# Inverse Optimization via Learning Feasible Regions

Ke Ren [1] Peyman Mohajerin Esfahani [2] Angelos Georghiou [3]

## Abstract

We study inverse optimization (IO) where the goal is to use a parametric optimization program as the hypothesis class to infer relationships between input-decision pairs. Most of the literature focuses on learning only the objective function, as learning the constraint function (i.e., feasible regions) leads to nonconvex training programs. Motivated by this, we focus on learning feasible regions for known linear objectives and introduce two training losses along with a hypothesis class to parameterize the constraint function. Our hypothesis class surpasses the previous objective-only method by naturally capturing discontinuous behaviors in input-decision pairs. We introduce a customized block coordinate descent algorithm with a smoothing technique to solve the training problems, while for further restricted hypothesis classes, we reformulate the training optimization as a tractable convex program or mixed integer linear program. Synthetic experiments and two power system applications, including comparisons with state-of-the-art approaches, showcase and validate the proposed approach.

## 1. Introduction

Inverse optimization (IO) reverses traditional optimization. While classical optimization finds optimal decisions based on predefined objectives and constraints, IO takes decisions as inputs and identifies the objective and/or constraints that make these decisions either approximately or precisely optimal. Interest in IO has surged in recent years, leading to advancements in learning theory (Aswani et al., 2018; Mohajerin Esfahani et al., 2018; Chan & Kaw, 2020) and recently in reinforcement learning (Zattoni Scroccaro et al., 2025), as well as applications such as transportation (Zhang & Paschalidis, 2017; Chen et al., 2021), system control (Akhtar et al., 2021), robotics (Dimanidis et al., 2025), healthcare (Ayer, 2015; Ajayi et al., 2022), and finance (Li, 2021). More recently, the IO framework has also been successfully applied to the Amazon Last Mile Routing Research Challenge, where the goal is to learn and replicate human drivers' routing preferences (Zattoni Scroccaro et al., 2025).

[1]Amazon [2]University of Toronto and Delft University of Technology [3]University of Cyprus. Correspondence to: Ke Ren <renkea@amazon.com >.

*Proceedings of the 42nd International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

The IO model builds on a parametric *forward* optimization model that represents the decision-generating process. Given an input signal $\boldsymbol{s} \in \mathbb{R}^k$, the following optimization problem defined through $f, g : \mathbb{R}^k \times \mathbb{R}^n \to \mathbb{R}$, generates an optimal solution $\boldsymbol{x} \in \mathbb{R}^n$ which is observed, however, it is possible that the observation is contaminated by noise.

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x},\boldsymbol{s}) \qquad \text{s.t.} \quad g(\boldsymbol{x},\boldsymbol{s}) \le 0 \tag{1}$$

In the most general case, decision makers have no knowledge of $f$ and $g$ but have access to $N$ independent pairs of input/decisions $\{\boldsymbol{s}_i, \boldsymbol{x}_i\}_{i=1}^N$ from problem (1). As the space of all possible objective and constraint functions is vast, the decision-maker seeks to approximate $f$ and $g$ by candidates from some parametric hypothesis spaces $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\theta}}$, $\boldsymbol{\theta} \in \Theta$, where $\Theta$ represents a finite-dimensional parameter set. Correspondingly, in IO, decision-makers aim to learn $\boldsymbol{\theta}$ such that for each signal $\boldsymbol{s}_n$, $\boldsymbol{x}_n$ is optimal (or approximately optimal) in the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{s}) \qquad \text{s.t.} \quad g_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{s}) \le 0. \tag{2}$$

Through a supervised learning lens, the parameterized model (2) can be viewed as a hypothesis class for learning the mapping between the input $\boldsymbol{s}$ and the output decision $\boldsymbol{x}$ generated by the forward optimization (1).

### 1.1. Contributions

This paper studies the learning problem of IO where, unlike the majority of the literature that focuses on a parametric form for the objective function $f_{\boldsymbol{\theta}}$, the main objective is to learn the constraints function $g_{\boldsymbol{\theta}}$. In this context, the paper has three main contributions:

**Loss functions compatible with IO data:** Generalizing existing literature, we introduce two loss functions $\ell_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{s})$ to evaluate model fit (2) with parameter $\boldsymbol{\theta}$ on the input-output data pair $(\boldsymbol{x},\boldsymbol{s})$ from (1) (Proposition 2.1). Regarding performance on unseen (test) data, we argue that the IO setting encompasses a "true" counterpart of these losses, unlike general supervised learning problems; see Figure 2 (bottom) for their geometric representation.

**Hypothesis Classes with Convex and MILP Reformulations:** We propose a new hypothesis class to parameterize the constraint function $g_{\boldsymbol{\theta}}$, which is significantly richer than existing IO approaches (Example 3.2). However, it results in non-convex training problems. We demonstrate that special cases of this hypothesis class allow the optimization of the training loss $\ell_{\boldsymbol{\theta}}$ to be exactly reformulated into tractable convex optimization (Theorem 3.3) and mixed-integer linear programs (MILP) (Proposition D.1), which can be solved efficiently with off-the-shelf solvers.

**Smoothed block-coordinate descent algorithm:** We further exploit the structure of the proposed losses and devise a tailored block coordinate descent algorithm together with a smoothing technique to train our IO setting in for the generic hypothesis class (Algorithm 2). The smoothing technique provides an interesting insight into the relation between the two proposed loss functions, justifying why one often presents better computation results when optimized using vanilla gradient descent (Algorithm 1); see Remark 3.4.

The paper is organized as follows: Section 2 introduces two losses and explores their properties. Section 3 presents the hypothesis class together with the proposed coordinate descent algorithm with a smoothing technique to solve the general problem, along with convex and MILP reformulations for specific cases. Section 4 covers computational studies. Limitations and future work are discussed in Section 5. Proofs and additional computational studies are in the appendix.

### 1.2. Related Works

Extensive studies have been carried out to estimate the objective functions in data-driven IO. The various approaches primarily diverge in terms of the loss functions utilized to capture the disparity between predictions and observations. These include the KKT loss (Keshavarz et al., 2011), first-order loss (Bertsimas et al., 2015), predictability loss (Aswani et al., 2018), and suboptimality loss (Mohajerin Esfahani et al., 2018). The properties and relationships of these approaches are summarized (Mohajerin Esfahani et al., 2018). Other methodologies are also proposed under different settings including online settings (Bärmann et al., 2017; Dong et al., 2018). In the landscape of IO for feasible regions, the majority of existing works have traditionally concentrated on right-hand side parameters (Dempe & Lohse, 2006; Güler & Hamacher, 2010; Xu et al., 2016; Saez-Gallego et al., 2016; Lu et al., 2018). However, recent endeavors by (Aswani et al., 2018; Tan et al., 2020; Ghobadi & Mahmoudzadeh, 2020) have expanded the scope by considering various aspects of constraint parameters.

The work (Aswani et al., 2018) incorporates general constraint parameters into their model, departing from the mainstream. However, their model encounters intractability issues, relying on enumerations to evaluate the problem. Similarly, (Tan et al., 2020) faces tractability challenges, employing sequential quadratic programming within a bi-level optimization framework for solving constraint parameters in linear programming. While (Chan & Kaw, 2020) and (Ghobadi & Mahmoudzadeh, 2020) propose tractable solutions, their settings impose some restrictions: (Chan & Kaw, 2020) does not consider a data-driven setting and confines their analysis to a single observation in linear programming. The paper (Ghobadi & Mahmoudzadeh, 2020) extends this limitation to multi-point scenarios, but only for the linear programming case where the multiple points are merely feasible solutions, not necessarily optimal.

## 2. Learning Frameworks

**Inverse optimization as supervised learning.** From a machine learning perspective, the IO problem can be categorized as a supervised learning problem, where $\boldsymbol{s}$ represents the independent variable and $\boldsymbol{x}$ represents the response. Therefore, it is natural to define a loss minimization procedure (as shown in (3)) to find the unknown parameters $\boldsymbol{\theta}$.

$$\min_{\boldsymbol{\theta}\in\Theta} \quad \frac{1}{N}\sum_{i=1}^{N}\ell_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{s}_i). \tag{3}$$

An appropriate loss is required to measure both the feasibility and optimality of any given solution $\boldsymbol{x}$.

### 2.1. Loss Function

In this section, we introduce two loss functions that relax feasibility and optimality in problem (2) while penalizing deviations. Two potential reasons why problem (2) might fail to replicate the feasibility and optimality of problem (1) are: $(i)$ the hypothesis class used for the constraints $g_{\boldsymbol{\theta}}$ and/or the objective $f_{\boldsymbol{\theta}}$ may lack the complexity to capture the behavior of (1), and $(ii)$ the training data could be noisy, meaning the signal $\boldsymbol{s}$ and/or the decisions $\boldsymbol{x}$ may be influenced by unaccounted measurement noise. Typically, the degree of infeasibility of a data point can be quantified by $g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$, with positive values indicating infeasibility, while suboptimality can be assessed via

$$J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) := f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) - \left[\begin{array}{ll} \min\limits_{\boldsymbol{y}\in\mathbb{R}^n} & f_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{s}) \\ \text{s.t.} & g_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{s}) \le 0 \end{array}\right]. \tag{4}$$

For a fixed $\boldsymbol{\theta}$, $J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$ quantifies the difference between the objective value achieved by data point $(\boldsymbol{x}, \boldsymbol{s})$ under the hypothesized objective $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$ and the optimal value of (2) with signal $\boldsymbol{s}$. In noise-free scenarios, negative $J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$

suggest over-constraint in (2), while positive values imply overly relaxed feasibility, rendering $(\boldsymbol{x}, \boldsymbol{s})$ suboptimal.

The loss functions are termed *predictability* and *suboptimality loss*, respectively.

$$\ell_{\boldsymbol{\theta}}^{\mathrm{p}}(\boldsymbol{x}, \boldsymbol{s}) := \begin{bmatrix} \min & \|\boldsymbol{\gamma}\| \\ \text{s.t.} & \boldsymbol{\gamma} \in \mathbb{R}^n \\ & g_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\gamma}, \boldsymbol{s}) \leq 0 \\ & J_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\gamma}, \boldsymbol{s}) \leq 0 \end{bmatrix}, \tag{5a}$$

$$\ell_{\boldsymbol{\theta}}^{\mathrm{sub}}(\boldsymbol{x}, \boldsymbol{s}) := \begin{bmatrix} \min & \|(\gamma_f, \gamma_o)\| \\ \text{s.t.} & \gamma_o, \gamma_f \in \mathbb{R}_+ \\ & g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq \gamma_f \\ & J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq \gamma_o \end{bmatrix}. \tag{5b}$$

Figure 1 illustrates the two loss functions. The $\boldsymbol{\gamma}$ variable (red) in the predictability loss $\ell_{\boldsymbol{\theta}}^{\mathrm{p}}(\boldsymbol{x}, \boldsymbol{s})$ allows for repositioning the observed $\boldsymbol{x}$ to achieve feasibility and optimality while penalizing the extent of adjustment. Note that $\boldsymbol{\gamma}$ doesn't merely project $\boldsymbol{x}$ into the feasible region $g_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s})$ but balances between infeasibility and suboptimality. In the suboptimality loss $\ell_{\boldsymbol{\theta}}^{\mathrm{sub}}(\boldsymbol{x}, \boldsymbol{s})$, $\gamma_f$ and $\gamma_o$ (light blue) act as slack variables, regulating the levels of infeasibility and suboptimality independently.

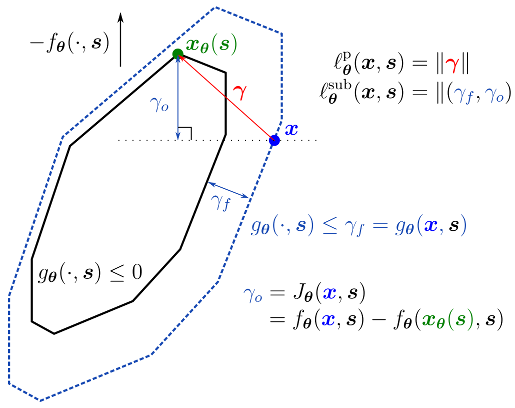
$-f_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s})$
$\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s})$
$\ell_{\boldsymbol{\theta}}^{\mathrm{p}}(\boldsymbol{x}, \boldsymbol{s}) = \|\boldsymbol{\gamma}\|$
$\ell_{\boldsymbol{\theta}}^{\mathrm{sub}}(\boldsymbol{x}, \boldsymbol{s}) = \|(\gamma_f, \gamma_o)\|$
$\gamma_o$
$\boldsymbol{\gamma}$
$\boldsymbol{x}$
$\gamma_f$
$g_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s}) \leq \gamma_f = g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$
$g_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s}) \leq 0$
$\gamma_o = J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$
$= f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) - f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s})$

*Figure 1.* Pictorial representation of the predictability and suboptimality loss functions.

