7$ consumers, and utilize Algorithm 2 with $T = 10$ and a period of $K = 48$ hours. As demonstrated in Figure 5.6, when the feedback is perfect this algorithm is able to correctly learn the opponents' parameters in circa 12 hours (rounds). We assume that, at some point $k \geq 12$ in time, the aggregate load limit $\overline{L}_k$ suddenly drops by 50%, e.g. due to a line fault, for 5 hours, and show in Figure 5.9 that consumers can react efficiently to this situation by adjusting their aggregate load. Clearly, this collective behavior is guaranteed in the long term and would not hold if the drop would happen at $k < 12$.



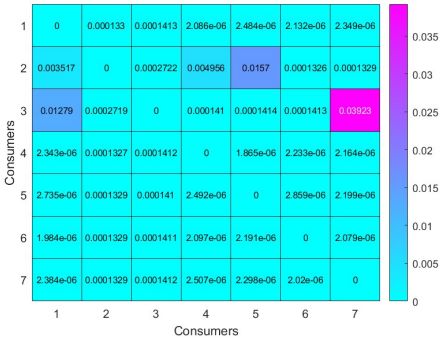
(a) T = 5, partial feedback type 1



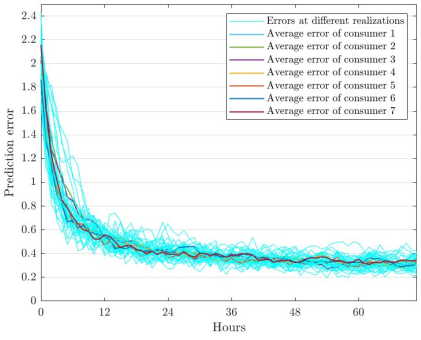
(b) T = 5, partial feedback type 1



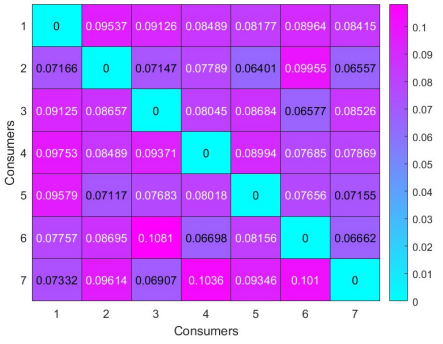
(c) T = 5, partial feedback type 2



(d) T = 5, partial feedback type 2

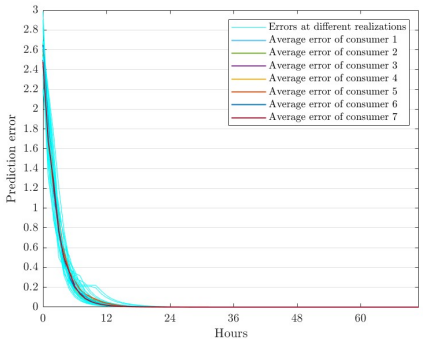


(e) T = 5, noisy, partial feedback type 2

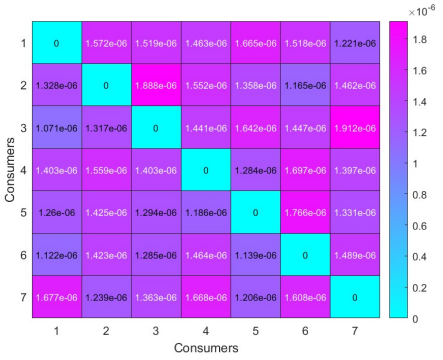


(f) T = 5, noisy, partial feedback type 2

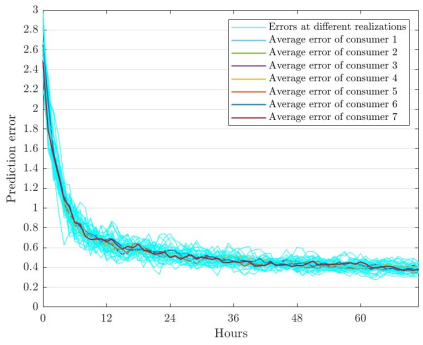
Figure 5.5: Prediction and estimation errors when using Algorithm 2, T = 5



