Learning in Inverse Optimization

Incenter Cost, Augmented Suboptimality Loss, and Algorithms

Scroccaro, Pedro Zattoni; Atasoy, Bilge; Esfahani, Peyman Mohajerin

DOI
10.1287/opre.2023.0254

Publication date
2025

Document Version
Final published version

Published in
Operations Research

Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Learning in Inverse Optimization: Incenter Cost, Augmented Suboptimality Loss, and Algorithms

Pedro Zattoni Scroccaro, Bilge Atasoy, Peyman Mohajerin Esfahani

To cite this article:
Pedro Zattoni Scroccaro, Bilge Atasoy, Peyman Mohajerin Esfahani (2025) Learning in Inverse Optimization: Incenter Cost, Augmented Suboptimality Loss, and Algorithms. Operations Research 73(5):2661-2679. https://doi.org/10.1287/opre.2023.0254

https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 73, No. 5, September–October 2025, pp. 2661–2679
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Learning in Inverse Optimization: Incenter Cost, Augmented Suboptimality Loss, and Algorithms

Pedro Zattoni Scroccaro,[a,*] Bilge Atasoy,[b] Peyman Mohajerin Esfahani[a]

[a] Delft Center of Systems and Control, Delft University of Technology, 2628 CD Delft, Netherlands; [b] Department of Maritime and Transport Technology, Delft University of Technology, 2628 CD Delft, Netherlands

*Corresponding author

Contact: P.ZattoniScroccaro@tudelft.nl, https://orcid.org/0000-0002-6752-6328 (PZS); b.atasoy@tudelft.nl, https://orcid.org/0000-0002-1606-9841 (BA); P.MohajerinEsfahani@tudelft.nl, https://orcid.org/0000-0003-1286-8782 (PME)

Received: May 12, 2023
Revised: January 19, 2024; June 18, 2024
Accepted: July 7, 2024
Published Online in Articles in Advance: September 30, 2024

Area of Review: Optimization

https://doi.org/10.1287/opre.2023.0254

Abstract. In inverse optimization (IO), an expert agent solves an optimization problem parametric in an exogenous signal. From a learning perspective, the goal is to learn the expert's cost function given a data set of signals and corresponding optimal actions. Motivated by the geometry of the IO set of consistent cost vectors, we introduce the "incenter" concept, a new notion akin to the recently proposed circumcenter concept. Discussing the geometric and robustness interpretation of the incenter cost vector, we develop corresponding tractable convex reformulations that are in contrast with the circumcenter, which we show is equivalent to an intractable optimization program. We further propose a novel loss function called *augmented suboptimality loss* (ASL), a relaxation of the incenter concept for problems with inconsistent data. Exploiting the structure of the ASL, we propose a novel first-order algorithm, which we name *stochastic approximate mirror descent*. This algorithm combines stochastic and approximate subgradient evaluations, together with mirror descent update steps, which are provably efficient for the IO problems with discrete feasible sets with high cardinality. We implement the IO approaches developed in this paper as a Python package called InvOpt. Our numerical experiments are reproducible, and the underlying source code is available as examples in the InvOpt package.

Funding: This work was partially supported by the European Research Council [Grant TRUST-949796].
Supplementary Review: The empirical results in this paper were replicated. The code, data, and files required to reproduce the results were reviewed and are available at https://doi.org/10.1287/opre.2023.0254.cd.
Supplemental Material: This article includes an online appendix, computer code and data supporting the study's findings, and replication files. All supplemental materials, including the code, data, and files required to reproduce the results, are available at https://doi.org/10.1287/opre.2023.0254.cd.

Keywords: inverse optimization • convex optimization • first-order algorithms

1. Introduction

In inverse optimization (IO) problems, our goal is to model the behavior of an expert agent, which given an exogenous signal, returns a response action. The underlying assumption of IO is that to compute its response, the expert agent solves an optimization problem parametric in the exogenous signal. We assume to know the constraints imposed on the expert but not its cost function. Therefore, our goal is to model the cost function being optimized by the expert using examples of exogenous signals and corresponding expert response actions. For example, consider the problem of learning how to route vehicles using examples from experienced drivers. Given a set of customer demands (exogenous signal), the experienced driver chooses a certain vehicle route to serve the customers (expert response). Assuming the drivers solve some kind of route optimization problem to compute their routes, where their cost function depends on the drivers' evaluation of how costly it is to drive from one customer to another, one could look at this problem as an IO problem. Thus, one could use IO tools to learn the cost function being used by these drivers (i.e., we learn the preferences of the expert drivers when driving from a certain customer to another). IO tools have also been used in many other application areas: for instance, healthcare problems (Chan et al. 2014, 2022), modeling of consumer behavior (Bertsimas et al. 2015), transportation problems (Burton and Toint 1992, Chow and Recker 2012, Zattoni Scroccaro et al. 2024), portfolio selection (Bertsimas et al. 2012), network design

(Faragó et al. 2003), forecast of electricity prices (Saez-Gallego and Morales 2017), and control systems (Akhtar et al. 2021). For more examples of applications of IO, we refer the reader to the recent review paper by Chan et al. (2023) and references therein.

The literature on IO can be roughly divided into so-called *classical* and *data-driven* IO. In classical IO, the goal is usually to find a cost function that renders a *single* signal-response pair optimal: that is, a cost function under which the observed response is optimal. A direct approach to modeling this scenario leads to a bilevel optimization problem; thus, much of the early IO literature focuses on reformulating this problem into a single-level tractable program (Ahuja and Orlin 2001). This idea has been extended to different classes of optimization problems, such as conic and integer programs (Heuberger 2004, Ahmed and Guan 2005, Iyengar and Kang 2005, Schaefer 2009, Wang 2009). On the other hand, data-driven IO usually deals with problems with multiple signal-response examples, and it is not necessarily assumed that there exists a cost function consistent with all signal-response data. In this scenario, the IO problem can be viewed as a supervised learning problem with multivariate output where the IO model forms a hypothesis class. In this view, one minimizes a training loss function to find a good" fit" to the input (response) and output (optimizer) data. With this in mind, much of the data-driven IO literature focuses on the choice of the training loss function, particularly because the usual supervised learning losses lead to nonconvex programs (Aswani et al. 2018). Examples of such losses are the *Karush-Kuhn-Tucker (KKT) loss* (Keshavarz et al. 2011), the *first-order loss* (Bertsimas et al. 2015), the *predictability loss* (Aswani et al. 2018), and the *suboptimality loss* (SL) (Mohajerin Esfahani et al. 2018). It is worth noting that in the standard IO setting, directly learning the cost function (i.e., regression with the signal pair as the input and the cost function value as the scaler output) is typically not an option because the available data contain only the signal-response pair and not the value of the unknown cost function. Recently, in a framework called "predict, then optimize," Elmachtoub and Grigas (2022) consider this additional information and propose another loss function coined as smart "predict, then optimize" loss (SPO). Other concepts have also been investigated in the context of IO problems, such as goodness of fit (Chan et al. 2019), robustness against misspecification (Ghobadi et al. 2018), and learning constraints instead of cost functions (Ghobadi and Mahmoudzadeh 2021).

Recently, Besbes et al. (2023) showed that given a data set of expert examples and the set $\mathbb{C}$, which is defined as the set of cost vectors consistent with the data set, the optimal solution to the IO problem, for a certain regret performance measure, is the so-called *circumcenter* of the set $\mathbb{C}$. To the best of our knowledge, this is the first minimax regret result for IO problems. To derive this result, the authors exploit the geometry of IO problems and provide strong regret guarantees. The general optimization program associated with computing the circumcenter vector, however, turns out to contain intractable problem instances. Moreover, when the IO data set is inconsistent with a single cost function, it is not clear how one could generalize the circumcenter concept to solve the IO problem. These shortcomings motivate us to propose a new concept called "incenter" to select an IO cost vector from the set of consistent costs. In light of this, the main contributions of this work are summarized as follows.