The predictability loss was first proposed in (Aswani et al., 2018) using a slightly different formulation. The naming convention can be explained by defining variable $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ shifts $\boldsymbol{x}$, leading to the objective function $\min_{\boldsymbol{y} \in \mathcal{X}(\boldsymbol{s})}, \|\boldsymbol{y} - \boldsymbol{x}\|$. Here, $\mathcal{X}(\boldsymbol{s})$ denotes the set of optimal solutions of problem (2). Consequently, the loss penalizes the discrepancy between the observed decision $\boldsymbol{x}$ and the potential predicted decision $\boldsymbol{y}$. Unlike the loss proposed in (Mohajerin Esfahani et al., 2018) which assumed known constraints, the suboptimality loss extends the loss to unknown constraints penalizing both infeasibility and suboptimality.

The following proposition shows that the proposed loss functions (5) are well defined, in the sense that for any $\boldsymbol{\theta} \in \Theta$ and $(\boldsymbol{x}, \boldsymbol{s})$ in the training dataset, the loss $\ell_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$ is zero if and only if $\boldsymbol{x}$ is also an optimal solution of (2). In other words, given a rich enough hypothesis class for $g_{\boldsymbol{\theta}}$ and $f_{\boldsymbol{\theta}}$, the set of optimal solutions of (2) coincides with the set of observed optimal solutions of (1). Moreover, the statement implies that if the optimizer $\boldsymbol{\theta}^*$ of (3) does not achieve a zero loss, i.e., there exists $(\boldsymbol{x}, \boldsymbol{s})$ in the training data such that $\ell_{\boldsymbol{\theta}^*}(\boldsymbol{x}, \boldsymbol{s}) > 0$, then there is no other $\boldsymbol{\theta}$ in the hypothesis class that perfectly describes the measured pair $(\boldsymbol{x}, \boldsymbol{s})$. We term this property as *full characterization*.

**Proposition 2.1** (Full characterization)**.** *For any $\boldsymbol{\theta} \in \Theta$ and the IO data pair $(\boldsymbol{x}, \boldsymbol{s})$ generated by the forward model (1), both predictability $\ell_{\boldsymbol{\theta}}^{p}(\boldsymbol{x}, \boldsymbol{s})$ and suboptimality $\ell_{\boldsymbol{\theta}}^{sub}(\boldsymbol{x}, \boldsymbol{s})$ loss defined in (5) satisfy*

$$\ell_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = 0 \iff \boldsymbol{x} \in \begin{bmatrix} \underset{\boldsymbol{y} \in \mathbb{R}^n}{\arg\min} & f_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{s}) \\ \text{s.t.} & g_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{s}) \leq 0 \end{bmatrix}.$$

## 2.2. Measuring Performance

In this subsection, we define metrics for the goodness of fit in an out-of-sample evaluation. In general, there are two ways to measure the performance. The first method is to directly measure the feasibility and optimality of the optimal solutions generated by learned optimization problems. To this end, for fixed $\boldsymbol{\theta}' \in \Theta$, define $\boldsymbol{x}_{\boldsymbol{\theta}'}^*(\boldsymbol{s})$ to be an optimizer from problem (2). Thus we evaluate the *true loss* $\sum_{i=1}^{N_{\text{out}}} \ell(\boldsymbol{x}_{\boldsymbol{\theta}'}^*(\boldsymbol{s}_i), \boldsymbol{s}_i)$ using the true predictability loss

$$\ell^{\mathrm{p}}(\boldsymbol{x}, \boldsymbol{s}) := \begin{bmatrix} \min & \|\boldsymbol{\gamma}\| \\ \text{s.t.} & \boldsymbol{\gamma} \in \mathbb{R}^n \\ & g(\boldsymbol{x} + \boldsymbol{\gamma}, \boldsymbol{s}) \leq 0 \\ & J(\boldsymbol{x} + \boldsymbol{\gamma}, \boldsymbol{s}) \leq 0 \end{bmatrix} \tag{6a}$$

and true suboptimality loss

$$\ell^{\mathrm{sub}}(\boldsymbol{x}, \boldsymbol{s}) := \begin{bmatrix} \min & \|(\gamma_f, \gamma_o)\| \\ \text{s.t.} & \gamma_o \in \mathbb{R}_+, \gamma_f \in \mathbb{R}_+ \\ & g(\boldsymbol{x}, \boldsymbol{s}) \leq \gamma_f \\ & J(\boldsymbol{x}, \boldsymbol{s}) \leq \gamma_o \end{bmatrix}. \tag{6b}$$

In most practical scenarios, we do not have access to the forward problem (1), preventing a direct evaluation of the true performance of $\boldsymbol{\theta}$. Nevertheless, we often have additional data $\{(\boldsymbol{x}_i, \boldsymbol{s}_i)\}_{i=1}^{N_{\text{out}}}$ that was not utilized during the training phase. As an alternative, we can straightforwardly compute $\sum_{i=1}^{N_{\text{out}}} \ell_{\boldsymbol{\theta}'}(\boldsymbol{x}_i, \boldsymbol{s}_i)$ for a selected $\boldsymbol{\theta}' \in \Theta$ for both the predictability and suboptimality loss functions. We illustrate the relationships between these four metrics below in Figures 2. From Figure 2 (bottom), it can be seen that if the recovered optimal solution $\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s})$ coincide with the observed optimal $\boldsymbol{x}$, all losses become zero. Additionally, the true predictability loss $\ell^{\mathrm{p}}$ and sample-based loss $\ell_{\boldsymbol{\theta}}^{\mathrm{p}}$ are equivalent when optimal solutions are unique for both true and recovered problems. It is also worth noting that the true suboptimality loss coincides with the "Smart Predict, then Optimize" (SPO) loss in (Elmachtoub & Grigas, 2022); see Remark 4.4 in (Zattoni Scroccaro et al., 2024) for more details.

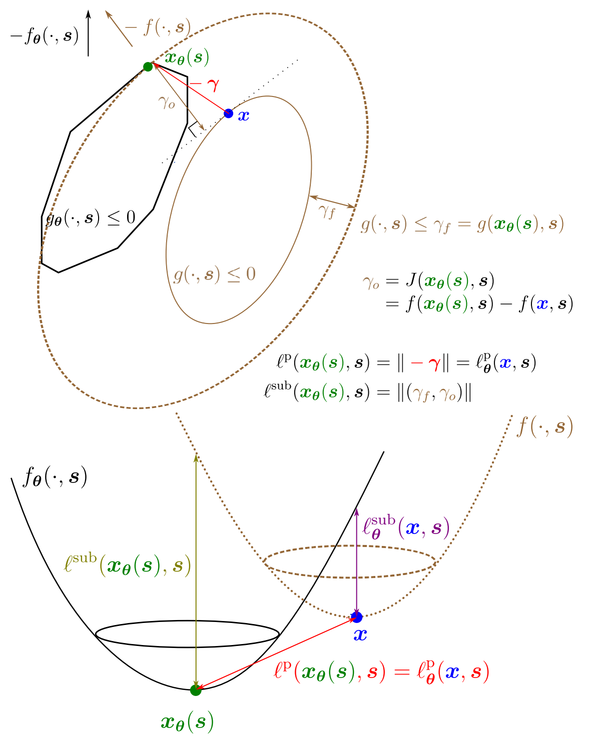
$-f_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s})$
$-f(\cdot, \boldsymbol{s})$
$\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s})$
$-\boldsymbol{\gamma}$
$\gamma_o$
$\boldsymbol{x}$
$\gamma_f$
$g_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s}) \leq 0$
$g(\cdot, \boldsymbol{s}) \leq \gamma_f = g(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s})$
$g(\cdot, \boldsymbol{s}) \leq 0$
$\gamma_o = J(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s}) = f(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s}) - f(\boldsymbol{x}, \boldsymbol{s})$
$\ell^{\mathrm{p}}(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s}) = \| -\boldsymbol{\gamma} \| = \ell^{\mathrm{p}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$
$\ell^{\mathrm{sub}}(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s}) = \|(\gamma_f, \gamma_o)\|$
$f(\cdot, \boldsymbol{s})$
$f_{\boldsymbol{\theta}}(\cdot, \boldsymbol{s})$
$\ell^{\mathrm{sub}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$
$\ell^{\mathrm{sub}}(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s})$
$\boldsymbol{x}$
$\ell^{\mathrm{p}}(\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{s}) = \ell^{\mathrm{p}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$
$\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{s})$

*Figure 2.* **Top:** Pictorial representation of true predictability and suboptimality losses. **Bottom:** relationship between estimated out-of-sample loss ($\ell^{\mathrm{p}}_{\boldsymbol{\theta}}$, $\ell^{\mathrm{sub}}_{\boldsymbol{\theta}}$) and true out-of-sample loss ($\ell^{\mathrm{p}}$, $\ell^{\mathrm{sub}}$).

## 3. Hypothesis Class for $g_\theta$ and Reformulations

In this section, we restrict the admissible constraint function $g_{\boldsymbol{\theta}}$ to a specific hypothesis class and reformulate the predictability and suboptimality losses. For the remainder of the paper, we assume the objective function is known and focus on the unknown constraints.

Let $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})$ and $\boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})$ be a matrix and vector with appropriate dimensions, and restrict the constraint to

$$g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \min_{\boldsymbol{z} \in \mathcal{Z}} \|\boldsymbol{x} - \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})\|. \tag{7a}$$

The function uses a latent variable $\boldsymbol{z}$, which resides in the predetermined conic *primitive set*

$$\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^p \, : \, H\boldsymbol{z} - \boldsymbol{h} \in \mathcal{K}\} \tag{7b}$$

where matrices $H \in \mathbb{R}^{l \times p}$ and $\boldsymbol{h} \in \mathbb{R}^l$ and the proper convex cone $\mathcal{K}$ are given.

Intuitively, the hypothesis class (17) controls the feasible region of $\boldsymbol{x}$ by manipulating the primitive set $\mathcal{Z}$. Indeed, we can see that the constraint $g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq 0$ can be reformulated to $\boldsymbol{x} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})$, $\boldsymbol{z} \in \mathcal{Z}$. In other words, for each pair $(\boldsymbol{x}, \boldsymbol{s})$ there exists a $\boldsymbol{z} \in \mathcal{Z}$ that maps to $\boldsymbol{x}$ through the linear map $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})$. Hence, through the choice of $\boldsymbol{\theta}$, the matrix $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})$ can scale, rotate and project the primitive set $\mathcal{Z}$, while vector $\boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})$ is responsible for translating the set. An alternative way to view the hypothesis is that the learning model induces the policy $\boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z}(\boldsymbol{s}) + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})$ for some $\boldsymbol{z}(\boldsymbol{s}) \in \mathcal{Z}$, hence problem (2) aims to learn the policy that best fits the training data. The choice of the primitive set $\mathcal{Z}$ plays a crucial role in approximation highlighted in the next remark.

**Remark 3.1** (Choice of primitive sets)**.** *When the primitive set $\mathcal{Z}$ is a $p$-dimensional simplex, the resulting policy can be seen as a switched version of $p$ different policies, each represented by a column of $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})$. The latent variable $\boldsymbol{z}_i(\boldsymbol{s})$ acts as the switching mechanism. More broadly, the matrix $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})$ projects the high-dimensional simplex (or any other polytopic primitive set) onto the space of $\boldsymbol{x}$. Increasing the dimensions of the primitive set enhances the flexibility of the hypothesis class. Conversely, if the constraints of the forward problem (1) are believed to be ellipsoidal, an appropriate choice is $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^n \mid \|\boldsymbol{z}\|_2 \leq 1\}$, with $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})$ rotating and scaling the set accordingly.*

The following example illustrates the richness of the IO models with the constraint class (17) by demonstrating $(i)$ how it can learn a forward problem with a discontinuous policy, and $(ii)$ why no IO model with only objective learning (e.g., (Mohajerin Esfahani et al., 2018)) is sufficient to achieve the same.