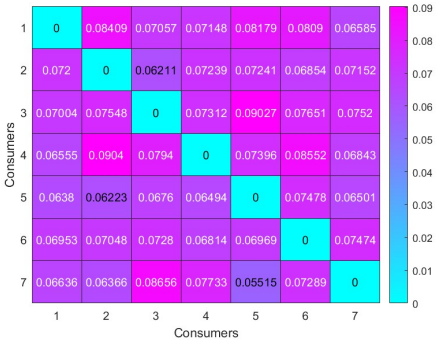
(a) T = 10



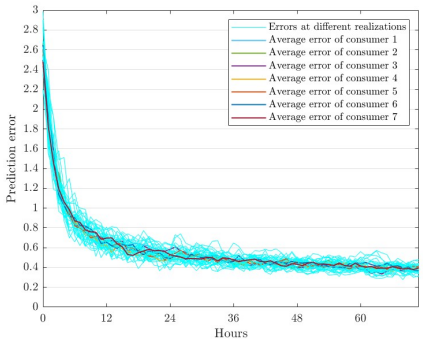
(b) T = 10



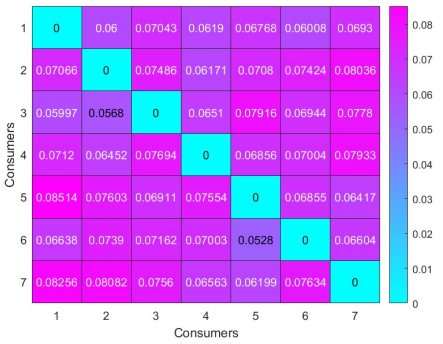
(c) T = 10, noisy feedback



(d) T = 10, noisy feedback

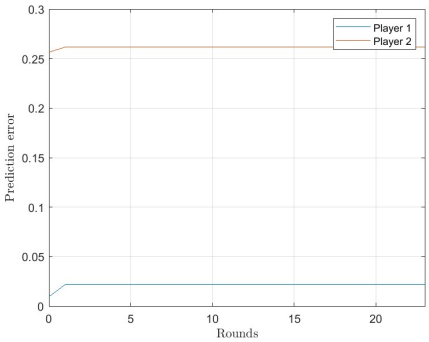


(e) T = 10, noisy, partial feedback type 2

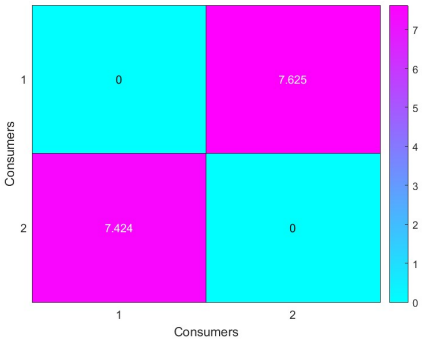


(f) T = 10, noisy, partial feedback type 2

Figure 5.6: Prediction and estimation errors when using Algorithm 2, T = 10



(a) Prediction error



(b) Estimation error

Figure 5.7: Uncertainty in the constraints right-hand side

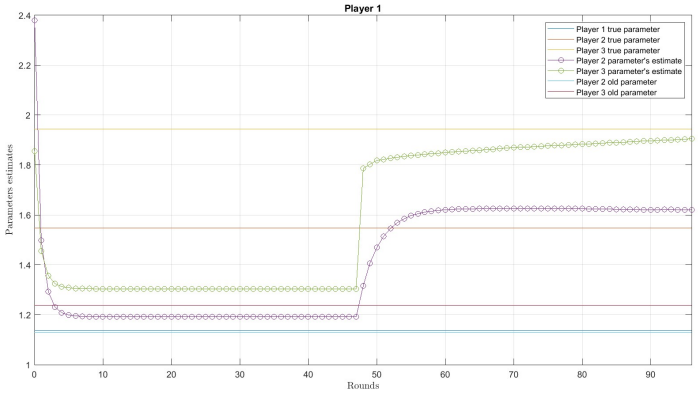


Figure 5.8: Consumer 1 adapts to a change in other consumers' preferences

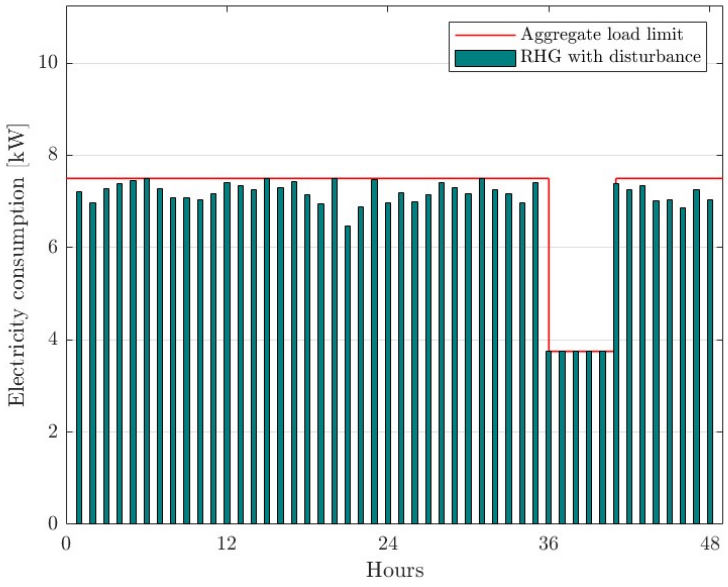


Figure 5.9: Consumers reaction to a 50% drop in the aggregate load limit $\overline{L}_k$

Chapter 6

Conclusions and outlook

In this thesis, we considered receding horizon games to model competitive situations where agents aim at learning each others' behavior in order to play optimally in the long term. We focused on the class of potential games, i.e., games for which a function exists whose value reflects the change in one agent's outcome when it changes its strategy, and we addressed in particular situations where this function, called potential, is (strictly) convex. Our setup allows for multidimensional decision variables and individual constraints, and is able to satisfy global coupling constraints in the long term.
Our main objectives were *i*) to delineate a framework for the problem of learning preferences in receding horizon games adaptable to different scenarios, and *ii*) to identify distributed algorithms to guide agents in learning the correct behavior parameters, while keeping communication between them to a minimum. Towards these goals, we outlined a general framework for receding horizon games that can handle different feedback models, and modelled the behavior of agents through a set of parameters influencing their objective functions or constraints, and we proposed two