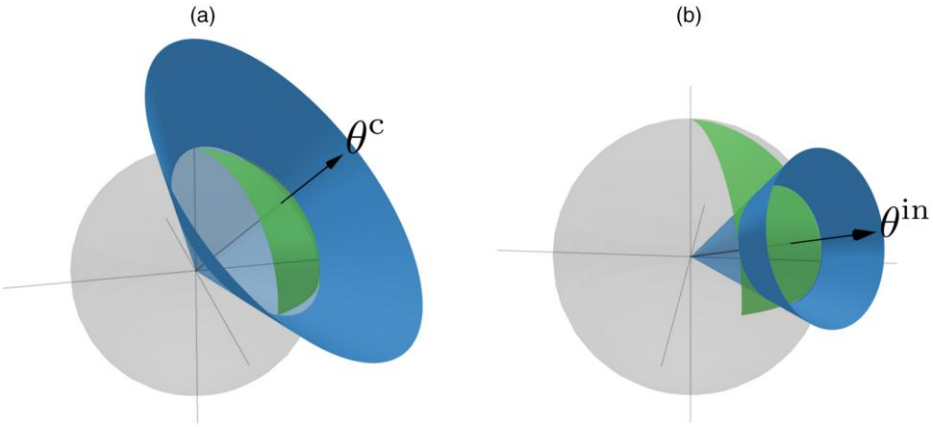
(i) **Geometric and robustness interpretations.** Motivated by the geometry of the set of consistent cost vectors, we introduce the concept of an incenter (Definition 3.2) and provide insights into its geometry (Figure 1) and robustness (Remark 3.1) in comparison with the *circumcenter* concept from Besbes et al. (2023).

(ii) **Convex reformulations and tractability.** We develop tractable convex reformulations of the incenter (Theorem 3.2 and Corollary 3.1) along with a geometric interpretation of these characterizations (Figure 2). This may be of particular interest as the corresponding circumcenter concept is equivalent to an intractable optimization program (Theorem 3.1). To establish this intractability result, we draw connections between the circumcenter/incenter concepts and the well-known problem of extremal volume balls (Remark 3.2). Moreover, because the problem of extremal volume balls is a special case of the problem of extremal volume ellipsoids, this connection allows us to derive "ellipsoidal" versions of the incenter and circumcenter concepts (e.g., Equation (12)), which perform well in our numerical experiments (Section 6.1).

(iii) **Generalization to inconsistent data and augmented suboptimality loss (ASL).** Generalizing to the inconsistent data setting, we propose a novel loss function for IO problems, which we name ASL (Definition 4.1). This loss can be interpreted as a *relaxation* of the incenter concept to handle IO problems with inconsistent data (Equation 16). In special cases, this formulation of the IO problem can be shown to be equivalent to the so-called structured Support Vector Machine (SVM) formulation of structured prediction problems, revealing a connection between IO and structure prediction (Remark 4.2). We further propose a general convex reformulation of the ASL for IO problems with mixed-integer feasible sets (Theorem 4.1 and Corollary 4.1), which generalizes several reformulations from the literature (Remark 4.4).

(iv) **Tailored first-order algorithm: stochastic approximate mirror decent (SAMD).** Motivated by the structure of IO loss functions, we propose a novel first-order algorithm, which we name SAMD. This algorithm exploits the finite sum structure of the IO problem to compute stochastic gradients, uses approximate

Figure 1. (Color online) Geometrical Visualization of the Circumcenter and Incenter Vectors

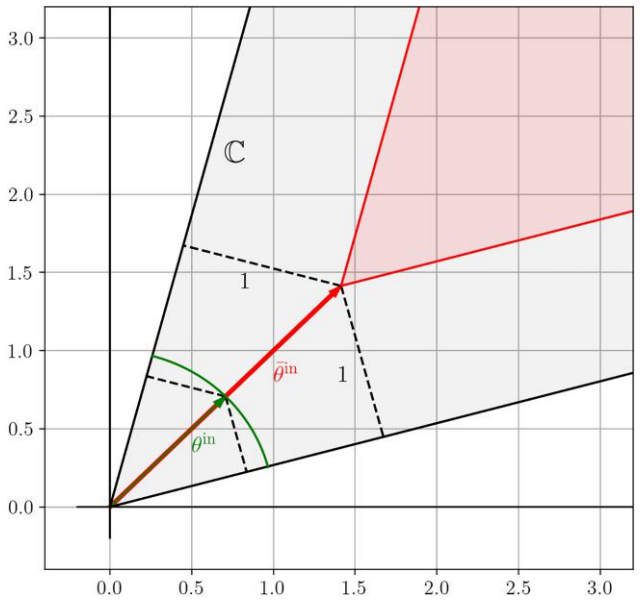


Notes. The green regions are the intersection of $\mathbb{C}$ with the sphere, and the blue cones are revolution cones with an aperture angle equal to the optimal value of (7) and (8). (a) Circumcenter θ^{c} defined in (7). (b) Incenter θ^{in} defined in (8).

evaluating IO loss functions to compute approximate subgradients, and uses mirror descent update steps for problems with favorable geometry (Sections 5.1–5.3). We prove convergence rates for this algorithm (Proposition 5.1) and show how the components of this algorithm can be tailored to exploit the structure of IO loss functions (Algorithm 5.1).

(v) **Inverse optimization Python package.** We implement the IO approaches developed in this paper as a Python package called InvOpt (Zattoni Scroccaro 2023). This is a Python package developed to solve general IO problems. Our numerical experiments are reproducible, and the underlying source code is available as examples in the InvOpt package.

Figure 2. (Color online) Geometrical Illustration of Corollary 3.1 in a Simple Two-Dimensional Example



Notes. The gray region represents $\mathbb{C}$, the green region represents the feasible set of (10), and the red region represents the feasible set of (11). As can be seen, the optimal solution of (11) can be interpreted as the smallest norm vector inside an inner cone with boundaries one unit away from the boundaries of $\mathbb{C}$. Also, it can be seen that by normalizing $\bar{\theta}^{in}$, we retrieve θ^{in}, an optimal solution of (10).

The rest of the paper is organized as follows. In Section 2, we formalize the IO problem. In Section 3, we present approaches based on the incenter concept. In Section 4, we define the augmented suboptimality loss and show how it can be used to tackle IO problems with inconsistent data and mixed-integer feasible sets. In Section 5, we present first-order algorithms tailored to IO problems. In Section 6, we report our numerical results. In Section 7, we conclude the paper and discuss future research directions. In Online Appendix EC.1, we discuss the theoretical properties of the ASL. In Online Appendix EC.2, we present concise reformulations for some special IO problems with continuous feasible sets. In Online Appendix EC.3, we present proofs, and in Online Appendix EC.4, we present further numerical results.

Notation. The trace, range space, and Moore–Penrose inverse of a matrix $Q \in \mathbb{R}^{m\times m}$ are denoted as $\mathrm{Tr}(Q)$, $\mathcal{R}(Q)$, and $Q^{\dagger}$, respectively. For two symmetric matrices $Q, R \in \mathbb{R}^{m\times m}$, $Q \succcurlyeq R$ means that $Q - R$ is positive semidefinite. The Euclidean inner product between two vectors $x, y \in \mathbb{R}^{m}$ is denoted by $\langle x, y\rangle$. The set of vectors with nonnegative components is denoted as $\mathbb{R}^{n}_{+} := \{x \in \mathbb{R}^{n} : x \geq 0\}$. The cardinality, complement, interior, and convex hull of a set A are denoted as $|A|$, $\overline{A}$, $\mathrm{int}(A)$, and $\mathrm{conv}(A)$, respectively. The set of integers from one to N is denoted as $[N]$. A set of indexed values is compactly denoted by $\{x_i\}_{i=1}^{N} := \{x_1, \ldots, x_N\}$. The Euclidean projection of x onto a set A is defined as $\Pi_A(x) := \arg\min_{y\in A} \|y - x\|_2$. The Euclidean angle between two nonzero vectors θ and $\tilde{\theta}$ is defined as $a(\theta, \tilde{\theta}) := \arccos\langle \theta, \tilde{\theta}\rangle / (\|\theta\|_2 \|\tilde{\theta}\|_2)$. For vectors $x, y \in \mathbb{R}^{m}$, $x \odot y$ and $\exp(x)$

mean element-wise multiplication and element-wise exponentiation, respectively. Moreover, whenever the arguments of $\exp(\cdot)$ or $\log(\cdot)$ are matrices, they should be interpreted as the usual matrix exponentiation and matrix logarithm, respectively. The vector $\mathrm{vec}(Q) \in \mathbb{R}^{pq}$ denotes the vectorization of the matrix $Q \in \mathbb{R}^{p\times q}$ formed by stacking the columns of Q into a single-column vector. The symbol $\otimes$ denotes the Kronecker product. The vector $e_j \in \mathbb{R}^n$ denotes the vector of zeros except for the jth element, which is equal to one. The vector of ones is represented by one.