**Example 3.2** (Constraint vs. objective learning in IO)**.** *Motivated by power systems, consider*

$$\min_{\boldsymbol{x} \in [0,2]^2} \quad s x_1 + (1-s) x_2 \quad \text{s.t.} \quad x_1 + x_2 = 1, \tag{8}$$

*where $x_1$ and $x_2$ represent two generators output aiming to meet the demand $x_1 + x_2 = 1$ at the lowest possible cost. The signal $s \in [0,1]$ indicates the per-unit production cost of the first generator, while the cost of the other is proportional to $1 - s$. The optimal policy of (8) is*

$$(x_1^*(s), x_2^*(s)) = \begin{cases} (1,0) & \textit{if } s < 0.5, \\ ([0,1],[0,1]) & \textit{if } s = 0.5, \\ (0,1) & \textit{if } s > 0.5, \end{cases} \tag{9}$$

*such that $x_1^*(0.5) + x_2^*(0.5) = 1$. The hypothesis class (17) can recover the optimal policy by defining $\mathcal{Z} = \{\boldsymbol{z} \in [0,1]^2 \, : \, \boldsymbol{e}^\top \boldsymbol{z} = 1\}$ as the two-dimensional unit simplex, and selecting $A_{\boldsymbol{\theta}}(\boldsymbol{s}) = I$ (the $2 \times 2$ identity) and $\boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}) = 0$ such that the resulting policy yields $x_1(s) = z_1(s)$ and $x_2(s) = z_2(s)$. Selecting $z_1(s)$ and $z_2(s)$ as in (9) satisfies $\mathcal{Z}$ by construction, and recovers the optimal policy. However, attempting to learn a quadratic function*

*$f_{\boldsymbol{\theta}}(\boldsymbol{x}, s) = \boldsymbol{x}^\top Q_{\boldsymbol{\theta}} \boldsymbol{x} + \boldsymbol{x}^\top A_{\boldsymbol{\theta}} s$ results in the linear policy $\boldsymbol{x}(s) = Q_{\boldsymbol{\theta}}^{-1} \boldsymbol{A}_{\boldsymbol{\theta}} s$ for any $s \in (0,1)$ while saturating at the boundary for any $s = 0$ or $s = 1$, indicating that learning a quadratic cost is indeed insufficient to capture the optimal policy.*

### 3.1. Reformulation for Linear Objective Functions

For the remainder of the paper, and without loss of generality, we make the assumption that we can express $A_{\boldsymbol{\theta}}(\boldsymbol{s})$ and $b_{\boldsymbol{\theta}}(\boldsymbol{s})$ as affine functions of $\boldsymbol{s}$ through $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \boldsymbol{A}_0 + \boldsymbol{A}_1 s_1 + \ldots + \boldsymbol{A}_K s_K$ and $\boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \boldsymbol{b}_0 + \boldsymbol{b}_1 s_1 + \ldots + \boldsymbol{b}_K s_K$ where $\boldsymbol{A}_k \in \mathbb{R}^{n \times p}$ and $\boldsymbol{b}_k \in \mathbb{R}^n$ for $k = 0, \ldots, K$, i.e., $\boldsymbol{\theta} = (\{\boldsymbol{A}_k, \boldsymbol{b}_k\}_{k=0}^K)$. The following theorem provides the reformulation for the learning problem (3).

**Theorem 3.3** (Exact reformulation)**.** *Let $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ given in (17). The learning problems can be reformulated as follows, using the predictability loss*

$$
\begin{aligned}
\min\ & \frac{1}{N} \sum_{i=1}^N \|\boldsymbol{\gamma}_i\| \\
\text{s.t. } & \boldsymbol{A}_k \in \mathbb{R}^{n \times p},\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k \le K, \\
& \left.\begin{array}{l}
\boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{z}_i \in \mathbb{R}^p,\ \boldsymbol{\lambda}_i \in \mathcal{K}^* \\
\boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) \\
H\boldsymbol{z}_i - \boldsymbol{h} \in \mathcal{K} \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{\lambda}_i^\top H = 0 \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top (\boldsymbol{x}_i + \boldsymbol{\gamma}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \le 0
\end{array}\right\} \forall i \le N
\end{aligned}
\tag{10a}
$$

*and using the suboptimality loss*

$$
\begin{aligned}
\min\ & \frac{1}{N} \sum_{i=1}^N \|(\gamma_{f,i}, \gamma_{o,i})\| \\
\text{s.t. } & \boldsymbol{A}_k \in \mathbb{R}^{n \times p},\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k \le K, \\
& \left.\begin{array}{l}
\gamma_{f,i}, \gamma_{o,i} \in \mathbb{R}_+, \boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{z}_i \in \mathbb{R}^p \\
\boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) \\
\|\boldsymbol{\gamma}_i\| \le \gamma_{f,i} \\
H\boldsymbol{z}_i - \boldsymbol{h} \in \mathcal{K}, \boldsymbol{\lambda}_i \in \mathcal{K}^* \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{\lambda}_i^\top H = 0 \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top (\boldsymbol{x}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \le \gamma_{o,i}
\end{array}\right\} \forall i \le N
\end{aligned}
\tag{10b}
$$

Both problems (10) share similar complexity in the sense that the constraints are linear except for the bilinear term $A_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i$. The problem can be efficiently approximated in practice by performing block coordinate descent (Bertsekas, 1999) on matrices $\{\boldsymbol{A}_k\}_{k=0}^K$ and vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ sequentially until convergence. The gradient descent-based algorithm for the predictability loss in (10) is outlined in Algorithm 1. A similar algorithm can be derived for the suboptimality loss. It updates $\{\boldsymbol{A}_k\}_{k=1}^K$ with gradient descent and solves (10) to compute $\{\boldsymbol{z}_i\}_{i=1}^N$ for fixed $\{\boldsymbol{A}_k\}_{k=1}^K$. The gradient of the predictability loss (10) with respect to $\{\boldsymbol{A}_k\}_{k=0}^K$ is as follows: For some $(\boldsymbol{x}, \boldsymbol{s})$, denote $\boldsymbol{z}^*$ as the optimal latent variables, $\boldsymbol{\beta}^*$ as the optimal dual multipliers of constraint $\boldsymbol{x} + \boldsymbol{\gamma} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})$, and $\boldsymbol{\mu}^*$ as the optimal dual multipliers of constraint $\boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \boldsymbol{\lambda}^\top H = 0$. The gradient of the loss function is

$$
\frac{\partial}{\partial \boldsymbol{A}_k} \ell_{\boldsymbol{\theta}}^{\mathrm{p}}(\boldsymbol{x}, \boldsymbol{s}) = s_k(\boldsymbol{c}(\boldsymbol{s})\boldsymbol{\mu}^{*\top} - \boldsymbol{\beta}^* \boldsymbol{z}^{*\top}), \quad \forall k = 0, \ldots, K\,, \tag{11}
$$

where $s_0 = 1$. In practice, the step size $\eta$ of the gradient descent can by dynamically updated at each iteration by performing backtracking such as Armijo's rule or other criteria.

**Algorithm 1** Gradient descent based algorithm for (10) using predictability loss

```
1: Initialization: $\{A_k^1\}_{k=0}^K$, $T, \eta, t = 1$;
2: while $t \le T$ do
3:     Solve (10) and denote by $\{\boldsymbol{z}_i^*, \boldsymbol{\beta}_i^*, \boldsymbol{\mu}_i^*)\}_{i=1}^N$ the optimal solution and dual multipliers.
4:     $\boldsymbol{A}_k^{t+1} = \boldsymbol{A}_k^t - \eta \sum_{i=1}^N s_k(\boldsymbol{c}(\boldsymbol{s}_i)\boldsymbol{\mu}_i^\top - \boldsymbol{\beta}_i^* \boldsymbol{z}_i^{*\top})$.
5:     $t = t + 1$.
6: end while
```

### 3.2. Adaptive Smoothing

One can inspect that the optimum of these programs (10) is typically a nonsmooth function in the variable $\{\boldsymbol{A}_k\}_{k=0}^K$ due to the inner optimization over the multiplicative decision variable $\{\boldsymbol{z}_i\}_{i=1}^N$. This observation motivates us to deploy smoothing techniques (e.g., Nesterov's smoothing (Nesterov et al., 2018)) to improve our algorithm performance. With this in mind, the next remark provides a connection between the two optimization programs (10) (i.e., predictability vs suboptimality loss), which intuitively sheds light on why the suboptimality loss has often better numerical performance from a computational viewpoint, see numerical experiments in Table 1 and more details in Appendix H.2.

**Remark 3.4** (Suboptimality loss as smoothed predictability loss)**.** *Consider the suboptimality problem (10b). The feasibility variable $\gamma_{f,i}$ of (10b) coincides with Nesterov's smoothing counterpart (Nesterov, 2005) of the predictability objective function in (10a) when the distance function is consistent with the underlying norm in these programs with an appropriate smoothness parameter. We refer to Appendix E for the details to formalize this discussion.*

Inspired by Remark 3.4, we define smoothed predictability by relaxing the hard constraint involving $\{\boldsymbol{A}_k\}_{k=0}^K$ and penalizing them in the loss function. This is formally defined in the next definition (suboptimality loss follows the same logic and will be presented in Appendix E). For clarity, we assume $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ as given in (17).

**Definition 3.5** (Smoothed predictability loss)**.** *The $\epsilon$-smoothed counterpart of the predictability loss (10a) is de-*

*fined through the optimization program*

$$\begin{aligned}
\min\ & \frac{1}{N}\sum_{i=1}^{n}\|\gamma_i\| + \epsilon_1\sum_{i=1}^{n}\|\gamma_{s1}\| + \epsilon_2\sum_{i=1}^{n}\|\gamma_{s2}\| \\
\text{s.t. } & \boldsymbol{A}_k \in \mathbb{R}^{n\times p},\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k = 1,\dots,K, \\
& \left.\begin{array}{l}
\boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{z}_i \in \mathbb{R}^p,\ \boldsymbol{\lambda}_i \in \mathcal{K}^* \\
\boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) + \boldsymbol{\gamma}_{s1} \\
H\boldsymbol{z}_i - \boldsymbol{h} \in \mathcal{K} \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top(\boldsymbol{x}_i + \boldsymbol{\gamma}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \le 0 \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{\lambda}_i^\top H + \boldsymbol{\gamma}_{s2} = 0
\end{array}\right\}\ \forall i \le N.
\end{aligned} \tag{12}$$

In order to optimize the smoothed losses (12), we choose to adaptively increase the penalizing coefficients $\epsilon = (\epsilon_1, \epsilon_2)$. This allows the magnitude of the slack variables $\gamma_{s1}$ and $\gamma_{s2}$ to diminish to zero, thus solving the original predictability loss in (10). With this in mind, we present the final algorithm in Algorithm 2.

**Algorithm 2** Adaptive smoothing algorithm

```
1: Initialization: ${\boldsymbol{A}_k^1}_{k=0}^K$, $T$, $\eta$, $t = 1$, $\epsilon_1$ and $\epsilon_2$ for predictability loss;
2: while $t \le T$ do
3:     Solve (12) and denote by $\{\boldsymbol{z}_i^*, \boldsymbol{\beta}_i^*, \boldsymbol{\mu}_i^*, \boldsymbol{\gamma}_{s1}^*, \boldsymbol{\gamma}_{s2}^*\}_{i=1}^N$ the optimal solution, dual multipliers, and smoothing variables.
4:     $\boldsymbol{A}_k^{t+1} = \boldsymbol{A}_k^t - \eta\sum_{i=1}^N s_k(\boldsymbol{c}(\boldsymbol{s}_i)\boldsymbol{\mu}_i^\top - \boldsymbol{\beta}_i^* \boldsymbol{z}_i^{*\top})$.
5:     Re-solve problem (12) to get new values of $\{\boldsymbol{\gamma}_{s1}^*, \boldsymbol{\gamma}_{s2}^*\}_{i=1}^N$
6:     If the change in $\sum_{i=1}^N\|\boldsymbol{\gamma}_{s1^*}\|$ or $\sum_{i=1}^N\|\boldsymbol{\gamma}_{s2^*}\|$ is small enough, increase the value of $\epsilon_1$or $\epsilon_2$.
7:     $t = t + 1$.
8: end while
```

We use simple updating rules for adjusting $(\epsilon_1, \epsilon_2)$ in Algorithm 2: The initial values are $\epsilon_1 = \epsilon_2 = 1$, and every time the change in the values of $\sum_{i=1}^N\|\boldsymbol{\gamma}_{s1}\|^2$ and $\sum_{i=1}^N\|\boldsymbol{\gamma}_{s2}\|^2$ are less than $0.01/10^{(\log_2(\epsilon_1)+1)}$, we multiply the parameters $\epsilon_1$ and $\epsilon_2$ by 2.