online distributed algorithms that agents can use to iteratively update their estimates of others' parameters. To conclude, our approaches have been tested on a resource planning problem regarding electricity consumption. We obtained overall good results when learning objective functions parameters, as our algorithms could learn sets of parameters able to predict opponents' actions, but our results were negative in all cases where the parameters affect the constraints.

6.1 Further research directions

First of all, our work needs to be completed with a theoretical analysis of the proposed algorithms. Indeed, we only provide numerical results, but it would be nice to try and prove some of the convergence results guaranteed in [11] and [6], the works from which we adapted our algorithms. Another aspect that could be improved regards learning rates tuning, which was not done properly during the thesis. We could complete the work by analyzing its performances when these parameters are tuned correctly.

Coupling constraints

In our work, constraints that couple the decision variables of all the agents are only satisfied in the long term. This is because, while it is true that individual agents can take coupling constraints into account when computing an equilibrium of the game, it is also the case that this equilibrium is computed locally by each agent, without any communication. Hence, when agents play actions coming from their local equilibria, the resulting set of actions is not guaranteed to satisfy the global constraints also in the true underlying game. Clearly, in the long term, when agents are able to predict others' actions locally, these constraints are satisfied. Since, in

many applications, it is of critical importance that these global constraints are always satisfied, the work could be completed by guaranteeing global constraints satisfaction at any time, maybe using a method inspired by [34], assuming the existence of multiplier variables that agents can exchange to guarantee these constraints.