2. Problem Description

Let us begin by formalizing the IO problem. Let $s \in \mathbb{S}$ be an exogenous signal and $\mathbb{S}$ be the signal space. The expert agent is assumed to solve the parametric *forward optimization problem*

$$\min_{x\in\mathbb{X}(s)} F(s,x), \tag{1}$$

where $\mathbb{X}(s)$ is the feasible set and $F : \mathbb{S}\times\mathbb{X} \to \mathbb{R}$ is the expert's cost function, where we define $\mathbb{X} := \cup_{s\in\mathbb{S}}\mathbb{X}(s)$. Recall that the cost F is unknown to us, and we only have access to a data set of N pairs of exogenous signals and respective expert optimal decisions $\hat{\mathcal{D}} := \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$: that is,

$$\hat{x}_i \in \arg\min_{x\in\mathbb{X}(\hat{s}_i)} F(\hat{s}_i, x), \quad \forall i \in [N].$$

We use the notation "$\hat{}$" to indicate signal-response data. Using these data, our goal is to learn a cost function that when minimized, reproduces the expert's decisions as well as possible.

Naturally, we can only search for functions in a restricted function space. In this work, we consider the following parametric *hypothesis space*:

$$\mathcal{F}_\phi := \{\langle\theta, \phi(\cdot,\cdot)\rangle : \theta \in \mathbb{R}^p\}, \tag{2}$$

where $\theta \in \mathbb{R}^p$ is called the *cost vector* and $\phi : \mathbb{S}\times\mathbb{X} \to \mathbb{R}^p$ is called *feature mapping*, which maps a signal-response pair (s, x) to a feature vector $\phi(s,x) \in \mathbb{R}^p$. In practice, choosing a suitable mapping ϕ is part of the modeling of the IO problem. To exemplify the generality of this hypothesis space, the next example shows how to model a quadratic function as a member of $\mathcal{F}_\phi$. Other utility functions from the literature (for instance, the Constant Elasticity of Substitution function and the Cobb–Douglas function) can also be easily fitted into our framework (Chen and Kılınç-Karzan 2020, appendix A).

Example 2.1 (Quadratic Hypothesis). Consider a quadratic function

$$F(s,x) = \langle x, Q_{xx}x\rangle + \langle x, Q_{xs}s\rangle + \langle x, q\rangle,$$

parametrized by $(Q_{xx}, Q_{xs}, q) \in \mathbb{R}^{n\times n}\times\mathbb{R}^{n\times m}\times\mathbb{R}^n$. Using the identity $\mathrm{vec}(AXB) = \langle \mathrm{vec}(X), B\otimes A^\top\rangle$, we can rewrite this quadratic function as

$$\begin{aligned} F(s,x) &= \langle \mathrm{vec}(Q_{xx}), x\otimes x\rangle + \langle \mathrm{vec}(Q_{xs}), s\otimes x\rangle + \langle x, q\rangle \\ &= \langle \theta, \phi(s,x)\rangle, \end{aligned}$$

where

$$\theta := \begin{bmatrix} \mathrm{vec}(Q_{xx}) \\ \mathrm{vec}(Q_{xs}) \\ q \end{bmatrix} \quad \text{and} \quad \phi(s,x) := \begin{bmatrix} x\otimes x \\ s\otimes x \\ x \end{bmatrix}.$$

Thus, our IO goal is to find a cost vector θ such that when solving the optimization problem

$$\min_{x\in\mathbb{X}(s)} \langle\theta, \phi(s,x)\rangle, \tag{3}$$

we can reproduce (or in some sense, approximate) the action that the expert would have taken given the same signal s. An important object in IO problems is $\mathbb{C}$, the set of cost vectors consistent with the data $\hat{\mathcal{D}}$.

Definition 2.1 (Consistent Cost Vectors). Given a data set $\hat{\mathcal{D}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$, feature mapping ϕ, and feasible set $\mathbb{X}$, the *set of consistent cost vectors* is defined as

$$\mathbb{C} := \left\{ \theta \in \mathbb{R}^p : \begin{array}{c} \langle\theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i)\rangle \le 0, \\ \forall x_i \in \mathbb{X}(\hat{s}_i),\ \forall i \in [N] \end{array} \right\}. \tag{4}$$

In other words, $\mathbb{C}$ corresponds to the set of cost vectors θ that render the expert responses $\hat{x}_i$'s optimal (i.e., $\hat{x}_i \in \arg\min_{x_i\in\mathbb{X}(\hat{s}_i)}\langle\theta, \phi(\hat{s}_i, x_i)\rangle$) for all $i \in [N]$. Geometrically, the set $\mathbb{C}$ is a cone, defined by the intersection of half-spaces defined by hyperplanes that pass through the origin.

3. Incenter Cost Vector

From Definition 2.1, one can observe that a trivial element in the set $\mathbb{C}$ is $\theta = 0$. However, this trivial choice is not really of interest because it yields any response $x \in \mathbb{X}(s)$ to the signal s an optimal solution; hence, there is no differentiation between different responses. Therefore, when $\mathbb{C}$ contains other elements than $\theta = 0$ (that is, the data are consistent with the hypothesis space in the sense that the expert's true cost is $F(s,x) = \langle\theta^\star, \phi(s,x)\rangle$ for some nonzero element $\theta^\star$), it is natural to restrict the search space to $\mathbb{C}\setminus\{0\}$. It is also worth noting that a similar issue concerns the feature map ϕ. Namely, from a modeling perspective, it is important to consider a feature class where $\phi(s,x)$ is not identically zero in the second argument over the set $\mathbb{X}(s)$ for typical signal variables $s \in \mathbb{S}$. This assumption, together with excluding the trivial cost vector $\theta = 0$ from the search space, is necessary. Thus, in this section, we assume $\mathbb{C}\setminus\{0\} \neq \emptyset$ and that ϕ is not trivially zero, and we discuss different ways to choose a cost vector from the set $\mathbb{C}$. A straightforward way to do this is by solving a feasibility optimization problem, which we call the

feasibility program:

$$\begin{aligned} \min_{\theta} \quad & 0 \\ \text{s.t.} \quad & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i) \rangle \leq 0 \\ & \qquad \forall x_i \in \mathbb{X}(\hat{s}_i), \ \ \forall i \in [N] \\ & \|\theta\| = 1. \end{aligned} \tag{5}$$

Problem (5) simply translates the idea that we wish to find some nonzero $\theta \in \mathbb{C}$. Notice that the normalization constraint $\|\theta\| = 1$ excludes the trivial solution $\theta = 0$, and it is nonrestrictive in the sense that $\arg\min_{x \in \mathbb{X}(s)} \langle \theta, \phi(s,x) \rangle = \arg\min_{x \in \mathbb{X}(s)} \langle \alpha\theta, \phi(s,x) \rangle$, for any $\alpha > 0$. By solving the IO problem using the feasibility program (5), we implicitly suggest that any cost vector $\theta \in \mathbb{C} \setminus \{0\}$ is an equally good solution to the IO problem. However, one could argue that there are more principled ways to choose a cost vector from the set of consistent vectors $\mathbb{C}$.

3.1. Geometry, Robustness, and Tractability

Considering two cost vectors θ and $\theta^\star$ with their corresponding responses defined as

$$\begin{aligned} & x^\theta \in \arg\min_{x \in \mathbb{X}(\hat{s})} \langle \theta, \phi(\hat{s}, x) \rangle \quad \text{and} \\ & x^\star \in \arg\min_{x \in \mathbb{X}(\hat{s})} \langle \theta^\star, \phi(\hat{s}, x) \rangle, \end{aligned} \tag{6a}$$

Besbes et al. (2023) define the regret

$$R(\theta, \theta^\star) := \langle \theta^\star, \phi(\hat{s}, x^\theta) - \phi(\hat{s}, x^\star) \rangle. \tag{6b}$$

The regret (6b) is a distance function between two cost vectors that measures the difference via the impact of their respective optimizers x^θ and $x^\star$ evaluated through the second argument, here the ground truth $\theta^\star$. As shown in Besbes et al. (2023), the worst-case optimal cost vector θ for the regret $R(\theta, \theta^\star)$ is the so-called *circumcenter* of $\mathbb{C}$ as defined in the following.