The convergence property of Algorithms 1 and 2 is formally summarized in the following proposition.

**Proposition 3.6** (Convergence)**.** *Let $\boldsymbol{\theta}^t = \{\boldsymbol{A}_k^t\}_{k=0}^K$ where $\{\boldsymbol{A}_k^t\}_{k=0}^K$ is the outcome of Algorithm 1 (Algorithm 2, respectively) after $t$ iterations with the Armijo rule stepsize, and $\{\boldsymbol{b}_k^t\}_{k=0}^K$ be the solution of the proposed predictability or suboptimality loss function defined in (10) (the smoothed version (27) in Section E.1 in the supplementary, respectively) when the matrices $\boldsymbol{A}_k$ are set to the proposed algorithm outcome. Then, the loss function value is monotonically decreasing, i.e., for any pair $(\boldsymbol{x}, \boldsymbol{s})$ we have $\ell_{\boldsymbol{\theta}^t}(\boldsymbol{x}, \boldsymbol{s}) \ge \ell_{\boldsymbol{\theta}^{t+1}}(\boldsymbol{x}, \boldsymbol{s}) \ge 0$, and hence, it convergences to a finite nonnegative value (local optimal).*

We illustrate the difference between Algorithms 1 and 2 through a noiseless example where the true hypothesis class is covered by the primitive set. The final results are reported in Table 1; see also Appendix H.2 for further related details. It is worth mentioning that when using the vanilla gradient decent Algorithm 1, the suboptimality loss has much better performance (since it has better smoothness), whereas leveraging the smooth counterpart of Algorithm 2 achieves competitive performance with both loss functions (see also Figure 5 for more details concerning the relevant statistics).

*Table 1.* Training loss of Algorithms 1 and 2 on different losses, where the global optimal is zero.

| loss | Alg 1 | Alg 2 |
|---|---|---|
| Predictability | $1.90 \pm 0.36$ | $(2.5 \pm 7.2) \times 10^{-4}$ |
| Suboptimality | $0.03 \pm 0.05$ | $(8.6 \pm 2.1) \times 10^{-4}$ |

## 3.3. Convex and Mixed-integer Reformulations

In this section, we discuss choices of $g_{\boldsymbol{\theta}}$ for which problems (10) can be cast as convex or mixed-integer programs which can be solved using off-the-shelf solvers. Restricting $\boldsymbol{A}_\theta(\boldsymbol{s}) = \alpha \in \mathbb{R}_+$, i.e., $\boldsymbol{\theta} = (\alpha, \{\boldsymbol{b}_k\}_{k=0}^K)$ will achieve a convex reformulation. This approximation only allows for scaling and translation of the primitive set $\mathcal{Z}$ and does not permit rotation or projection. The following proposition provides the convex reformulation of the predictability loss. The suboptimality loss can be reformulated in a similar way.

**Proposition 3.7** (Tractable convex reformulation)**.** *Let $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ defined in (17) with $\boldsymbol{A}_\theta(\boldsymbol{s}) = \alpha \in \mathbb{R}_+$ and $\boldsymbol{\theta} = (\alpha, \{\boldsymbol{b}_k\}_{k=0}^K)$. Then, the predictability loss (10) can be reformulated as the convex optimization*

$$\begin{aligned}
\min\ & \frac{1}{N}\sum_{i=1}^{N}\|\gamma_i\| \\
\text{s.t. } & \alpha \in \mathbb{R}_+,\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k \le K, \\
& \left.\begin{array}{l}
\boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{\zeta}_i \in \mathbb{R}^p,\ \boldsymbol{\lambda}_i \in \mathcal{K}^* \\
\boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{\zeta}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) \\
H\boldsymbol{\zeta}_i - \alpha\boldsymbol{h} \in \mathcal{K} \\
\boldsymbol{\alpha} c(\boldsymbol{s}_i)^\top - \boldsymbol{\lambda}_i^\top H = 0 \\
\boldsymbol{c}(\boldsymbol{s}_i)^\top(\boldsymbol{x}_i + \boldsymbol{\gamma}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \le 0
\end{array}\right\}\ \forall i \le N.
\end{aligned} \tag{13}$$

The auxiliary variable $\boldsymbol{\zeta}$ replaces $\alpha\boldsymbol{z}$ in constraints $\boldsymbol{x} + \boldsymbol{\gamma} = \alpha\boldsymbol{z} + \boldsymbol{b}(\boldsymbol{s})$ and $H\alpha\boldsymbol{z} - \alpha\boldsymbol{h} \in \mathcal{K}$, where the latter stems from multiplying $H\boldsymbol{z} - \boldsymbol{h} \in \mathcal{K}$ by $\alpha \in \mathbb{R}_+$. Despite its limitations, this choice of hypothesis class can enhance computational efficiency for large-scale problems.

Restricting either $\boldsymbol{z}$ or $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})$ to be binary, allows to reformulate the bilinear term $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z}$ using linear inequalities (McCormick inequalities) and formulating the problem as a mixed-interger linear program. The application domain will dictate which of the two will be binary, with two interesting examples emerging. For the first, notice that setting $\mathcal{Z}_{\text{bin}} := \mathcal{Z} \cap \{0,1\}^p$ constitutes a restriction to the hypothesis class. The next observation provides conditions under which the restriction to the integer $\mathcal{Z}$ is done without loss of optimality.

**Observation 3.8** (Discrete vs. continuous primitive sets)**.** *Let $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ given in (17) with $\mathcal{Z}_{bin} = \{\boldsymbol{z} \in \{0,1\}^p : \boldsymbol{e}^\top \boldsymbol{z} = 1\}$, and let $\boldsymbol{\theta}^*$ be the optimal value of problem (10) using the predictability loss. If $\boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}_{\boldsymbol{\theta}^*}(\boldsymbol{s})$ is not parallel to any of the facets of the simplex for all $\boldsymbol{s}$, then the optimal value of the problem will coincide with the optimal value of the predictability loss problem (10) where $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^p_+ : \boldsymbol{e}^\top \boldsymbol{z} = 1\}$.*

For the second case, assume that $g_{\boldsymbol{\theta}}$ given in (17) with $\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \boldsymbol{A} \in \{-1,0,1\}^{n \times p}$ and $\boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}) : \Theta \mapsto \mathbb{R}^n$ for all $\boldsymbol{s}$. This choice of the hypothesis class can be useful for learning equality constraints, for example, learning the physical links in a network where $\boldsymbol{A}$ dictates the connectivity of the network. Notice that the number of binary variables needed for the reformulation of the problem is independent to the size of the dataset. We will see an example of this hypothesis class used in Section 4.2 to learn the structure of a power network.

## 4. Numerical Experiments

We apply our methods on an instance of a power system described in (Bampou & Kuhn, 2011). Both problems have the same forward problem, i.e., a network flow problem with the following formulation. We consider a power system consists of a set of regions $\mathcal{R} = \{1, \cdots, 5\}$ with electricity demands $s^{\text{demand}}_r$, $r \in R$. Demands are satisfied by a set $\mathcal{N} = \{1,2,3\}$ of power plans, where each plant $n \in \mathcal{N}$ produces $x_n$ units of energy at costs $s^{\text{cost}}_n$. Regions are connected by a set $\mathcal{M} = \{1, \cdots, 5\}$ of directed transmission lines. Each line $m \in \mathcal{M}$ has a capacity of $\bar{f}_m$ units of energy. A pictorial representation of the network is given in Figure 3 (top).

The forward problem is formulated as follows where $\mathcal{N}(r)$ denotes the set of generators in node $r$, with $\mathcal{M}_+(r)$ and $\mathcal{M}_-(r)$ denoting the sets of incoming and outgoing flows from node $r$, respectively.

$$\begin{array}{ll} \min & {\boldsymbol{s}^{\text{cost}}}^\top \boldsymbol{x} \\ \text{s.t.} & x_n \in \mathbb{R}_+,\ x_n \leq C_n,\ \forall n \in \mathcal{N}, \\ & f_m \in \mathbb{R},\ |f_m| \leq \bar{f}_m,\ \forall m \in \mathcal{M}, \\ & \sum_{n \in \mathcal{N}(r)} x_n + \sum_{m \in \mathcal{M}_+(r)} f_m \\ & \quad = \sum_{m \in \mathcal{M}_-(r)} f_m + s^{\text{demand}}_r,\ \forall r \in \mathcal{R}. \end{array} \tag{14}$$

We set $C_n = 3.5$ for all $n \in \mathcal{N}$ and $\bar{f}_m = 3.5$ for all $m \in \mathcal{M}$. We generate $N_{\text{train}} = 100$ data points for training and $N_{\text{test}} = 200$ data points for testing, by generating signals uniformly at random from $s^{\text{cost}}_1 \in [0.2, 1]$, $s^{\text{cost}}_2 \in [0.2, 0.5]$, $s^{\text{cost}}_3 \in [1, 2]$ and $s^{\text{demand}}_1 \in [0.3, 1.5]$, $s^{\text{demand}}_2 \in [0.36, 1.8]$, $s^{\text{demand}}_3 \in [0.42, 2.1]$, $s^{\text{demand}}_4 \in [0.48, 2.4]$ and $s^{\text{demand}}_5 \in [0.54, 2.7]$, and solving problem (14) to obtain pairs $\{\boldsymbol{s}_i, \boldsymbol{x}_i\}_{i=1}^{N=100}$. Notice that the flow decisions $f_m$ are treated as lurking variables and are not observed.

### 4.1. Problem 1: Inferring Generation Policy without Knowing Constraints

In the first experiment, we assume the decision-maker is aware of the objective function of the forward problem but lacks knowledge of the constraint's structure. This scenario intuitively represents a situation where decision-makers aim to minimize total costs without being aware of any specific business rules. We compare four policies: $(i)$ We use the hypothesis (17) where $\mathcal{Z}$ is the unit simplex of dimension $p \in \{3, 6, 9\}$ using the Adaptive Smoothing Algorithm 2 with a limit of 3000 iterations, $(ii)$ we use the convex formulation discussed in Section 3.3 using the unit simplex of dimension $p = 3$, $(iii)$ we compare against the linear policy induced by learning the quadratic function $f_{\boldsymbol{\theta}}(\boldsymbol{x}, s) = \boldsymbol{x}^\top Q_{\boldsymbol{\theta}} \boldsymbol{x} + \boldsymbol{x}^\top A_{\boldsymbol{\theta}} \boldsymbol{s}$, see Example 3.2, and $(iv)$ we use the hypothesis (17) where $\mathcal{Z}$ is the unit simplex of dimension $p \in \{3, 6, 9\}$ and solve problems (10) using the non-convex quadratic solver of Gurobi v11.0.3 with a time limit of 1800 seconds. We note that policy $(iii)$ will coincide with the linear regression policy $\min_{\boldsymbol{\theta} \in \Theta} \sum_{i=1}^{N_{\text{train}}} \|\boldsymbol{x}_i - (\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i))\|_2^2$ since the constraints of the problem are assumed unknown. For all methods, we solve both the corresponding predictability and suboptimality loss problems. Table 2 shows the out-of-sample performance of each method using the predictability $\ell^{\text{p}}_\theta$ and suboptimality $\ell^{\text{sub}}_\theta$ losses, as well as the true predictability $\ell^{\text{p}}$ and suboptimality $\ell^{\text{sub}}$ losses. As discussed in Section 2.2, by definition $\ell^{\text{p}}_\theta = \ell^{\text{p}}$. We observe the following: $(i)$ The convex formulation is the most computationally efficient but may yield lower quality solutions compared to other methods. $(ii)$ Learning a quadratic cost is significantly inferior to other methods. $(iii)$ Increasing the dimension $p$ of the unit simplex enhances the hypothesis class flexibility, improving out-of-sample performance for both the predictability and suboptimality loss. $(iv)$ Algorithm 2 performs significantly better compared to using Gurobi in almost all cases, particularly the most complex ones. We further investigate the performance of the proposed approach using the larger IEEE 14-bus system (Leon et al., 2020). The results are presented in Appendix F.