Non-potential games

While Algorithm 2 is tailored to potential games, Algorithm 1 could be also applied to games that are not potential. Clearly, while this would be a simpler extension when considering the suboptimality or the first-order losses, the predictability loss is measured with respect to the optimal set, thus requiring to compute an equilibrium of the game. Since at each iteration of the algorithm, $N(N-1)$ games have to be solved, this is much more efficient when these games are potential.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the 21st International Conference on Machine Learning* (2004).

[2] Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch. Paschalidis. "Data-Driven Estimation in Equilibrium using Inverse Optimization". In: *Mathematical Programming* 153 (2015), pp. 595–633.

[3] Ross Boczar, Nikolai Matni, and Benjamin Recht. "Finite-Data Performance Guarantees for the Output-Feedback Control of an Unknown System". In: (2018).

[4] Timothy C. Y. Chan, Rafid Mahmood, and Ian Yihang Zhu. "Inverse Optimization: Theory and Applications". In: *https://arxiv.org/abs/2109.03920* (2022).

[5] Ruidi Chen et al. "Learning from Past Bids to Participate Strategically in Day-Ahead Electricity Markets". In: *IEEE Transactions on Smart Grid* 10.5 (Sept. 2019).

[6] Stefan Clarke et al. *Learning Rationality in Potential Games*. 2023.

[7] Jeremy Coulson, John Lygeros, and Florian Dörfler. "Data-Enabled Predictive Control: In the Shallows of the DeePC". In: 2019 18th European Control Conference (ECC). 2019, pp. 307–312.

[8] Jeremy Coulson, John Lygeros, and Florian Dörfler. "Distributionally Robust Chance Constrained Data-enabled Predictive Control". In: (2021).

[9] Jeremy Coulson, John Lygeros, and Florian Dörfler. "Regularized and Distributionally Robust Data-Enabled Predictive Control". In: 2019 IEEE Conference on Decision and Control (CDC). 2019.

[10] Constantinos Daskalakis, Maxwell Fishelson, and Noah Golowich. "Near-Optimal No-Regret Learning in General Games". In: (Jan. 2023).

[11] Chaosheng Dong, Yiran Chen, and Bo Zeng. "Generalized Inverse Optimization through Online Learning". In: 32nd Conference on Neural Information Processing Systems (NeurIPS 2018). Montréal, Canada, 2018.

[12] Peyman Mohajerin Esfahani et al. *Data-driven Inverse Optimization with Imperfect Information*. 2017.

[13] Sophie Hall et al. "Receding Horizon Games with Coupling Constraints for Demand-Side Management". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022, pp. 3795–3800.

[14] Jason Hartline, Vasilis Syrgkanis, and Éva Tardos. "No-Regret Learning in Bayesian Games". In: *Advances in Neural Information Processing Systems* 28 (2015).

[15] Joao P. Hespanha. *Noncooperative game theory: An introduction for engineers and computer scientists*. Princeton University Press, 2017.

[16] IEA World Energy Balances. *Electricity consumption by sector, Italy 1990-2020*. https://www.iea.org/data-and-statistics/data-product/world-energy-statistics-and-balances. 2022.

[17] IEA World Energy Balances. *World Energy Balances 2022 Database documentation*. https://iea.blob.core.windows.net/assets/25266100-859c-4b9c-bd46-cc4069bd4412/WORLDBAL_Documentation.pdf. 2022.

[18] Italian National Institute of Statistics. *Resident population of 1st January*. http://dati.istat.it. 2022.

[19] Arezou Keshavarz, Yang Wang, and Stephen Boyd. "Imputing a Convex Objective Function". In: *2011 IEEE International Symposium on Intelligent Control (ISIC)* (Sept. 2011).

[20] Bahare Kiumarsi et al. "Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics". In: *Automatica* 50 (2014), pp. 1167–1175.

[21] Robert Kleinberg, Georgios Piliouras, and Éva Tardos. "Multiplicative Updates Outperform Generic No-Regret Learning in Congestion Games". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing* (2009), pp. 533–542.

[22] Walid Krichene, Benjamin Drighès, and Alexander M. Bayen. "Online Learning of Nash Equilibria in Congestion Games". In: *SIAM Journal on Control and Optimization* 53 (2015), pp. 1056–1081.