Definition 3.1 (Circumcenter (Besbes et al. 2023)). Let $\mathbb{C}$ be a nonempty set. A *circumcenter* of $\mathbb{C}$ is defined as

$$\theta^c \in \arg\min_{\theta \neq 0} \max_{\substack{\tilde{\theta} \in \mathbb{C} \\ \tilde{\theta} \neq 0}} a(\theta, \tilde{\theta}). \tag{7}$$

In the original definition of Besbes et al. (2023), the nonzero constraints over the cost vectors are enforced via $\|\theta\|_2 = \|\tilde{\theta}\|_2 = 1$, which does not change the circumcenter because the angle in the objective is scale invariant (recall that the Euclidean angle between two nonzero vectors θ and $\tilde{\theta}$ is defined as $a(\theta, \tilde{\theta}) = \arccos\langle \theta, \tilde{\theta} \rangle / (\|\theta\|_2 \|\tilde{\theta}\|_2)$). Geometrically, θ^c can be interpreted as a point in the axis of the revolution cone with the smallest aperture angle containing $\mathbb{C}$. Informally speaking, the circumcenter approach chooses a cost vector by looking for the vector in $\mathbb{C}$ farthest away (in terms of the angle) from the "corners" of $\mathbb{C}$ (more precisely, from the extreme rays of $\mathbb{C}$). However, in this study, we identify the following three possible shortcomings when using the circumcenter concept to choose an IO model.

(i) **Computational tractability.** Computing the circumcenter, in general, turns out to be computationally intractable.

(ii) **Robustness interpretation.** Although the circumcenter is, by definition, robust to an adversarial (i.e., worst-case) ground truth data-generating model, the challenge in practice is often related to noisy data.

(iii) **Inconsistent data.** The circumcenter is only defined for problems with consistent data (i.e., nonempty $\mathbb{C}$) and to the best of our knowledge, cannot be straightforwardly extended to IO problems with inconsistent data.

In the rest of this section, we first formalize the tractability result (i), and then, we introduce a new notion that addresses these possible shortcomings related to circumcenter.

Theorem 3.1 (Intractability of Circumcenter). *Assume $\mathbb{C}$ is a nonempty polyhedral cone. Then, solving the inner maximization of (7) for any θ is equivalent to maximizing a quadratic function over a polytope, which is NP hard.*

Definition 3.2 (Incenter). Let $\mathbb{C}$ be a nonempty set. An *incenter* of $\mathbb{C}$ is defined as

$$\theta^{\text{in}} \in \arg\max_{\theta \neq 0} \min_{\substack{\tilde{\theta} \in \text{int}(\mathbb{C}) \\ \tilde{\theta} \neq 0}} a(\theta, \tilde{\theta}). \tag{8}$$

We will show how the incenter Problem (8) can be reformulated as a tractable convex optimization problem (as opposed to the NP hardness of the circumcenter), and by relaxing the reformulation, it can be straightforwardly extended to handle problems with inconsistent data. Let us first elaborate more on the intuition behind the incenter vector in Definition 3.2 compared with the circumcenter vector in Definition 3.1. Geometrically, an incenter of $\mathbb{C}$ can be interpreted as the vector farthest away (in terms of the angle) from the *exterior* of $\mathbb{C}$ or in other words, the vector in $\mathbb{C}$ farthest away from the boundary of $\mathbb{C}$. To provide insight into these different centers, we visualize them in a simple three-dimensional space in Figure 1. Interestingly, these notions also have robustness interpretations, which are different from existing learning models in which the robustness is introduced explicitly via regularization or distributional robust approaches (Mohajerin Esfahani et al. 2018).

Remark 3.1 (Robustness: Circumcenter Versus Incenter). A circumcenter θ^c in (7) can be interpreted as the vector most robust to an adversary with the authority to choose the true data-generating cost vector in $\mathbb{C}$ to be farthest away from our learned model, which is a notion captured by the regret performance measure introduced in Besbes et al. (2023). On the other hand,

the incenter θ^{in} (8) can be viewed as the vector farthest away from the boundary of $\mathbb{C}$. Because each facet of $\mathbb{C}$ is determined by a signal-response $(\hat{s}_i, \hat{x}_i)$ pair in the data set of the IO problem, the incenter vector can be interpreted as the cost vector most robust to perturbations of the training data set (i.e., perturbations of the facets of $\mathbb{C}$).

Let us provide some additional details supporting the robustness interpretation in Remark 3.1. To formalize this robustness notion, recall the definition of the set of consistent cost vectors $\mathbb{C} = \{\theta \in \mathbb{R}^p : \langle \theta, \Delta(\hat{s}_i, \hat{x}_i, x)\rangle \le 0,\ \forall x \in \mathbb{X}(\hat{s}_i),\ \forall i \in [N]\}$, where we define $\Delta(\hat{s}_i, \hat{x}_i, x) := (\phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i))/\|\phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i)\|_2$ if $\phi(\hat{s}_i, \hat{x}_i) \neq \phi(\hat{s}_i, x_i)$ and $\Delta(\hat{s}_i, \hat{x}_i, x) := 0$ otherwise. Differently from the definition in Equation (4), here we simply normalize the vectors $\phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i)$, which does not change the set $\mathbb{C}$. Notice that the vectors $\Delta(\hat{s}_i, \hat{x}_i, x)$ are defined by the data $\{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ and in turn, define the set $\mathbb{C}$. Thus, the vector $\theta \in \mathbb{C}$ most robust to perturbations in the data (i.e., perturbation of $\Delta(\hat{s}_i, \hat{x}_i, x)$) can be found by solving the minimax problem

$$\max_{\substack{\|\theta\|_2=1 \\ \theta\in\mathbb{C}}} \ \min_{\substack{w\in\mathbb{R}^p \\ (\hat{s}_k,\hat{x}_k)\in\{(\hat{s}_i,\hat{x}_i)\}_{i=1}^N \\ x\in\mathbb{X}(\hat{s}_k)}} \ \|w\|_2^2$$
$$\text{s.t.} \quad \langle \theta, \Delta(\hat{s}_k, \hat{x}_k, x) + w\rangle = 0. \tag{9}$$

In Problem (9), the "max player" optimizes for the vector $\theta \in \mathbb{C}$ that requires the largest perturbation w so that it lies in the hyperplane $\langle \theta, \Delta(\hat{s}_k, \hat{x}_k, x) + w\rangle = 0$ (i.e., a perturbed facet of $\mathbb{C}$). The "min player" tries to find the "most vulnerable" $\Delta(\hat{s}_k, \hat{x}_k, x)$: that is, the facet of $\mathbb{C}$ that requires the smallest perturbation vector w to make $\langle \theta, \Delta(\hat{s}_k, \hat{x}_k, x) + w\rangle = 0$. It can be shown that the optimal perturbation is $w = -(\langle \theta, \Delta(\hat{s}_k, \hat{x}_k, x)\rangle / \|\theta\|_2^2)\theta$, and substituting it in the objective function of (9), one can show that the optimization Problem (9) is equivalent to the incenter reformulation of Theorem 3.2.

As a final comment, even ignoring the computational cost of computing the circumcenter vector versus computing the incenter vector, for cases when the set of consistent vectors $\mathbb{C}$ is not empty, the incenter approach may perform better in practice (e.g., see the results in Section 6.1). In light of the discussion in Remark 3.1, this suggests that approaches robust to perturbations in the data (i.e., incenter) might be preferred to the one that is robust to an adversarial true data-generating model (i.e., circumcenter).

3.2. Reformulations

Next, we present a reformulation of (8). This reformulation will be used to derive tractable incenter-based approaches to IO problems, and we show that if $\text{int}(\mathbb{C}) \neq \emptyset$, one can compute the incenter vector by solving an optimization problem without a norm equality constraint.

Theorem 3.2 (Incenter Reformulation). *Problem* (8) *can be reformulated as*

$$\begin{aligned}(\theta^{\text{in}}, r^{\text{in}}) \in \arg\max_{\theta, r}\ & r \\ \text{s.t. } & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i)\rangle \\ & + r\|\phi(\hat{s}_i, x_i) - \phi(\hat{s}_i, \hat{x}_i)\|_2 \le 0 \\ & \qquad \forall x_i \in \mathbb{X}(\hat{s}_i),\ \forall i \in [N] \\ & \|\theta\|_2 = 1.\end{aligned} \tag{10}$$

Corollary 3.1 (Incenter Convex Characterization). *Assume* $\text{int}(\mathbb{C}) \neq \emptyset$. *Then, Problem* (10) *can be solved as*

$$\begin{aligned}\overline{\theta}^{\text{in}} \in \arg\min_{\theta}\ & \|\theta\|_2 \\ \text{s.t. } & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i)\rangle \\ & + \|\phi(\hat{s}_i, x_i) - \phi(\hat{s}_i, \hat{x}_i)\|_2 \le 0 \\ & \qquad \forall x_i \in \mathbb{X}(\hat{s}_i),\ \forall i \in [N],\end{aligned} \tag{11}$$

where $\theta^{\text{in}} = \overline{\theta}^{\text{in}}/\|\overline{\theta}^{\text{in}}\|_2$ *is an optimal solution of Problem* (10).