### 4.2. Problem 2: Learning Power Network Structures

In the second experiment, we assume the decision-maker knows the capacity constraints and demand locations but

*Table 2.* Summary of out-of-sample performances.

| Hypothesis Class and Loss | | Evaluation method | | | |
|---|---|---|---|---|---|
| | loss | $\ell^p_\theta/\ell^p$ | $\ell^{\text{sub}}_\theta$ | $\ell^{\text{sub}}$ | time |
| Alg 2, $p=3$ | predict. | 3.20 | 0.17 | 0.83 | 569.0s |
| | subopt. | 5.50 | 0.18 | 2.45 | 430.8s |
| Alg 2, $p=6$ | predict. | 1.91 | 0.05 | 0.51 | 1159.6s |
| | subopt. | 1.50 | 0.03 | 0.42 | 754.1s |
| Alg 2, $p=9$ | predict. | 1.58 | 0.05 | 0.69 | 591.7s |
| | subopt. | 1.61 | 0.04 | 0.50 | 590.6$s$ |
| Convex, $p=3$ | predict. | 3.36 | 2.07 | 0.34 | 0.31s |
| | subopt. | 10.75 | 0.31 | 7.72 | 0.15$s$ |
| quadratic cost | predict. | 2.58 | 2.56 | 0.63 | 2.54s |
| Gurobi $p=3$ | predict. | 6.32 | 4.44 | 0.49 | 1800s |
| | subopt. | 3.36 | 0.05 | 1.30 | 1800s |
| Gurobi $p=6$ | predict. | 6.54 | 4.31 | 0.76 | 1800s |
| | subopt. | 7.76 | 0.15 | 4.96 | 1800s |
| Gurobi $p=9$ | predict. | 6.66 | 4.25 | 0.77 | 1800s |
| | subopt. | 11.00 | 4.63 | 5.19 | 1800s |

is unaware of the transmission line positions. The goal is to recover the network structure using inverse optimization, in effect learning the last constraint of problem (14). In a network with 5 nodes, there are 10 possible transmission line connections, and we aim to identify the configuration that best fits the data. Using the same experimental setup as the previous example, we treat the flow decisions as latent variables. Leveraging the structure of (17), we can re-express the policy $\boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{A_\theta}(\boldsymbol{s})\boldsymbol{z}(\boldsymbol{s}) + \boldsymbol{b_\theta}(\boldsymbol{s}), \boldsymbol{z}(\boldsymbol{s}) \in \mathcal{Z}$ as the following 5 constraints, each corresponding to a node in the network

$$\hat{\boldsymbol{x}}(\boldsymbol{s}) = \boldsymbol{A}\boldsymbol{z}(\boldsymbol{s}) + \boldsymbol{s}^{\text{demand}},\ \boldsymbol{z}(s) \in \mathcal{Z} = [-\bar{f}_m, \bar{f}_m]^{10},$$

where $\hat{\boldsymbol{x}}(\boldsymbol{s}) = [x_1(\boldsymbol{s}), 0, x_2(\boldsymbol{s}), 0, x_3(\boldsymbol{s})]^\top$ denotes the injection of energy at in the network. Notice that generators exist in nodes 1, 3 and 5, see Figure 3 (top). The decision matrix $\boldsymbol{A} \in \{0,1\}^{5\times 10}$ controls the connectivity of the network, e.g., if $A_{1,2} = 1$, transmission line 2 emanates from node 1. Note that $\boldsymbol{A}$ is subject to additional constraints to ensure that there are only $C_5^2 = 10$ potential line connections. The auxiliary vector $\boldsymbol{z}(\boldsymbol{s})$ represents the energy flow in the transmission lines. Therefore, we define the primitive set as a hyper-rectangle $\mathcal{Z} = [-\bar{f}_m, \bar{f}_m]^{10}$, which mirrors the transmission capacities. Since we assume that the locations of the generators are known, the given matrix $G$ dictates their position in the system. The inverse optimization problem is presented in Appendix G and optimizes over matrices $\boldsymbol{A}$ . As discussed in Section 3.3, the problem can be cast as a mixed-integer linear program for both the predictability and suboptimality losses. Both approaches achieved zero loss indicating that the recovered problem is able to exactly describe the training data. The recovered network is presented in Figure 3 (bottom) in the appendix. It is interesting to note that the two networks have different transmission line configuration, with the recovered network having 4 lines instead of 5 that the original network has. Nevertheless,

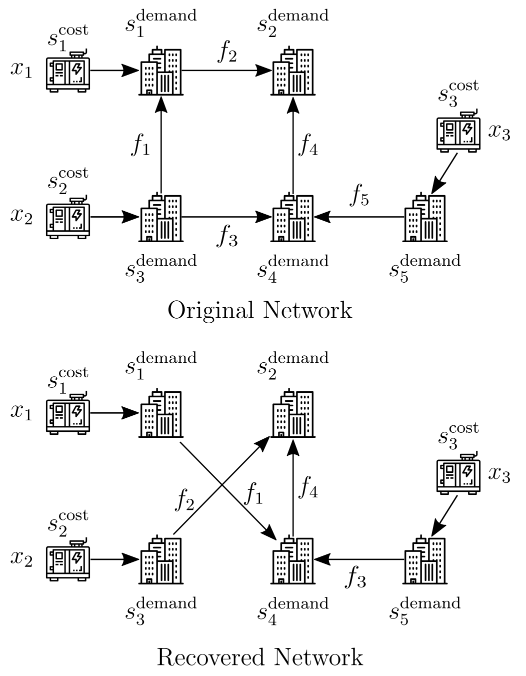

*Figure 3.* **Top:** Power network topology used in the forward problem in Section 4. **Bottom:** Recovered network from inverse optimization in Section 4.2.

even with 4 lines, the recovered problem is able exactly match the observed data and achieve zero loss. Finally, we compare our approach with the state-of-the-art method for learning problem constraints proposed by (Aswani et al., 2018), which also utilizes predictability loss. Their method involves enumerating all potential solutions, assessing the loss for each configuration. It is worth noting that there are 10 potential line connections, each of which can either be absent or present, resulting in $2^{10}$ configurations. We estimated that evaluating each configuration takes on average of 2.64 seconds, thus enumerating all possibilities would take approximately $2.64 \times 2^{10}$ seconds to find the global optimum. In contrast, our proposed method solves the training problem in just 23.5 seconds.

## 5. Limitations and Future Work

In closing, we acknowledge several limitations: $(i)$ Our method supports only known linear objectives, excluding more complex structures like quadratic objectives which will be interesting to examine further. However, as shown in Example 3.2, the proposed approach does creates complex policy structures, unlike the linear policy induced from learning a quadratic objective. $(ii)$ It is worth exploring further cases where the hypothesis class leads to convex and MILP formulations (similar to Section 3.3), reducing reliance on local search algorithms. $(iii)$ Our hypothesis class induces a non-trivial relationship between the choice of the primitive set and resulting policy behavior. Future work will focus on understanding the impact of the primitive set on the policies it generates, akin to the choice of basis functions in classical linear regression.

## Acknowledgements

This work was partly supported by the European Research Council (ERC) project TRUST-949796.

## 6. Impact Statements

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

Ajayi, T., Lee, T., and Schaefer, A. J. Objective selection for cancer treatment: An inverse optimization approach. *Operations Research*, 70(3):1717–1738, 2022.

Akhtar, S. A., Kolarijani, A. S., and Mohajerin Esfahani, P. Learning for control: An inverse optimization approach. *IEEE Control Systems Letters*, 6:187–192, 2021.

Aswani, A., Shen, Z.-J., and Siddiq, A. Inverse optimization with noisy data. *Operations Research*, 66(3):870–892, 2018.

Ayer, T. Inverse optimization for assessing emerging technologies in breast cancer screening. *Annals of Operations Research*, 230:57–85, 2015.

Bampou, D. and Kuhn, D. Scenario-free stochastic programming with polynomial decision rules. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 7806–7812, 2011.

Bärmann, A., Pokutta, S., and Schneider, O. Emulating the expert: inverse optimization through online learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 400–410, 2017.

Bertsekas, D. *Convex optimization algorithms*. Athena Scientific, 2015.

Bertsekas, D. P. *Nonlinear programming*. Athena Scientific Optimization and Computation Series. Athena Scientific, Belmont, MA, second edition, 1999.

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

Chan, T. C. and Kaw, N. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 282(2):415–427, 2020.

Chen, L., Chen, Y., and Langevin, A. An inverse optimization approach for a capacitated vehicle routing problem. *European Journal of Operational Research*, 295(3):1087–1098, 2021.

Dempe, S. and Lohse, S. Inverse linear programming. In *Recent Advances in Optimization*, pp. 19–28. Springer, 2006.

Dimanidis, I., Ok, T., and Mohajerin Esfahani, P. Offline reinforcement learning via inverse optimization. *preprint available at arXiv:2502.20030*, 2025.

Dong, C., Chen, Y., and Zeng, B. Generalized inverse optimization through online learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Elmachtoub, A. N. and Grigas, P. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

Ghobadi, K. and Mahmoudzadeh, H. Inferring linear feasible regions using inverse optimization. *European Journal of Operational Research*, 2020.

Güler, c. and Hamacher, H. W. Capacity inverse minimum cost flow problem. *Journal of Combinatorial Optimization*, 19(1):43–59, 2010.

Keshavarz, A., Wang, Y., and Boyd, S. Imputing a convex objective function. In *2011 IEEE international symposium on intelligent control*, pp. 613–619, 2011.

Leon, L. M., Bretas, A. S., and Rivera, S. Quadratically constrained quadratic programming formulation of contingency constrained optimal power flow with photovoltaic generation. *Energies*, 13(13):3310, 2020.

Li, J. Y.-M. Inverse optimization of convex risk functions. *Management Science*, 67(11):7113–7141, 2021.

Lu, T., Wang, Z., Wang, J., Ai, Q., and Wang, C. A data-driven stackelberg market strategy for demand response-enabled distribution systems. *IEEE Transactions on Smart Grid*, 10(3):2345–2357, 2018.

Mohajerin Esfahani, P., Shafieezadeh-Abadeh, S., Hanasusanto, G. A., and Kuhn, D. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167:191–234, 2018.

Nesterov, Y. Smooth minimization of non-smooth functions. *Mathematical programming*, 103:127–152, 2005.

Nesterov, Y. et al. *Lectures on Convex Optimization*, volume 137. Springer, 2018.

Saez-Gallego, J., Morales, J. M., Zugno, M., and Madsen, H. A data-driven bidding model for a cluster of price-responsive consumers of electricity. *IEEE Transactions on Power Systems*, 31(6):5001–5011, 2016.

Tan, Y., Terekhov, D., and Delong, A. Learning linear programs from optimal decisions. *arXiv preprint arXiv:2006.08923*, 2020.

Xu, Z., Deng, T., Hu, Z., Song, Y., and Wang, J. Data-driven pricing strategy for demand-side resource aggregators. *IEEE Transactions on Smart Grid*, 9(1):57–66, 2016.

Zattoni Scroccaro, P., Atasoy, B., and Mohajerin Esfahani, P. Learning in inverse optimization: Incenter cost, augmented suboptimality loss, and algorithms. *Operations Research*, 2024.

Zattoni Scroccaro, P., van Beek, P., Mohajerin Esfahani, P., and Atasoy, B. Inverse optimization for routing problems. *Transportation Science*, 59(2):301–321, 2025.

Zhang, J. and Paschalidis, I. C. Data-driven estimation of travel latency cost functions via inverse optimization in multi-class transportation networks. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 6295–6300, 2017.

## A. Proof of Proposition 2.1

*Proof.* We first prove the statement for the predictabilily loss. Assume that there exists $\boldsymbol{\theta} \in \Theta$ and $(\boldsymbol{x}, \boldsymbol{s})$ in the training dataset such that $\ell^{\mathrm{p}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = 0$. This implies that there exists $\boldsymbol{\gamma} = 0$ since the objective of the predictability loss in (5) is $\|\gamma\|$. Thus, the constraints are satisfied with $g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq 0$ which implies that $\boldsymbol{x}$ is feasible in (2). Moreover, by construction it implies that

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \geq \begin{bmatrix} \min\limits_{\boldsymbol{y} \in \mathbb{R}^n} & f_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{s}) \\ \text{s.t.} & g_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{s}) \leq 0 \end{bmatrix}. \tag{15}$$

In addition, since $\boldsymbol{\gamma} = 0$ it implies that $J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq 0$, which combined with (15) imply that indeed $\boldsymbol{x}$ is an optimizer of (2).