[23] Volodymyr Kuleshov and Okke Schrijvers. "Inverse Game Theory: Learning Utilities in Succinct Games". In: International Conference on Web and Internet Economics. Ed. by Springer. 2015, pp. 413–427.

[24] Andrew Y. Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the 17th International Conference on Machine Learning* (2000), pp. 663–670.

[25] Thomas D. Nielsen and Finn V. Jensen. "Learning a decision maker's utility function from (possibly inconsistent behavior)". In: *Artificial Intelligence* 160 (2004), pp. 53–78.

[26] Yi Ouyang, Mukul Gagrani, and Rahul Jain. "Learning-based Control of Unknown Linear Systems with Thompson Sampling". In: (2017).

[27] Simon Risanger, Stein-Erik Fleten, and Steven A. Gabriel. "Inverse Equilibrium Analysis of Oligopolistic Electricity Markets". In: *IEEE Transactions on Power Systems* 35.6 (Nov. 2020).

[28] R. Tyrrell Rockafellar and Asen L. Dontchev. *Implicit Functions and Solution Mappings*. Springer Monographs in Mathematics, 2009.

[29] Pier Giuseppe Sessa et al. "Learning to Play Sequential Games versus Unknown Opponents". In: 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Vancouver, Canada, 2020.

[30] Pier Giuseppe Sessa et al. "No-Regret Learning in Unknown Games with Correlated Payoffs". In: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Vancouver, Canada, 2019.

[31] M. Vidyasagar and Rajeeva L. Karandikar. "A learning theory approach to system identification and stochastic adaptive control". In: *Probabilistic and randomized methods for design under uncertainty* (2006), pp. 265–302.

[32] Benjamin Weißing. "The polyhedral projection problem". In: *Mathematical Methods of Operation Research* 91 (2020), pp. 55–72.

[33] Jibang Wu et al. “Inverse Game Theory for Stackelberg Games: the Blessing of Bounded Rationality”. In: 36th Conference on Neural Information Processing Systems (NeurIPS 2022). New Orleans, United States, 2022.

[34] Peng Yi and Lacra Pavel. “An operator splitting approach for distributed generalized Nash equilibria computation”. In: *Automatica* 102 (2019), pp. 111–121.

Appendix A

Single-level reformulations

A.1 Inverse quadratic optimization problem with the suboptimality loss using Lagrange duality

Consider an optimization problem with strictly convex quadratic objective function and linear constraints

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^\top Qx + q^\top x \\ \text{s.t.} \quad & Ax \leq b. \end{aligned} \tag{A.1}$$

When trying to learn $\theta = (Q, q)$ using the suboptimality loss, we get to the following update rule

$$\hat{\theta}_{k+1} = \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k \max_{y\in\mathcal{X}} \phi(x,\theta) - \phi(y,\theta), \tag{A.2}$$

which can be formulated in single-level form as

$$\begin{aligned} \hat{\theta}_{k+1} &= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k \max_{y\in\mathcal{X}} \phi(x,\theta) - \phi(y,\theta) \\ &= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k \max_{y\in\mathcal{X}} \frac{1}{2}x^\top Qx + q^\top x - \frac{1}{2}y^\top Qy - q^\top y \\ &= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k (\frac{1}{2}x^\top Qx + q^\top x) - \eta_k \min_{y\in\mathcal{X}} (\frac{1}{2}y^\top Qy + q^\top y) \\ &= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k (\frac{1}{2}x^\top Qx + q^\top x) - \eta_k \max_{\lambda\geq 0} \mathcal{L}(\lambda) \\ &= \arg\min_{\theta\in\Theta,\lambda\geq 0} \frac{1}{2}\mathcal{R}(\theta) + \eta_k (\frac{1}{2}x^\top Qx + q^\top x - \mathcal{L}(\lambda)), \end{aligned} \tag{A.3}$$

where the program $\max_{\lambda\geq 0} \mathcal{L}(\lambda)$ is the dual of $\min_{y\in\mathcal{X}} \frac{1}{2}y^\top Qy + q^\top y$ obtained using *Lagrange duality*. In particular, let's write the Lagrangian of the primal problem as

$$L(y,\lambda) = \frac{1}{2}y^\top Qy + q^\top y + \lambda^\top (Ay - b). \tag{A.4}$$