Figure 2 presents a geometrical intuition behind the alternative reformulation of Corollary 3.1 in a simple two-dimensional example. Exploiting the nonempty interior assumption $\text{int}(\mathbb{C}) \neq \emptyset$, we were able to formulate Problem (11) without a norm equality constraint, making it a convex optimization problem. In particular, it is tractable whenever $\mathbb{X}(\hat{s}_i)$ has finitely many elements for all $i \in [N]$. In Sections 4.2 and 5, we discuss approaches to deal with cases when $\mathbb{X}(\hat{s}_i)$ may have infinitely many elements. The tractability results in Theorem 3.1 and Corollary 3.1 build on a connection to well-known problems in the optimization literature: computing extremal volume balls.

Remark 3.2 (Connection with Extremal Volume Balls). Interestingly, the circumcenter (respectively, incenter) problem is closely related to the problem of finding the minimum (respectively, maximum) volume ball that covers (respectively, lies inside) a nonempty polyhedron. The difference is that instead of optimizing the volume of a ball so that it covers (respectively, lies inside) a nonempty polyhedron, we optimize the aperture angle of a circular cone so that it covers (respectively, lies inside) a polyhedral cone (see Figure 1). It is known that computing the minimum volume ball that contains a polyhedron defined by linear inequalities is an intractable convex optimization problem (Nemirovski 1996, section 10.5.1), and Theorem 3.1 indicates that the circumcenter has the same property. On the other hand, finding the maximum volume ball that lies inside a polyhedron defined by linear inequalities (also known as the Chebyshev center of the polyhedron) is known to be equivalent to a

convex optimization problem (Boyd and Vandenberghe 2004, section 8.5.1), and Corollary 3.1 also confirms that the incenter has the same property.

Because the problem of extremal volume balls is a special case of the more general problem of extremal volume ellipsoids (Boyd and Vandenberghe 2004, section 8.4), one natural generalization of the incenter and circumcenter concepts is to use *ellipsoidal cones*. For instance, an ellipsoidal generalization of the circumcenter concept is used by Besbes et al. (2023) for online IO problems. An ellipsoidal generalization of the incenter reformulation of Theorem 3.2 is

$$\begin{aligned} \max_{\theta, A} \ & \log(\det(A)) \\ \text{s.t.} \ & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i) \rangle \\ & \quad + \|A(\phi(\hat{s}_i, x_i) - \phi(\hat{s}_i, \hat{x}_i))\|_2 \ \leq \ 0 \\ & \qquad\qquad \forall x_i \in \mathbb{X}(\hat{s}_i), \ \forall i \in [N] \\ & \|\theta\|_2 = 1, \quad A \succcurlyeq 0, \end{aligned} \tag{12}$$

which is based on Boyd and Vandenberghe (2004, section 8.4.2). Even though to the best of our knowledge, there are no theoretical results for the performance of ellipsoidal incenter and circumcenter approaches for offline IO problems, their performance in our numerical experiments in Section 6 is indeed promising.

We end this section presenting a generalization of the program (11). This generalization will give us the flexibility to exploit structures specific to different IO problems. We generalize (11) as

$$\begin{aligned} \min_{\theta} \ & \mathcal{R}(\theta) \\ \text{s.t.} \ & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i) \rangle + d(\hat{x}_i, x_i) \ \leq \ 0 \\ & \qquad\qquad \forall x_i \in \mathbb{X}(\hat{s}_i), \ \forall i \in [N] \\ & \theta \in \Theta. \end{aligned} \tag{13}$$

This generalization occurs on three levels.

(i) **Regularizer:** We generalize the objective function to a general regularization function $\mathcal{R} : \mathbb{R}^p \to \mathbb{R}$. For example, we could have $\mathcal{R}(\theta) = \|\theta - \hat{\theta}\|_2^2$, where $\hat{\theta}$ is an a priori belief or estimate of the true cost vector, or $\mathcal{R}(\theta) = \|\theta\|$ for a general norm.

(ii) **Distance in the response space:** Instead of computing the distance between vectors using the Euclidean distance, we consider a general distance function $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$. For instance, $d(\hat{x}, x) = \|\hat{x} - x\|$ on continuous spaces or $d(\hat{x}, x) = I(\hat{x}, x)$ on discrete spaces, where $I(\hat{x}, x) = 0$ if $\hat{x} = x$; otherwise, $I(\hat{x}, x) = 1$. We note that d could also be a function of $\hat{s}$. For instance, we could use it to penalize the distance between $\hat{x}$ and x in the *feature space* by choosing $d(\hat{x}, x) = \|\phi(\hat{s}, \hat{x}) - \phi(\hat{s}, x)\|$. For simplicity, we omit this dependency.

(iii) **Prior information constraint:** We add the additional constraint $\theta \in \Theta$. The set Θ is used to encode any prior information or assumption we may have on the expert's true cost function (e.g., nonnegativity of the cost vector).

Notice that as long as $\mathcal{R}(\theta)$ is convex in θ and the set Θ is convex, Problem (13) is a convex optimization problem.

4. Augmented Suboptimality Loss

In the previous section, we have shown how one can tackle IO problems when the data set $\hat{\mathcal{D}}$ is consistent with some cost $F(s, x) = \langle \theta^\star, \phi(s, x) \rangle \in \mathcal{F}_\phi$. Moreover, to use the proposed approaches in practice, we also need to be able to list all elements of $\mathbb{X}(\hat{s}_i)$ for all $i \in [N]$ because each of these elements induces one inequality constraint (e.g., see (13)). In this section, we present approaches to tackle an IO problem based on a *loss function*, which will allow us to handle problems with possibly inconsistent data and when the cardinality of $\mathbb{X}(\hat{s}_i)$ is very large (or even infinite). We first introduce a novel loss function for IO problems and show how solving the IO problem using this loss can be interpreted as a relaxation of the optimization program (13).

Recall that in IO problems, our goal is to find a parameter vector θ such that when solving the optimization problem $\min_{x \in \mathbb{X}(s)} \langle \theta, \phi(s, x) \rangle$, we can reproduce (or approximate) the action the expert would have taken given the same signal $s \in \mathbb{S}$ (see Section 2). In the data-driven IO literature, this problem is usually posed as a (regularized) empirical loss minimization problem

$$\min_{\theta \in \Theta} \kappa \mathcal{R}(\theta) + \frac{1}{N} \sum_{i=1}^{N} \ell_\theta(\hat{s}_i, \hat{x}_i), \tag{14}$$

where $\mathcal{R} : \mathbb{R}^p \to \mathbb{R}$ is a regularization function, κ is a nonnegative regularization parameter, and $\ell_\theta : \mathbb{S} \times \mathbb{X} \to \mathbb{R}$ is some loss function, designed in a way that by solving (14), we find a cost vector θ such that the function $\langle \theta, \phi(s, x) \rangle$ achieves our IO goal. Similar to supervised learning methods in machine learning, the regularization parameter κ should be chosen to improve the out-of-sample performance of the learned model (e.g., using crossvalidation techniques). In this work, we propose a new loss function for IO problems, which we name *augmented suboptimality loss*.

Definition 4.1 (Augmented Suboptimality Loss). Given a signal-response pair $(\hat{s}, \hat{x})$, the augmented suboptimality loss of a cost vector θ is

$$\ell_\theta(\hat{s}, \hat{x}) := \max_{x \in \mathbb{X}(\hat{s})} \{ \langle \theta, \phi(\hat{s}, \hat{x}) - \phi(\hat{s}, x) \rangle + d(\hat{x}, x) \}, \tag{15}$$

where ϕ is a feature mapping and d is a distance function.

To simplify the notation, we omit the dependence of the ASL on ϕ and d. We note that the ASL can be the well-known *suboptimality loss* (Mohajerin Esfahani et al. 2018) for the special case with no distance penalization (i.e., $d(\hat{x}, x) = 0$).