To prove the reverse implication, consider any optimal $\boldsymbol{x}$ from (2). Since $\boldsymbol{x}$ is feasible and optimal in (2) it implies that $\boldsymbol{\gamma} = 0$ is feasible in (5). Moreover, since the objective function in (5) is $\|\gamma\|$, there does not exist another $\boldsymbol{\gamma} \in \mathbb{R}^n$ that achieves a lower objective, thus $\boldsymbol{\gamma} = 0$ is also optimal, hence $\ell^{\mathrm{p}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = 0$, which concludes the proof. The proof for the suboptimality loss follows similar arguments. □

## B. Proof of Theorem 3.3

The proof of Theorem 3.3 can be obtained from the following lemma.

**Lemma B.1.** *Let $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$. The constraints of the predictability loss in* (5) *can be reformulated as follows*

$$\begin{aligned} g_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\gamma}, \boldsymbol{s}) \leq 0 \quad &\iff \quad \begin{cases} \exists \boldsymbol{z} \in \mathcal{Z} \\ \boldsymbol{x} + \boldsymbol{\gamma} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}) \end{cases} \\ J_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\gamma}, \boldsymbol{s}) \leq 0 \quad &\iff \quad \begin{cases} \exists \boldsymbol{\lambda} \in \mathcal{K}^* \\ \boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x} + \boldsymbol{\gamma} - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})) - \boldsymbol{\lambda}^\top \boldsymbol{h} \leq 0 \\ \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\lambda}^\top H = 0 \end{cases} \end{aligned} \tag{16a}$$

*and the constraints of the suboptimality loss in* (5) *can be reformulated as follows*

$$\begin{aligned} g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq \gamma_f \quad &\iff \quad \begin{cases} \exists \boldsymbol{z} \in \mathcal{Z},\, \boldsymbol{\gamma} \in \mathbb{R}^n \\ \boldsymbol{x} + \boldsymbol{\gamma} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}) \\ \|\gamma\| \leq \gamma_f \end{cases} \\ J_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) \leq \gamma_o \quad &\iff \quad \begin{cases} \exists \boldsymbol{\lambda} \in \mathcal{K}^* \\ \boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x} - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})) - \boldsymbol{\lambda}^\top \boldsymbol{h} \leq \gamma_o \\ \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\lambda}^\top H = 0 \end{cases} \end{aligned} \tag{16b}$$

*where the norm used is the same as in the definition of $g_{\boldsymbol{\theta}}$.*

*Proof.* Recall that the definition of $g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s})$ is

$$g_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \min_{\boldsymbol{z} \in \mathcal{Z}} \|\boldsymbol{x} - \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s})\|. \tag{17a}$$

The function uses a latent variable $\boldsymbol{z}$, which resides in the predetermined conic *primitive set*

$$\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^p \,:\, H\boldsymbol{z} - \boldsymbol{h} \in \mathcal{K}\} \tag{17b}$$

Plug the above definition into $g_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\gamma}, \boldsymbol{s}) \leq 0$ gives:

$$\begin{aligned} &\exists \boldsymbol{z} \in \mathcal{Z},\, \boldsymbol{\gamma} \in \mathbb{R}^n \\ &\boldsymbol{x} + \boldsymbol{\gamma} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}). \end{aligned}$$

The reformulation of $J_{\boldsymbol{\theta}}$ is done via the dualization.

$$J_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\gamma}, \boldsymbol{s}) \leq 0$$

is equivalent to

$$\boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x}+\boldsymbol{\gamma}) - \begin{bmatrix} \min & \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{y} \\ \text{s.t.} & \boldsymbol{y} \in \mathbb{R}^n, \\ & \boldsymbol{y} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}) \\ & H\boldsymbol{z} - \boldsymbol{h} \in \mathcal{K},\ \boldsymbol{\gamma} \in \mathbb{R}^n \end{bmatrix} \leq 0$$

After dualizing the second term, we obtain:

$$\boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x}+\boldsymbol{\gamma}) - \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{\lambda}^\top \boldsymbol{h} \leq 0 \tag{18}$$

$$\boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\lambda}^\top H = 0 \tag{19}$$

The proof of suboptimality follows the same procedure.

□

## C. Proof of Proposition 3.6

*Proof.* The proof for both Algorithms 1 and 2 follows the same arguments. First, note that the monotonicity of the proposed algorithms' outcome is a straightforward consequence of its coordinate descent nature; see (Bertsekas, 2015, Section 6.5) for similar techniques. More specifically, a classical result of gradient decent algorithms with several popular stepsizes, including the Armijo rule used in this work, ensures that the desired loss function is monotonically decreasing over the iterations (Bertsekas, 2015, Section 2.1). This observation of the gradient descent over the coordinate $\boldsymbol{A}$ of $\boldsymbol{\theta}$ , along with solving the convex optimization in (10) over the coordinate $\boldsymbol{b}$ and the fact that the loss function is uniformly nonnegative (bounded from below), concludes that the loss function remains monotonically non-increasing across the iterations. □

## D. MILP formulation and Proposition D.1

**Proposition D.1** (MILP reformuation)**.** *Let $f_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ given in* (17) *with $\mathcal{Z} = \{\boldsymbol{z} \in \{0,1\}^p : \boldsymbol{e}^\top \boldsymbol{z} = 1\}$. The predictability loss in* (5) *can be reformulated as*

$$\begin{array}{ll} \min & \|\boldsymbol{\gamma}\| \\ \text{s.t.} & \boldsymbol{\gamma} \in \mathbb{R}^n,\ \boldsymbol{z} \in \{0,1\}^p,\ \lambda \in \mathbb{R} \\ & \boldsymbol{x} + \boldsymbol{\gamma} = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}(\boldsymbol{s}) \\ & \boldsymbol{e}^\top \boldsymbol{z} = 1 \\ & \boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x}+\boldsymbol{\gamma}) - \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{b}(\boldsymbol{s}) + \lambda \leq 0 \\ & \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \lambda \boldsymbol{e}^\top = 0 \end{array} \tag{20}$$

*Proof.* The predictability loss with $f_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ given in (17) with $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^p_+ : \boldsymbol{e}^\top \boldsymbol{z} = 1\}$ can be written as

$$\begin{array}{ll} \min & \|\boldsymbol{\gamma}\| \\ \text{s.t.} & \boldsymbol{\gamma} \in \mathbb{R}^n,\ \boldsymbol{z} \in \mathbb{R}^p_+ \\ & \boldsymbol{x} + \boldsymbol{\gamma} = \boldsymbol{A}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}(\boldsymbol{s}), \quad \boldsymbol{e}^\top \boldsymbol{z} = 1 \\ & \boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x}+\boldsymbol{\gamma}) \leq \begin{bmatrix} \min & \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{y} \\ \text{s.t.} & \boldsymbol{y} \in \mathbb{R}^n,\ \boldsymbol{z} \in \mathbb{R}^p_+ \\ & \boldsymbol{y} = \boldsymbol{A}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}(\boldsymbol{s}) \\ & \boldsymbol{e}^\top \boldsymbol{z} = 1 \end{bmatrix} \end{array} \tag{21}$$

It is clear that the optimal value of (20) constitutes an upper bound on the optimal value of (21) since $\boldsymbol{z} \in \{0,1\}^p$ is a restriction to $\boldsymbol{z} \in \mathbb{R}^p_+$. Let $\boldsymbol{\gamma}^*$ be an optimal solution of problem (21). We next show that $\boldsymbol{\gamma}^*$ is feasible in (20) which will conclude the proof.

From problem (21), since $\boldsymbol{x} + \boldsymbol{\gamma}^*$ is a feasible solution in $\boldsymbol{x} + \boldsymbol{\gamma}^* = \boldsymbol{A}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}(\boldsymbol{s})$, $\boldsymbol{e}^\top \boldsymbol{z} = 1$, it implies that $\boldsymbol{x} + \boldsymbol{\gamma}^*$ is also feasible in

$$\begin{array}{rll} V_{\boldsymbol{\theta}}(\boldsymbol{s}) = & \min & \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{y} \\ & \text{s.t.} & \boldsymbol{y} \in \mathbb{R}^n,\ \boldsymbol{z} \in \mathbb{R}^p_+ \\ & & \boldsymbol{y} = \boldsymbol{A}(\boldsymbol{s})\boldsymbol{z} + \boldsymbol{b}(\boldsymbol{s}) \\ & & \boldsymbol{e}^\top \boldsymbol{z} = 1 \end{array} \tag{22}$$

The last constraint in (21) also implies that $\boldsymbol{x} + \boldsymbol{\gamma}^*$ is optimal in (22). Additionally, notice that since (22) is a linear program and $\boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{A}(\boldsymbol{s})$ is not parallel to any of the facets of the simplex, there exists a unique corner point in the simplex $\boldsymbol{e}^\top \boldsymbol{z} = 1$, i.e., $\boldsymbol{z}' \in \{0,1\}^p$, such that $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{s})\boldsymbol{z}' + \boldsymbol{b}(\boldsymbol{s})$ achieves the optimal value. Hence, constraint $\boldsymbol{c}(\boldsymbol{s})^\top(\boldsymbol{x} + \boldsymbol{\gamma}) \leq V_{\boldsymbol{\theta}}(\boldsymbol{s})$ ensures that only $\boldsymbol{z}'$ is feasible, thus $\boldsymbol{x} + \boldsymbol{\gamma}^*$ is feasible in problem (21), which concludes the proof. $\square$

## E. Smoothing

Despite that both the predictability and suboptimality loss are non-convex for the hypothesis class $g_{\boldsymbol{\theta}}$ given in (17) with and arbitrary choice of $\mathcal{Z}$, Algorithm 1 tends to be trapped in local minima much more often for the predictability than the suboptimality loss. In this section, we explain this fact by showing that suboptimality loss can be viewed as a smoothed version of predictability loss in the sense of the so-called Nesterov's smoothing. Inspired by this technique, we propose an adaptive smoothing algorithm to potentially escape the local optimum, hence improving the performance.

**Definition E.1** (Nesterov's smoothing (Nesterov et al., 2018))**.** *Consider the function in the form of $J(\boldsymbol{x}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \langle \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{y}\rangle - \phi(\boldsymbol{y})$ where $\phi$ is a convex function. We define the smooth counterpart of $f_\epsilon$ as*

$$J_\epsilon(\boldsymbol{x}) := \max_{\boldsymbol{y}\in\mathcal{Y}} \langle \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{y}\rangle - \phi(\boldsymbol{y}) - \epsilon d(\boldsymbol{y}),$$

*where $\epsilon > 0$ is the smoothing parameter, and $d(\boldsymbol{y})$ is called a prox function that (i) is continuous and 1-strongly convex on $\mathcal{Y}$ and (ii) $\min_{\boldsymbol{y}\in\mathcal{Y}} d(\boldsymbol{y}) = 0$. Then, $J_\epsilon(x)$ is $1/\epsilon$-smooth (i.e., its gradient $\nabla J_\epsilon(\boldsymbol{x})$ is $1/\epsilon$-Lipschitz continuous), and $J_\epsilon(\boldsymbol{x}) \leq J(\boldsymbol{x}) \leq J_\epsilon(\boldsymbol{x}) + \epsilon/2$ for all $\boldsymbol{x} \in \mathcal{Y}$.*

**Suboptimality loss as smoothed predictability loss.**

Considering the optimization problems (10), we define $\boldsymbol{\beta}_i$ as the dual multiplier of the equality constraints $\boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)$, and the respective penalization Lagrangian function

$$J(\boldsymbol{\theta}, \boldsymbol{Z}) := \max_{\boldsymbol{\beta}_i} \sum_{i=1}^{n} \boldsymbol{\beta}_i^\top \big(\boldsymbol{x}_i + \boldsymbol{\gamma}_i - \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\big) \quad \text{where} \quad \boldsymbol{Z} = \{\gamma_i, z_i\}_{i\leq N}\,. \tag{23}$$

Dualizing this linear constraint in the predictability loss program (10a) reformulates the program to

$$\begin{aligned} \min\ & \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{\gamma}_i\| + J(\boldsymbol{\theta}, \boldsymbol{Z}) \\ \text{s.t. } & \boldsymbol{A}_k \in \mathbb{R}^{n\times p},\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k \leq K, \\ & \left.\begin{array}{l} \boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{z}_i \in \mathbb{R}^p,\ \boldsymbol{\lambda}_i \in \mathcal{K}^* \\ H\boldsymbol{z}_i - \boldsymbol{h} \in \mathcal{K} \\ \boldsymbol{c}(\boldsymbol{s}_i)^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{\lambda}_i^\top H = 0 \\ \boldsymbol{c}(\boldsymbol{s}_i)^\top(\boldsymbol{x}_i + \boldsymbol{\gamma}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \leq 0 \end{array}\right\} \forall i \leq N \end{aligned} \tag{24}$$

Considering the prox function $d(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta} + \boldsymbol{\gamma}_i/\epsilon\|^2$, the smoothed version of the function $J$ is

$$J_\epsilon(\boldsymbol{\theta}, \boldsymbol{Z}) = \sum_{i=1}^{N} \frac{1}{2\epsilon}\|\boldsymbol{x}_i - \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\|^2 - \frac{1}{2\epsilon}\|\boldsymbol{\gamma}_i^2\|. \tag{25}$$

Replacing the above smoothed term in the predictability loss (24) yields the objective function

$$\sum_{i=1}^{N} \frac{1}{2\epsilon}\|\boldsymbol{x}_i - \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\|^2 + \left(\frac{1}{N} - \frac{1}{2\epsilon}\right)\|\boldsymbol{\gamma}_i\|. \tag{26}$$

If the smoothing level is $\epsilon = N/2$, and the norm in the objective (10b) is separable (i.e., $\|(\gamma_{f,i}, \gamma_{o,i})\| = |\gamma_{f,i}| + |\gamma_{o,i}|$), one can then see that the second term in (26) is cancelled and the first term coincides with the optimal solution $\gamma_{f,i}$ in the objective of the suboptimality loss in (10b). In other words, the variable penalizing the feasibility of each data point effectively is a smoothed version of the corresponding term in the predictability loss.