The Lagrangian dual function is defined as

$$\mathcal{L}(\lambda) := \min_{y} L(y,\lambda), \tag{A.5}$$

and by exploiting positive-definiteness of Q can be written as

$$\begin{aligned} \mathcal{L}(\lambda) =& \frac{1}{2}(\theta^\top - A^\top\lambda)^\top Q^{-1}(\theta^\top - A^\top\lambda) + \theta^\top Q^{-1}(\theta^\top - A^\top\lambda) + \\ &+ \lambda^\top AQ^{-1}(\theta^\top - A^\top\lambda) - \lambda^\top b. \end{aligned} \tag{A.6}$$

Clearly, since $\mathcal{L}(\lambda)$ is a pointwise minimum of an affine function ($L(y, \lambda)$ is linear in λ), it is a concave function. Moreover, since the constraints $\lambda \geq 0$ are linear, the dual program is a convex optimization problem.
Note that we considered linear constraints just to show a possible realization of the Lagrangian dual function, however, this method is applicable to general convex constraints $g(x) \leq 0$. Moreover, when the constraints are convex but not linear, the process to obtain the single-level form when using the first-order loss is identical, hence we will not discuss it. Instead, we show how to do it when the constraints are linear.

A.2 Inverse quadratic optimization problem with the first-order loss using LP duality

Consider an optimization problem with strictly convex quadratic objective function and linear constraints

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top Q x + q^\top x \\ \text{s.t.} \quad & Ax \leq b. \end{aligned} \tag{A.7}$$

When trying to learn $\theta = (Q, q)$ using the first-order loss, we get to the following update rule

$$\hat{\theta}_{k+1} = \arg\min_{\theta \in \Theta} \frac{1}{2} \mathcal{R}(\theta) + \eta_k \max_{y \in \mathcal{X}} \nabla_x \phi(x, \theta)^\top (x - y), \tag{A.8}$$

which can be formulated in single-level form as

$$\begin{aligned} \hat{\theta}_{k+1} &= \arg\min_{\theta \in \Theta} \frac{1}{2} \mathcal{R}(\theta) + \eta_k \max_{y \in \mathcal{X}} \nabla_x \phi(x, \theta)^\top (x - y) \\ &= \arg\min_{\theta \in \Theta} \frac{1}{2} \mathcal{R}(\theta) + \eta_k (x^\top Q + q^\top) x - \eta_k \min_{y \in \mathcal{X}} (x^\top Q + q^\top) y \\ &= \arg\min_{\theta \in \Theta} \frac{1}{2} \mathcal{R}(\theta) + \eta_k (x^\top Q + q^\top) x - \eta_k \max_{\lambda \in \Lambda} b^\top \lambda \\ &= \arg\min_{\theta \in \Theta, \lambda \in \Lambda} \frac{1}{2} \mathcal{R}(\theta) + \eta_k ((x^\top Q + q^\top) x - b^\top \lambda) \end{aligned} \tag{A.9}$$

where $\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, $\Lambda = \{\lambda \in \mathbb{R}^m_{\geq 0} \mid A^\top \lambda \geq x^\top Q + q^\top\}$ and, since $\min_{y \in \mathcal{X}} (x^\top Q + q^\top) y$ is a linear program, we can employ *LP duality* to obtain its dual $\max_{\lambda \in \Lambda} b^\top \lambda$.