4.1. Connections with Incenter

Next, we show how to solve Problem (14) when the loss function is the ASL. This reformulation is useful to tackle IO problems with inconsistent data, where there is no cost vector $\theta^\star$ that describes the data set $\hat{\mathcal{D}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ perfectly. Using a standard epigraph reformulation, one can see that the optimization Problem (14) with the ASL yields the program

$$\begin{aligned} \min_{\theta, \beta_1, \dots, \beta_N} \quad & \kappa \mathcal{R}(\theta) + \frac{1}{N} \sum_{i=1}^N \beta_i \\ \text{s.t.} \quad & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i) \rangle + d(\hat{x}_i, x_i) \le \beta_i \quad \forall x_i \in \mathbb{X}(\hat{s}_i), \ \forall i \in [N] \\ & \theta \in \Theta. \end{aligned} \tag{16}$$

Problem (16) is closely related to the incenter reformulation (13). More precisely, (16) can be interpreted as a *relaxation* of (13) to handle IO problems with inconsistent data. To see this, simply notice that we can arrive at (16) by adding *slack variables* to the constraints of (13) and using κ as a trade-off parameter between the sum of slack variables (i.e., the total violation of the constraints of (13)) and the regularization term $\mathcal{R}(\theta)$. Another way to see (14) as a relaxation of (13) is to write (13) in a "loss minimization" form. Namely, notice that (13) can be written as

$$\min_{\theta \in \Theta} \mathcal{R}(\theta) + \mathbb{I}\left(\frac{1}{N} \sum_{i=1}^N \ell_\theta(\hat{s}_i, \hat{x}_i) \right), \tag{17}$$

where the indicator function $\mathbb{I}(a) = 0$ if $a \le 0$; otherwise, $\mathbb{I}(a) = \infty$. Thus, (14) can also be interpreted as Lagrangian relaxation of (17). In Online Appendix EC.1, we present a different way to motivate the ASL, namely as a convex surrogate of the so-called *predictability loss*. There, we also discuss several properties of the ASL that are relevant to IO problems.

So far, we have only considered the case when the data $\{\hat{x}_i\}_{i=1}^N$ are feasible (i.e., $\hat{x}_i \in \mathbb{X}(\hat{s}_i)$). However, in practice, it may be the case that $\hat{x}_i \notin \mathbb{X}(\hat{s}_i)$ for some $i \in [N]$. This may happen because of noise in the data acquisition or simply because of a mismatch in the choice of the set mapping $\mathbb{X}$ compared with the true constraint set used by the expert agent. For example, consider the problem where the signal $\hat{s}_i$ represents a graph and $\mathbb{X}(\hat{s}_i)$ is the set of *Hamiltonian cycles* over this graph (i.e., a graph cycle that visits each node exactly once). This scenario is common when modeling vehicle routing problems, where the nodes of the graph represent a set of customers and the Hamiltonian cycle represents the sequence of customers who a driver visited (for example, to deliver packages). In this scenario, infeasible data could reflect, for example, missing data on the complete cycle driven by the driver (i.e., noise in the data acquisition) or the fact that in reality, the driver can visit the same customer multiple times to deliver missing packages (i.e., mismatch in the choice of the set mapping $\mathbb{X}(\hat{s}_i)$).

Remark 4.1 (Handling Infeasible Data). One way to handle infeasible data is just trying to ignore these cases, treating them as outliers in the data. Numerically, because the nonnegativity of the ASL depends on the assumption that $\hat{x}_i \in \mathbb{X}(\hat{s}_i)\ \forall i \in [N]$ (see Proposition EC.1 in the Online Appendix), having infeasible data may lead to negative loss values. Thus, a possible way to handle these cases is to use a modified loss function $\tilde{\ell}_\theta(\hat{s}, \hat{x}) = \max\{0, \ell_\theta(\hat{s}, \hat{x})\}$, which simply ignores negative loss values (i.e., infeasible data). This technique shares the same principle as the bounded rationality loss proposed in Mohajerin Esfahani et al. (2018). Importantly, this modification preserves the convexity of the loss minimization problem. For instance, this modification is equivalent to simply adding the constraints $\beta_i \ge 0,\ \forall i \in [N]$ to (16).

The IO problem shares some similarities with *structured prediction* problems, which commonly refer to the class of supervised learning problems where the output space consists of a finite set of structured objects (e.g., graphs) instead of a simple set of labels, like in standard multiclass classification problems (Taskar et al. 2005). Interestingly, the IO formulation in Equation (16) is closely related to some formulations proposed to solve structured prediction problems.

Remark 4.2 (Connections with Structured Prediction). In the incenter program (16), when $\mathcal{R}$ is the Euclidean norm, d is the Hamming distance, and feature function ϕ is linear in x, learning the incenter cost vector is equivalent to the maximum margin planning problem presented in Ratliff et al. (2006), which is a type of structured prediction problem. More generally, formulation (16) is closely related to the so-called structured support vector machine (SSVM) approach for structured prediction problems (Tsochantaridis et al. 2005, Nowozin and Lampert 2011).

The connection between IO problems with discrete feasible sets and the SSVM approach to structured prediction problems is related to the fact that when the constraint set $\mathbb{X}(\hat{s})$ has finitely many elements, each of these elements can be interpreted as a class in a multiclass classification problem. Under this interpretation, the optimization Problem (16) can be viewed as a generalization of the soft-margin SVM problem, where the "classes" $x \in \mathbb{X}(\hat{s})$ can be complex structured objects, which is precisely the SSVM problem. Moreover, the concept of "margin" in the SVM framework is related to the distance function d in (16), which comes from the fact that in our incenter-based IO formulation, we want to maximize the angle between the incenter vector and the boundaries of the set $\mathbb{C}$ (see Figure 1(b)). These observations reveal an interesting connection between incenter-based approaches for IO problems with discrete

constraint sets and SSVM approaches for structured prediction problems.

In the last part of this section, we revisit this relation between the incenter and circumcenter concepts through the lens of the regret R defined in (6). In fact, given two cost vectors $(\theta, \theta^\star)$, there is an asymmetry $R(\theta, \theta^\star) \neq R(\theta^\star, \theta)$, interestingly, each of which relating to one of these concepts.

Remark 4.3 (Connections with Regret). Consider the cost vectors $(\theta, \theta^\star)$ and the regret (6b).

(i) **SPO versus suboptimality losses:** The regret $R(\theta, \theta^\star)$ is indeed the smart "predict, then optimize" loss studied in Elmachtoub and Grigas (2022), with respect to which the circumcenter in Definition 3.1 is shown to be the worst-case optimal. However, the symmetric counterpart $R(\theta^\star, \theta)$ coincides with the suboptimality loss from Mohajerin Esfahani et al. (2018), which as discussed above, is a special case of the ASL connected to the incenter in Definition 3.2.

(ii) **Convexity and tractability:** The regret R is convex in the second argument (Mohajerin Esfahani et al. 2018) (note the connection to the suboptimality loss in the previous point) but inherently nonconvex in the first argument; see Elmachtoub and Grigas (2022) for the connection of the latter to the 0-1 loss in classification problems. This observation is indeed aligned with the intractability results of circumcenter (Theorem 3.1) and the tractability of incenter (Corollary 3.1). It is worth noting that Elmachtoub and Grigas (2022) also propose a convexified version of the SPO loss (i.e., the regret in the first argument), reminiscent of the hinge loss in classification problems.

(iii) **Additional required data measurements:** Given the cost vector θ and the IO data pair $(\hat{s}, x^\star)$ (for the definition of $x^\star$, see (6a)), the regret $R(\theta^\star, \theta)$ can be computed without the knowledge of the ground truth $\theta^\star$, whereas its symmetric counterpart $R(\theta, \theta^\star)$ does depend explicitly on $\theta^\star$ (or its projection through a feature function as in Elmachtoub and Grigas 2022).

4.2. General Reformulation for Mixed-Integer Feasible Sets

In this section, we present a way to reformulate the IO problem using the ASL for problems when $\mathbb{X}(\hat{s})$ is a mixed-integer set, which generalizes many formulations from the literature. For this purpose, we consider the mixed-integer feasible set

$$\mathbb{X}(\hat{s}) = \{(y,z) \in \mathbb{R}^u \times \mathbb{Z}^v : \hat{A}y + \hat{B}z \leq \hat{c},\, z \in \mathbb{Z}(\hat{w})\}, \quad (18)$$

where $\hat{s} := (\hat{A}, \hat{B}, \hat{c}, \hat{w})$ and $\mathbb{Z}(\hat{w})$ is a bounded set that may depend on the signal $\hat{w}$. Because y is a continuous variable, $\mathbb{X}(\hat{s})$ has infinitely many elements, and we cannot use (16) for this IO problem in practice. To solve this problem, one could use first-order iterative methods to directly optimize the IO loss minimization Problem (14) (we discuss such approaches in Section 5). In this section, we leverage classical tools from convex duality to reformulate (14) with the ASL and constraint set (18) as a tractable finite convex optimization.