### E.1. Adaptive smoothing

We presented the complete version with suboptimality loss in the following.

**Definition E.2.** *Inspired by the results above , we define smoothed predictability and suboptimality loss in the following. Again, for clarity, we assume $f_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{c}(\boldsymbol{s})^\top \boldsymbol{x}$ and $g_{\boldsymbol{\theta}}$ given in (17). The learning problem using the predictability loss can be reformulated as follows*

$$\begin{array}{ll} \min & \displaystyle\frac{1}{N}\sum_{i=1}^{n}\|\boldsymbol{\gamma}_i\| + \epsilon_1 \sum_{i=1}^{n}\|\boldsymbol{\gamma}_{s1}\| + \epsilon_2 \sum_{i=1}^{n}\|\boldsymbol{\gamma}_{s2}\| \\ \text{s.t.} & \boldsymbol{A}_k \in \mathbb{R}^{n\times p},\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k = 1,\dots,K, \\ & \left.\begin{array}{l} \boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{z}_i \in \mathbb{R}^p,\ \boldsymbol{\lambda}_i \in \mathcal{K}^* \\ \boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) + \boldsymbol{\gamma}_{s1} \\ H\boldsymbol{z}_i - \boldsymbol{h} \in \mathcal{K} \\ \boldsymbol{c}(\boldsymbol{s}_i)^\top(\boldsymbol{x}_i + \boldsymbol{\gamma}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \le 0 \\ \boldsymbol{c}(\boldsymbol{s}_i)^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{\lambda}_i^\top H + \boldsymbol{\gamma}_{s2} = 0 \end{array}\right\} \forall i \le N \end{array} \tag{27a}$$

*while the learning problem using the suboptimality loss can be reformulated as follows*

$$\begin{array}{ll} \min & \displaystyle\frac{1}{N}\sum_{i=1}^{n}\|(\gamma_{f,i}, \gamma_{o,i})\| + \epsilon_1 \sum_{i=1}^{n}\|\boldsymbol{\gamma}_{s1}\| \\ \text{s.t.} & \boldsymbol{A}_k \in \mathbb{R}^{n\times p},\ \boldsymbol{b}_k \in \mathbb{R}^n,\ \forall k = 1,\dots,K, \\ & \left.\begin{array}{l} \gamma_{f,i}, \gamma_{o,i} \in \mathbb{R}_+ \\ \boldsymbol{\gamma}_i \in \mathbb{R}^n,\ \boldsymbol{z}_i \in \mathbb{R}^p,\ \boldsymbol{\lambda}_i \in \mathcal{K}^* \\ \boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\boldsymbol{z}_i + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) \\ \|\boldsymbol{\gamma}_i\| \le \gamma_{f,i} \\ H\boldsymbol{z}_i - \boldsymbol{h} \in \mathcal{K} \\ \boldsymbol{c}(\boldsymbol{s}_i)^\top(\boldsymbol{x}_i - \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{s}_i)) - \boldsymbol{\lambda}_i^\top \boldsymbol{h} \le \gamma_{o,i} \\ \boldsymbol{c}(\boldsymbol{s}_i)^\top \boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{s}_i) - \boldsymbol{\lambda}_i^\top H + \boldsymbol{\gamma}_{s1} = 0 \end{array}\right\} \forall i \le N \end{array} \tag{27b}$$

### E.2. Convergence behavior of adaptive smoothing

In this section, we study the convergence behavior of Algorithm 2 using the experiment setting described in Section H.2. We plot the values of training loss and the four metrics defined in 2.2 to validate that the proposed method improves the performance of the solution. The plots are shown in Figure 4, where the training loss increases as more regularization is added to the optimization. However, the true losses keep decreasing implying better optimal solutions are being found each time.

Finally, it is worth noting that utilizing the algorithm without smoothing leads to inconsistent outcomes, particularly concerning predictability losses. Conversely, employing an adaptive smoothing algorithm here ensures consistent results are obtained. We record this behavior via an example in Figure 5. The experiment settings follow the descriptions introduced in Appendix H.2, where $p = 5$.

## F. Problem 1: IEEE 14-Bus System

We also apply our methods on IEEE 14-bus system. This power system consists of a set of regions $\mathcal{R} = \{1, \cdots, 14\}$ with electricity demands $s_r^{\text{demand}}$, $r \in R$. Demands are satisfied by a set three power plans attached to node 2, node 8, and node 13 respectively. Each plant $n \in \mathcal{N}$ produces $x_n$ units of energy at costs $s_n^{\text{cost}}$.

We set $C_n = 3.6$ for all generators and $\bar{f}_m = 3$ for all transmission lines. We generate $N_{\text{train}} = 100$ data points for training and $N_{\text{test}} = 200$ data points for testing, by generating signals uniformly at random from $s_{13}^{\text{cost}} \in [0.2, 1]$, $s_2^{\text{cost}} \in [0.2, 0.5]$, $s_8^{\text{cost}} \in [1, 2]$, and $s_1^{\text{demand}} \in [0.14, 0.7]$, $s_2^{\text{demand}} \in [0.14, 0.7]$, $s_3^{\text{demand}} \in [0.16, 0.8]$, $s_4^{\text{demand}} \in [0.16, 0.8]$, $s_5^{\text{demand}} \in [0.14, 0.7]$, $s_6^{\text{demand}} \in [0.1, 0.5]$, $s_7^{\text{demand}} \in [0.16, 0.8]$, $s_8^{\text{demand}} \in [0.54, 2.7]$, $s_9^{\text{demand}} \in [0.1, 0.2]$, $s_{10}^{\text{demand}} \in [0.12, 0.6]$, $s_{11}^{\text{demand}} \in [0.12, 0.6]$, $s_{12}^{\text{demand}} \in [0.1, 0.5]$, $s_{13}^{\text{demand}} \in [0.1, 0.5]$, and $s_{14}^{\text{demand}} \in [0.12, 0.6]$, solving problem (14) to obtain pairs $\{\boldsymbol{s}_i, \boldsymbol{x}_i\}_{i=1}^{N=100}$.

We apply the same method introduced in Section 4.1 and summarize the results in Table 3.

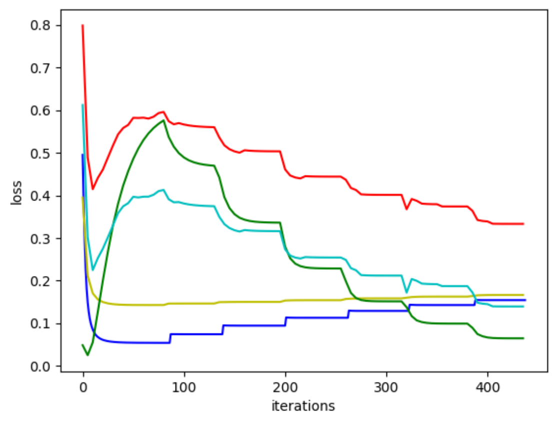

*Figure 4.* Training loss and test metrics. Blue: Training loss (predictability). Red: Out-of-sample predictability loss. Cyan: True predictability loss. Green: True suboptimality loss. Yellow: Out-of-sample suboptimality loss.

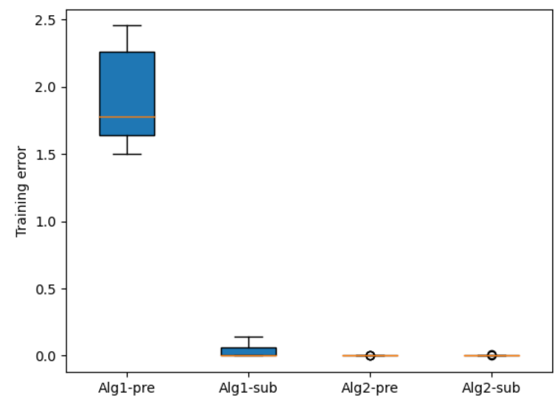

*Figure 5.* Comparison between convergence behaviors of predictability and suboptimality losses via both Algorithm 1 and 2. The y-axis represents the achieved training loss. Each box plot is based on the same 10 randomly generated datasets.

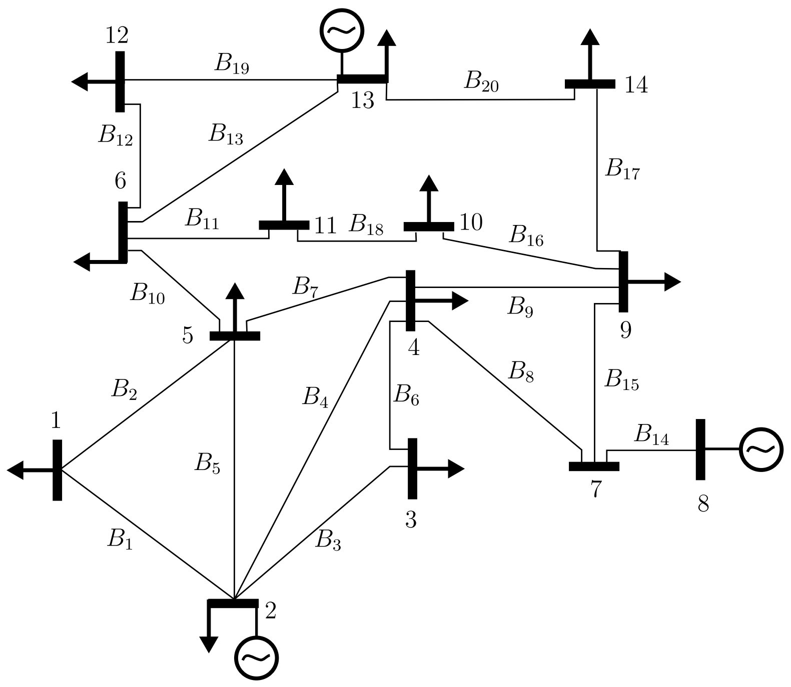

*Figure 6.* IEEE 14-bus system (Leon et al., 2020). Rectangles denote nodes, arrows denote loads, and circles denotes generators.