A.3 Inverse linear optimization problem

Consider an optimization problem with linear objective function and linear constraints

$$\begin{aligned} \min_{x} \quad & q^\top x \\ \text{s.t.} \quad & Ax \leq b. \end{aligned} \tag{A.10}$$

First of all, we notice that when applied to linear functions $\phi(x, \theta) = q^\top x$ the suboptimality loss in Equation 3.6 reduces to

$$\begin{aligned} l^s_\theta(x) &= \phi(x, \theta) - \min_{y \in \mathcal{X}} \phi(y, \theta) \\ &= q^\top x - \min_{y \in \mathcal{X}} q^\top y \\ &= \max_{y \in \mathcal{X}} q^\top (x - y) \\ &= \max_{y \in \mathcal{X}} \nabla_x \phi(x, \theta)^\top (x - y), \end{aligned} \tag{A.11}$$

which is the first-order loss in Equation 3.7.
Supposing we want to learn $\theta := (q)$ using one of the two loss functions, we get to the following update rule

$$\hat{\theta}_{k+1} = \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k \max_{y\in\mathcal{X}} q^\top (x - y), \tag{A.12}$$

which can be formulated in single-level form as

$$\begin{aligned}
\hat{\theta}_{k+1} &= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k \max_{y\in\mathcal{X}} q^\top (x - y) \\
&= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k q^\top x - \eta_k \min_{y\in\mathcal{X}} q^\top y \\
&= \arg\min_{\theta\in\Theta} \frac{1}{2}\mathcal{R}(\theta) + \eta_k q^\top x - \eta_k \max_{\lambda\in\Lambda} b^\top \lambda \\
&= \arg\min_{\theta\in\Theta,\lambda\in\Lambda} \frac{1}{2}\mathcal{R}(\theta) + \eta_k (q^\top x - b^\top \lambda),
\end{aligned} \tag{A.13}$$

where $\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, $\Lambda = \{\lambda \in \mathbb{R}^m_{\geq 0} \mid A^\top \lambda \geq q\}$ and, since $\min_{y\in\mathcal{X}} q^\top y$ is a linear program, we can employ *LP duality* to obtain its dual $\max_{\lambda\in\Lambda} b^\top \lambda$.

Appendix B

Polytopic constraints plots

At some point in the thesis, during the analysis of Algorithm 1 with the predictability loss applied to our case study in Chapter 5, we confronted with the necessity of visually understanding the positions of the estimates selected by the algorithm at each iteration with respect to the polytopic constraints of the optimization problem 4.4. Here we outline the method we used, which is based on Theorem 2.3 of [32].
Let's consider the following quadratic optimization problem

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top Q x + \theta^\top x \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{B.1}$$

and its KKT conditions

$$\begin{aligned} &1. \quad Qx + \theta + A^\top \lambda = 0 \\ &2. \quad (Ax - b)^\top \lambda = 0 \\ &3. \quad Ax \leq b \\ &4. \quad \lambda \geq 0. \end{aligned} \tag{B.2}$$

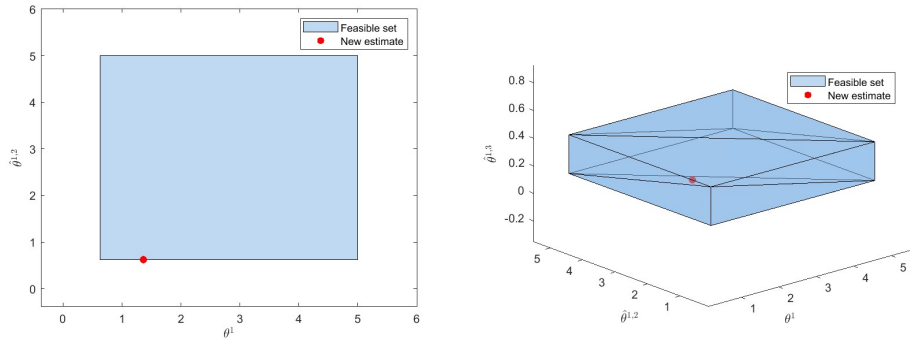
As explained above, whenever we use the predictability loss we need a perfect feedback in order to be able to solve the resulting MISOCP. Let's call the feedback x_f and let's substitute it in the KKT conditions. We get

$$\begin{aligned} &1. \quad Qx_f + \theta + A^\top \lambda = 0 \\ &2. \quad (Ax_f - b)^\top \lambda = 0 \\ &3. \quad Ax_f \leq b \\ &4. \quad \lambda \geq 0. \end{aligned} \tag{B.3}$$