For this reformulation, we use the following hypothesis function:

$$\begin{aligned}\langle \theta, \phi(\hat{s}, x)\rangle = {} & \langle y, Q_{yy} y\rangle + \langle y, Q\phi_1(\hat{w}, z)\rangle \\ & + \langle q, \phi_2(\hat{w}, z)\rangle,\end{aligned} \quad (19)$$

where $x := (y,z)$, $\theta := (\mathrm{vec}(Q_{yy}), \mathrm{vec}(Q), q) \in \mathbb{R}^{u^2+um+r}$, $Q_{yy} \succcurlyeq 0$ and where $\phi_1 : \mathbb{W} \times \mathbb{Z}^v \to \mathbb{R}^m$ and $\phi_2 : \mathbb{W} \times \mathbb{Z}^v \to \mathbb{R}^r$ are feature functions (ϕ can be written in terms of ϕ_1 and ϕ_2). The choice of a quadratic hypothesis and $\mathbb{X}(\hat{s})$ with linear inequality constraints is chosen because it generalizes the linear hypothesis case and also, for simplicity of exposition. In general, as long as the hypothesis function and constraint set are convex with respect to the continuous part of the decision vector (e.g., *conic representable* problems (Mohajerin Esfahani et al. 2018)), similar results could be derived. Continuing, we use $d(\hat{x}, x) = \|\hat{y} - y\|_\infty + d_z(\hat{z}, z)$; that is, we use the ∞-norm for the continuous part of the decision vector and a general distance function for the integer part of the decision vector. We use the ∞-norm for the continuous part of the decision vector because to use reformulation techniques based on convex duality, we need the inner maximization problem of the ASL to be concave in y, and by introducing auxiliary integer variables, we can reformulate the ∞-norm as a concave function of y. More precisely, we exploit the identity $\|\hat{y} - y\|_\infty = \max_{h \in \{-1,0,1\}^u, \|h\|_1 = 1} \langle h, \hat{y} - y\rangle$.

Theorem 4.1 (ASL and Mixed-Integer Feasible Set). *For the mixed-integer feasible set* (18), *hypothesis function* (19), *and distance function* $d(\hat{x}, x) = \|\hat{y} - y\|_\infty + d_z(\hat{z}, z)$, *Problem* (14) *can be reformulated as*

$$\begin{aligned}\min\ & \kappa \mathcal{R}(\theta) + \frac{1}{N}\sum_{i=1}^{N} \beta_i \\ \text{s.t.}\ & \theta = (\mathrm{vec}(Q_{yy}), \mathrm{vec}(Q), q) \in \Theta \\ & \lambda_{ijk} \geq 0, \quad \alpha_{ijk}, \beta_i \in \mathbb{R} \\ & \langle \theta, \phi(\hat{s}_i, \hat{x}_i)\rangle + \alpha_{ijk} + \langle \lambda_{ijk}, \hat{c}_i - \hat{B}_i z_{ij}\rangle \\ & \quad - \langle q, \phi_2(\hat{w}_i, z_{ij})\rangle + \langle h_k, \hat{y}_i\rangle + d_z(\hat{z}_i, z_{ij}) \leq \beta_i \\ & \begin{bmatrix} Q_{yy} & Q\phi_1(\hat{w}_i, z_{ij}) + h_k + \hat{A}_i^\top \lambda_{ijk} \\ * & 4\alpha_{ijk} \end{bmatrix} \succcurlyeq 0,\end{aligned} \quad (20)$$

where the constraints are for all $(i,j,k) \in [N] \times [M_i] \times [2u]$, $\mathbb{Z}(\hat{w}_i) := \{z_{i1}, \ldots, z_{ij}, \ldots, z_{iM_i}\}$, $M_i := |\mathbb{Z}(\hat{w}_i)|$, *and if* $k \leq u$ (*respectively*, $k > u$), h_k *is the vector of zeros except for the* k*th* (*respectively*, $(k-u)$*th*) *element, which is equal to* 1 (*respectively*, -1).

In case we use a linear hypothesis function instead of a quadratic one, the matrix inequality constraints of (20) reduce to much simpler linear equality constraints.

Corollary 4.1 (Linear Programming Reformulation for Linear Hypotheses). *For the mixed-integer feasible set* (18), *linear hypothesis function* (*i.e.*, (19) *with* $Q_{yy} = 0$), *and distance function* $d(\hat{x}, x) = \|\hat{y} - y\|_\infty + d_z(\hat{z}, z)$, *Problem* (14) *can be reformulated as*

$$\begin{aligned} \min\ & \kappa\mathcal{R}(\theta) + \frac{1}{N}\sum_{i=1}^{N}\beta_i \\ \text{s.t.}\ & \theta = (\mathrm{vec}(Q), q) \in \Theta, \quad \lambda_{ijk} \geq 0, \quad \beta_i \in \mathbb{R} \\ & \langle \theta, \phi(\hat{s}_i, \hat{x}_i)\rangle + \langle \lambda_{ijk}, \hat{c}_i - \hat{B}_i z_{ij}\rangle \\ & \quad - \langle q, \phi_2(\hat{w}_i, z_{ij})\rangle + \langle h_k, \hat{y}_i\rangle + d_z(\hat{z}_i, z_{ij}) \leq \beta_i \\ & Q\phi_1(\hat{w}_i, z_{ij}) + h_k + \hat{A}_i^\top \lambda_{ijk} = 0, \end{aligned} \tag{21}$$

where the constraints are for all $(i, j, k) \in [N] \times [M_i] \times [2u]$, $\mathbb{Z}(\hat{w}_i) := \{z_{i1}, \ldots, z_{ij}, \ldots, z_{iM_i}\}$, $M_i := |\mathbb{Z}(\hat{w}_i)|$, *and if* $k \leq u$ (*respectively*, $k > u$), h_k *is the vector of zeros except for the* k*th* (*respectively*, $(k-u)$*th*) *element*, *which is equal to* 1 (*respectively*, -1).

We note that one can reduce the number of constraints in these reformulations by a factor of $2u$ by using $d(\hat{x}, x) = d_z(\hat{z}, z)$ (i.e., only penalizing the integer part of the decision vector, which is equivalent to setting $h_k = 0 \ \forall k \in [2u]$).

Remark 4.4 (Generality of Theorem 4.1). Reformulation (20) in Theorem 4.1, for the IO problem with mixed-integer decision sets, generalizes several reformulations from the literature. For instance, for an IO problem with purely continuous decision sets and linear hypothesis functions, reformulation (20) with $d_z(\hat{z}, z) = 0$ and $h_k = 0 \ \forall k \in [2u]$ reduces to classical inverse linear optimization reformulations (Ahuja and Orlin 2001, Chan et al. 2019). Similarly, if the IO problem has a purely continuous decision set and a quadratic hypothesis function, then the program (20) with $d_z(\hat{z}, z) = 0$ and $h_k = 0 \ \forall k \in [2u]$ reduces to the linear matrix inequality (LMI) reformulation in(Akhtar et al. (2021). If the decision set is purely discrete (that is, $\mathbb{X}(\hat{s})$ has finitely many elements), then (20) and (21) recover (16).

In conclusion, by dualizing the continuous part of the problem, Theorem 4.1 and Corollary 4.1 present a way to deal with a case when the feasible set $\mathbb{X}(\hat{s})$ is a mixed-integer set. It is worth noting that when the feasible set $\mathbb{X}(\hat{s})$ is a mixed-integer set, the size of reformulations (20) and (21) grows linearly in the number of integer variables, in the dimension of the continuous variables, and in the size of the data set. That means that for inverse optimization problems with large data sets and/or mixed-integer sets with a large number of decision variables, these reformulations can be computationally challenging to solve. This issue is the main focus of our tailored first-order algorithm in the next section.

5. Tailored Algorithm: Stochastic Approximate Mirror Descent

All approaches to solving IO problems presented so far depend on optimization programs with possibly a large number of constraints. Given a data set $\{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$, these optimization problems have at least $\sum_{i=1}^N |\mathbb{X}(\hat{s}_i)|$ (or $\sum_{i=1}^N |\mathbb{Z}(\hat{w}_i)|$ for the mixed-integer case) constraints. When N is too large (i.e., we have too many signal-response examples) or $|\mathbb{X}(\hat{s}_i)|$ is too large, solving these optimization programs may be intractable in practice. Thus, the focus of this section is to develop a tailored first-order algorithm to tackle such IO problems in a provably efficient way, opening doors to a wider range of real-world applications. A straightforward first-order approach to solve (14) is the standard *project subgradient method* (Shor 1985). This is a general first-order optimization algorithm, and it converges to an optimal solution of the problem under mild convexity conditions on the objective function and constraint set. For example, this was the approach used in Ratliff et al. (2006) to solve the maximum margin planning problem (see Remark 4.2). Another idea is to solve (14) as a minimax problem. The paper by Taskar et al. (2006) showed that in some special cases, this problem can be formulated as a bilinear saddle point problem, and then, Taskar et al. (2006) used Nesterov's dual extragradient method (Nesterov 2007) to solve it. However, other than only being applicable to a restrictive class of problems, this approach also requires $2N$ projections onto $\mathbb{X}(\hat{s}_i)$ to be computed per iteration of the algorithm, which may be prohibitive in many applications. Other algorithms for minimax problems, such as Nemirovski's Mirror-Prox algorithm (Nemirovski 2004), would suffer from similar drawbacks. In this section, we introduce an efficient optimization algorithm that can be applied to general IO problems while also exploiting the specific structures that stem from IO problems. To this end, we define the following notion of a stochastic approximate subgradient.