*Table 3.* Summary of out-of-sample performances on IEEE 14-bus system

| Hypothesis Class and Loss | | Evaluation method | | | |
|---|---|---|---|---|---|
| | loss | $\ell^p_\theta/\ell^p$ | $\ell^{\text{sub}}_\theta$ | $\ell^{\text{sub}}$ | time |
| Algo 2, $p = 1$ | predictability | 0.83 | 0.68 | 0.12 | 3.5s |
| Algo 2, $p = 2$ | predictability | 0.43 | 0.06 | 0.20 | 148.8s |
| Algo 2, $p = 3$ | predictability | 0.51 | 0.06 | 0.24 | 157.8s |
| quadratic cost | predictability | 2.58 | 2.56 | 0.63 | 2.54s |

## G. Reformulation of Problem 2

Recall that the primitive set is chosen as a box, i.e., $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^M | -\bar{f}_m \leq z_m \leq \bar{f}_m, \forall m \in M\}$, where $M = \{1, \cdots, |R| = 5\}$ ($|R|$ is the total number of the demand locations). For the hypothesis class, we use $\boldsymbol{A_\theta} \in \{0,1\}^{5\times 10}$ and $\boldsymbol{b_\theta} = \boldsymbol{\delta}$. Matrix $A_{\boldsymbol{\theta}}$ are binary matrices indicating the connections of power networks ($C_5^2 = 10$ power lines at most). We use $\boldsymbol{Gx} - \boldsymbol{g} \in \mathcal{G}$ to denote the capacity constraints of the power plants. For clarity, we denote the power generation plan and costs by two 5-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{c}$. Each dimension corresponds to one node in the graph. For nodes that have no power plant, we simply set its value to zero. Then, the forward problem can be written as:

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\
s.t. \quad & \boldsymbol{x} = \boldsymbol{A_\theta z} + \boldsymbol{s}^{\text{demand}}, \\
& \boldsymbol{Hz} - \boldsymbol{h} \in \mathcal{K}, \quad (\textbf{equivalently } \mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^M | -\bar{f}_m \leq z_m \leq \bar{f}_m, \forall m \in M\}) \\
& \boldsymbol{Gx} - \boldsymbol{g} \in \mathcal{G}, \quad (\textbf{equivalently } 0 \leq \boldsymbol{x} \leq 3.5).
\end{aligned} \tag{28}$$

Then, the predictability loss can be formulated as a binary problem as shown in (29).

$$\begin{aligned}
\min_{\boldsymbol{\gamma}_i, \boldsymbol{z}_i, \boldsymbol{A_\theta}, \boldsymbol{\beta}_i, \boldsymbol{\lambda}_i} \quad & \frac{1}{n} \sum_{i=1}^N \|\boldsymbol{\gamma}_i\|_2^2 \\
s.t. \quad & \boldsymbol{x}_i + \boldsymbol{\gamma}_i = \boldsymbol{A_\theta z}_i + \boldsymbol{s}_i^{\text{demand}}, \forall i, \\
& \boldsymbol{Hz}_i - \boldsymbol{h} \in \mathcal{K}, \forall i, \\
& \boldsymbol{G}(\boldsymbol{x}_i + \boldsymbol{\gamma}_i) - \boldsymbol{g} \in \mathcal{G}, \forall i, \\
& \boldsymbol{c}_i^\top(\boldsymbol{s}_i)(\boldsymbol{x}_i + \boldsymbol{\gamma}_i) - \boldsymbol{c}^\top(\boldsymbol{s}_i)\boldsymbol{s}_i^{\text{demand}} - \boldsymbol{\lambda}_i^\top \boldsymbol{h} + \boldsymbol{\beta}_i^\top \boldsymbol{g} - \boldsymbol{\beta}_i^\top \boldsymbol{G} \boldsymbol{s}_i^{\text{demand}} \leq 0, \forall i, \\
& \boldsymbol{\lambda}_i \in \mathcal{K}^*, \boldsymbol{\beta}_i \in \mathcal{G}^*, \forall i, \\
& \boldsymbol{c}_i^\top \boldsymbol{A_\theta} - \boldsymbol{\lambda}_i^\top \boldsymbol{H} - \boldsymbol{\beta}_i^\top \boldsymbol{G A_\theta} = 0, \forall i, \\
& \boldsymbol{A_\theta} \in \{0,1\}^{5\times 10}.
\end{aligned} \tag{29}$$

## H. Further Numerical Experiments

In this section, we conduct experiments on synthetic optimization problems to validate the proposed reformulations under both noiseless and noisy scenarios. We consider the following forward problem.

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^\top \boldsymbol{x}, \quad \text{s.t.} \quad \|\boldsymbol{x} - \boldsymbol{e}\|_1 \leq h. \tag{30}$$

We assume the objective coefficients $\boldsymbol{c}$ are input signals, which follow uniform distributions. The goal is to recover the feasible region based on the observed values of $\boldsymbol{c}$ and $\boldsymbol{x}$. Notice that choices of distance metrics of the objective functions for both predictability and suboptimality loss do not affect any of the reformulations or algorithms proposed. Throughout the subsequent experiments, square loss will be employed for the sake of simplicity.

### H.1. Convex Reformulations

In this part, we assume $c_i \sim U[-1,1]$, $1 \leq i \leq n = 2$, and we set tje primitive set: $\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^2 \big| \|\boldsymbol{z}\|_1 \leq 1\}$ in the hypothesis class. For the training data, we randomly generate $N$ cost coefficients and calculate the corresponding optimal solutions of the forward problem. Experiments under noise or no noise are both conducted. If the training set contains noise, we add a Gaussian noise $N(0, 0.2)$ to each of its dimensions. The performance is evaluated based on a test set that also contains 500 randomly generated coefficients and solutions.

When there is no noise, we achieved zero loss for all metrics. The performance summary for the noisy case is summarized in Table 4. It can be seen that the performance for predictability loss improves with the increasing number of training data under i.i.d. noises. This phenomenon arises due to the statistically consistent nature of estimates generated by predictability loss, as also highlighted in (Aswani et al., 2018).

*Table 4.* Out-of-sample Performance of Convex formulation with Gaussian noise.

| | loss | $\ell_\theta^p$ | $\ell^p$ | $\ell_\theta^{\text{sub}}$ | $\ell^{\text{sub}}$ |
|---|---|---|---|---|---|
| $N = 100$ | predictability | 0.54 | 0.03 | 0.29 | 0.05 |
| | suboptimality | 0.66 | 0.14 | 0.35 | 0.28 |
| $N = 500$ | predictability | 0.52 | $1.7e^{-3}$ | 0.31 | $3.3e^{-3}$ |
| | suboptimality | 0.61 | 0.09 | 0.34 | 0.17 |
| $N = 1000$ | predictability | 0.51 | 0 | 0.32 | 0 |
| | suboptimality | 0.59 | 0.07 | 0.34 | 0.13 |

## H.2. Bilinear Reformulations with Simplex Primitive Set

In this section, we showcase the performance of our bilinear reformulation, where the primitive set is defined in a high-dimensional space. More specifically, we assume , i.e, $c_i \sim U[0,1], 1 \leq i \leq n = 5$, and we consider the following simplex $\mathcal{Z} = \left\{ \boldsymbol{z} \geqslant \boldsymbol{0} \;\middle|\; \sum_{i=1}^{p} z_i = 1 \right\}$ in a $p$-dimensional space as the primitive set. For this bilinear reformulation, we can both apply Algorithm 2 or MILP reformulation to find the optimal solutions.

**Experiment setting:** We randomly generate $N$ cost coefficients and calculate the corresponding optimal solutions. We consider both predictability loss and suboptimality loss. We use backtracking to adjust the step length according to the Armijo rule.

We first apply Algorithm 2 to optimize the predictability and suboptimality loss. The optimization is stopped after 500 iterations. We record the performance of the proposed two losses under noiseless and noisy scenarios. For noisy cases, a Gaussian noise $N(0, 0.2)$ is added to each of the dimensions of the original solutions. We also vary the number of observed samples under noisy cases to check the changes in performances. In noiseless cases, we also apply MILP reformulation to obtain the optimal solutions.

*Table 5.* Noiseless

| n=5 | N=100 | train | $\ell_\theta^p$ | $\ell^p$ | $\ell_\theta^{\text{sub}}$ | $\ell^{\text{sub}}$ | time |
|---|---|---|---|---|---|---|---|
| $p = 4$ | Predictability | 0.32 | 1.04 | 1.04 | 0.44 | 0.10 | 174.6s |
| | Suboptimality | 0.33 | 1.04 | 1.04 | 0.44 | 0.10 | 48.3s |
| | MILP | 0.25 | 0.99 | 0.99 | 0.45 | 0.07 | 901.2s |
| $p = 5$ | Predictability | 0 | 0 | 0 | 0 | 0 | 49s |
| | Suboptimality | 0 | 0 | 0 | 0 | 0 | 52.3s |
| | MILP | 0 | 0 | 0 | 0 | 0 | 902.7s |
| $p = 6$ | Predictability | 0 | 0 | 0 | 0 | 0 | 21.6s |
| | Suboptimality | 0 | 0 | 0 | 0 | 0 | 42.3s |
| | MILP | 0 | 0 | 0 | 0 | 0 | 543.5s |

*Table 6.* Noisy case with 100 training samples.

| $n = 5$ | $N = 100$ | train | $\ell_\theta^p$ | $\ell^p$ | $\ell_\theta^{\text{sub}}$ | $\ell^{\text{sub}}$ | time |
|---|---|---|---|---|---|---|---|
| $p = 4$ | Predictability | 0.52 | 1.17 | 0.97 | 0.60 | 0.10 | 87.4s |
| | Suboptimality | 0.43 | 1.42 | 1.22 | 0.57 | 0.91 | 85.0s |
| $p = 5$ | Predictability | 0.14 | 0.42 | 0.24 | 0.18 | 0.11 | 110.9s |
| | Suboptimality | 0.11 | 0.68 | 0.49 | 0.14 | 0.84 | 101.2s |
| $p = 6$ | Predictability | 0.13 | 0.50 | 0.31 | 0.19 | 0.07 | 131.0s |
| | Suboptimality | 0.11 | 0.65 | 0.46 | 0.14 | 0.72 | 121.9s |

*Table 7.* Noisy case with 200 training samples.

| $n = 5$ | $N = 200$ | train | $\ell_\theta^p$ | $\ell^p$ | $\ell_\theta^{\text{sub}}$ | $\ell^{\text{sub}}$ | time |
|---|---|---|---|---|---|---|---|
| $p = 4$ | Predictability | 0.52 | 1.17 | 0.97 | 0.58 | 0.10 | 170.4s |
| | Suboptimality | 0.44 | 1.46 | 1.26 | 0.56 | 0.97 | 208.3$s$ |
| $p = 5$ | Predictability | 0.16 | 0.37 | 0.18 | 0.17 | 0.05 | 259.2s |
| | Suboptimality | 0.11 | 0.78 | 0.58 | 0.14 | 0.89 | 222.4s |
| $p = 6$ | Predictability | 0.14 | 0.37 | 0.19 | 0.16 | 0.10 | 257.2s |
| | Suboptimality | 0.11 | 0.74 | 0.54 | 0.13 | 0.84 | 236.2s |

**Results.** The results are summarized in Table 5, 6, and 7. In scenarios without noise, the forward problem is accurately restored when the hypothesis class encompasses the true feasible region. In such instances, this requirement is equivalent to having $p \geq 5$ since the original feasible region comprises five extreme points. In noisy scenarios, we noticed enhanced performance with an enlarged sample size for predictability loss, with no corresponding improvement observed for suboptimality loss. It was also noted that the MILP reformulation exhibited superior overall performance in contrast to the gradient-descent-based algorithm. Nonetheless, in instances characterized by noise, the inherent complexity of MILP formulations renders them challenging (number of binary variables proportional to data size) to resolve within a one-hour timeframe. Finally, a visual representation of feasible region $g_{\theta^t}(\boldsymbol{x}, \boldsymbol{s}) \leq 0$ as the parameters $\theta^t$ are updated using Algorithm 2 is presented in Figure 7 for the noiseless case. We observe that the hypothesis class is able to rotate, scale, project and translate the primitive set by updating $\theta^t = (A^t, b^t)$.

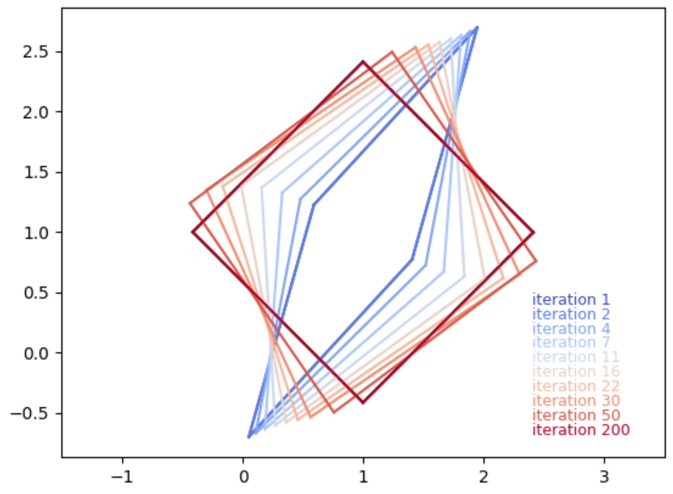

*Figure 7.* Visualization of $g_{\theta^t}(\boldsymbol{x}, \boldsymbol{s}) \leq 0$ for Algorithm 2. The feasible region of the forward problem is given by $\|\boldsymbol{x} - \boldsymbol{e}\| \leq 1$, and the learning algorithm recovers it by iteration 200.