We notice that condition 3 is now useless, so we discard it. From condition 2, we are now able to identify all the dual variables equal to zero, as they correspond to the loose primal constraints. Let's define $\lambda_Z := \{\lambda_i \in \lambda \mid \lambda_i = 0\}$ and $\lambda_{NZ} = \lambda \setminus \lambda_Z$, and partition the dual variables as $\lambda = \begin{bmatrix} \lambda_Z \\ \lambda_{NZ} \end{bmatrix} \in \begin{bmatrix} \mathbb{R}^z \\ \mathbb{R}^{nz} \end{bmatrix}$, where $z + nz = m$. At this point, condition 2 can be ignored as it doesn't bring us any information about λ_{NZ} and θ. We are left with:

$$\begin{aligned} &1. \quad Qx_f + \theta + A^\top \lambda = 0 \\ &4. \quad \lambda_{NZ} \geq 0. \end{aligned} \tag{B.4}$$

Consider condition 1, which is essentially an instance of Definition 2.1 of [32]. First of all, notice



(a) $N = 2$ (b) $N = 3$

Figure B.1: Polytopic constraints and new parameters estimate

that $Qx_f = c \in \mathbb{R}^{NT}$ is a constant vector. Moreover, $A^\top \lambda$ can be written as

$$\begin{aligned} A^\top \lambda &= \begin{bmatrix} A_{Z,Z} & A_{NZ,Z} \\ A_{Z,NZ} & A_{NZ,NZ} \end{bmatrix}^\top \begin{bmatrix} \lambda_Z \\ \lambda_{NZ} \end{bmatrix} \\ &= \begin{bmatrix} A_{Z,Z} & A_{Z,NZ} \\ A_{NZ,Z} & A_{NZ,NZ} \end{bmatrix} \begin{bmatrix} 0 \\ \lambda_{NZ} \end{bmatrix} \\ &= \begin{bmatrix} A_{Z,NZ} \\ A_{NZ,NZ} \end{bmatrix} \lambda_{NZ}, \end{aligned} \tag{B.5}$$

so by defining $A_{NZ} = \begin{bmatrix} A_{Z,NZ} \\ A_{NZ,NZ} \end{bmatrix}$ we obtain

$$1. \qquad c + \theta + A_{NZ}\lambda_{NZ}, \tag{B.6}$$

which is a system of NT equations in the unknowns $\theta \in \mathbb{R}^N$ and $\lambda_{NZ} \in \mathbb{R}^{nz}$. Since our objective is to plot the set of constraints on θ, we now want to reduce this system so that the only unknown is the vector θ. The insight is that each line of the system enforces a constraint on an entry of the unknown vector. For each line $i \in \{1, \dots, NT\}$ we have

$$\begin{cases} c_i + \theta_i + A_{NZ}(i,:)\lambda_{NZ} = 0 \\ \lambda_{NZ} \geq 0, \end{cases} \tag{B.7}$$

where $A_{NZ}(i,:) = [a_{ij}]_{j=1}^{nz} \in \mathbb{R}^{1 \times nz}$. We can distinguish 4 cases:

a) $a_{ij} \geq 0 \quad \forall j, \quad \exists j \mid a_{ij} \neq 0 \quad \rightarrow \quad c_i + \theta_i \leq 0;$

b) $a_{ij} \leq 0 \quad \forall j, \quad \exists j \mid a_{ij} \neq 0 \quad \rightarrow \quad c_i + \theta_i \geq 0;$

c) $a_{ij} = 0 \quad \forall j \quad \rightarrow \quad c_i + \theta_i = 0$

d) $\exists j \mid a_{ij} > 0, \quad \exists z \mid a_{iz} < 0 \quad \rightarrow$ no constraint on θ_i

In this way, we obtain a system of linear equations on $\theta \in \mathbb{R}^N$, representing the sides of the polyhedral constraint set. In Figure B.1 we plotted the results for unknown vectors in $\mathbb{R}^2$ and $\mathbb{R}^3$.