Definition 5.1 (Stochastic Approximate Subgradient). Let $f : \Theta \to \mathbb{R}$ be a convex function. We say that the random vector $\tilde{g}_\varepsilon(\theta)$ is a *stochastic approximate subgradient* of f at θ if $\mathbb{E}[\tilde{g}_\varepsilon(\theta)|\theta] = g_\varepsilon(\theta)$ and that $g_\varepsilon(\theta)$ is an ε-subgradient of f for some $\varepsilon \geq 0$: that is,

$$f(\theta) - f(\nu) \leq \langle g_\varepsilon(\theta), \theta - \nu\rangle + \varepsilon, \quad \forall \nu \in \Theta.$$

For different notions of approximate subgradients, see Ratliff et al. (2007) and Taşkesen et al. (2023). Next, we propose to solve IO problems with a novel algorithm, the *stochastic approximate mirror descent*. We first define the SAMD as an algorithm to optimize general convex programs and prove convergence rates for it. After that, we discuss how we can use the SAMD algorithm

to exploit the structure of IO problems. For a function f and set Θ, the SAMD updates are

$$\theta_{t+1} = \arg\min_{\theta\in\Theta}\{\eta_t\langle\tilde{g}_{\varepsilon_t}(\theta_t),\theta\rangle + \mathcal{B}_\omega(\theta,\theta_t)\}, \tag{22}$$

where η_t is the step size, the function $\mathcal{B}_\omega$ is the Bregman divergence with respect to $\omega:\Theta\to\mathbb{R}$ (Bubeck 2015, section 4), and $\tilde{g}_{\varepsilon_t}(\theta_t)$ is a *stochastic approximate subgradient* of f as defined in Definition 5.1. The SAMD algorithm can be interpreted as a combination of a stochastic mirror descent algorithm (Bubeck 2015, section 6.1) and an ε-subgradient method (Bertsekas 2015, section 3.3). Next, we prove a convergence rate for the SAMD algorithm, where we use the concept of *relative strong convexity*, which is a generalization of the standard strong convexity property (Lu et al. 2018). Namely, if $\mathcal{R}$ is α-strongly convex relative to $\mathcal{B}_\omega$, then $\mathcal{R}(x)-\mathcal{R}(y) \le \langle g(x),x-y\rangle - \alpha\mathcal{B}_\omega(y,x)$, where $g(x)\in\partial\mathcal{R}(x)$.

Proposition 5.1 (SAMD Convergence Rate). *Let $f:\Theta\to\mathbb{R}$ be a convex function and Θ be a convex set. Assume $\mathcal{B}_\omega(\theta,\nu)\le R^2$, for some $R>0$, and $\mathbb{E}[\|\tilde{g}_{\varepsilon_t}(\theta)\|_*^2|\theta]\le G^2$, $\forall\theta,\nu\in\Theta$. Using $\eta_t = c/\sqrt{t}$ for some constant $c>0$, the SAMD algorithm guarantees*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T}\theta_t\right)\right] - \min_{\theta\in\Theta} f(\theta) \le \left(\frac{R^2}{c}+cG^2\right)\frac{1}{\sqrt{T}} + \frac{1}{T}\sum_{t=1}^{T}\varepsilon_t.$$

Moreover, if we assume $f=h+\mathcal{R}$, where h is convex and $\mathcal{R}$ is α-strongly convex relative to $\mathcal{B}_\omega$, then using $\eta_t = 2/\alpha(t+1)$, the SAMD algorithm guarantees

$$\mathbb{E}\left[f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T}t\theta_t\right)\right] - \min_{\theta\in\Theta} f(\theta) \le \frac{2G^2}{\alpha(T+1)} + \frac{2}{T(T+1)}\sum_{t=1}^{T}t\varepsilon_t.$$

The convergence rate of the SAMD algorithm is a combination of the $O(1/\sqrt{T})$ (or $O(1/T)$) rate of the stochastic mirror descent algorithm plus a term that depends on $\{\varepsilon_t\}_{t=1}^T$ because of the use of approximate subgradients. Notice that if we use the SAMD algorithm with errors ε_t diminishing at a rate $O(1/t)$ (i.e., as t increases, we use increasingly more precise approximate subgradients), then the convergence rates of Proposition 5.1 reduce to $O(1/\sqrt{T})$ and $O(1/T)$. The SAMD algorithm differs from a simple projected subgradient method in three major ways; it uses (i) mirror descent updates, (ii) stochastic subgradients, and (iii) approximate subgradients. Next, we discuss how each of these properties can be used to exploit the structure of the IO Problem (14).

5.1. Mirror Descent Updates

For instance, consider (14) with $\mathcal{R}(\theta)=\|\theta\|_1$ and $\Theta=\mathbb{R}^p$. This problem can be equivalently written as

$$\begin{aligned}\min_{\theta\in\mathbb{R}^p}\quad & \frac{1}{N}\sum_{i=1}^{N}\ell_\theta(\hat{s}_i,\hat{x}_i)\\ \text{s.t.}\quad & \tilde{\kappa}\|\theta\|_1 \le 1\end{aligned} \tag{23}$$

for some $\tilde{\kappa}$ that depends on κ and the data $\{(\hat{s}_i,\hat{x}_i)\}_{i=1}^N$. Then, by introducing the nonnegative variables θ^+ and θ^-, defining $\tilde{\theta}:=(\theta^+,\theta^-)$ (i.e., the concatenation of θ^+ and $\theta-$), and setting $\theta=\theta^+-\theta^-=[I-I]\tilde{\theta}$ (here, I is the identity matrix and $[I-I]$ is a block matrix), (23) can be written as

$$\min_{\tilde{\theta}\in\Delta_{\tilde{\kappa}}}\frac{1}{N}\sum_{i=1}^{N}\ell_{[I-I]\tilde{\theta}}(\hat{s}_i,\hat{x}_i), \tag{24}$$

where $\Delta_{\tilde{\kappa}}:=\{\tilde{\theta}\in\mathbb{R}^{2p}:\tilde{\kappa}\|\tilde{\theta}\|_1\le 1,\tilde{\theta}\ge 0\}$ (Tibshirani 1996). In other words, (23) can be recast as an optimization problem over a simplex. Next, choosing $\omega(\theta)=\sum_{i=1}^p\theta_i\log(\theta_i)$, the SAMD updates applied to (24) can be written as "exponentiated updates":

$$\tilde{\theta}_{t+1} = \begin{cases}\tilde{\theta}_t\odot\exp(-\eta_t\tilde{g}_{\varepsilon_t}(\tilde{\theta}_t)) & \text{if } \tilde{\kappa}\|\tilde{\theta}_t\odot\exp(-\eta_t\tilde{g}_{\varepsilon_t}(\tilde{\theta}_t))\|_1\le 1\\ \dfrac{\tilde{\theta}_t\odot\exp(-\eta_t\tilde{g}_{\varepsilon_t}(\tilde{\theta}_t))}{\tilde{\kappa}\|\tilde{\theta}_t\odot\exp(-\eta_t\tilde{g}_{\varepsilon_t}(\tilde{\theta}_t))\|_1} & \text{otherwise.}\end{cases}$$

Exponentiated updates are known to have better convergence properties compared with standard subgradient descent updates for "simplex constrained" problems, as well as not requiring solving optimization problems to project onto the simplex. For a detailed discussion on the advantages of mirror descent updates for different settings, see Juditsky and Nemirovski (2011, section 5.7). In general, by choosing $\omega(\theta)=\frac{1}{2}\|\theta\|_2^2$, the SAMD updates reduce to standard project subgradient updates

$$\theta_{t+1}=\Pi_\Theta(\theta_t-\eta_t\tilde{g}_{\varepsilon_t}(\theta_t))$$

(that is, the subgradient method can be seen as a special case of the mirror descent algorithm).

5.2. Stochastic Subgradients

Using stochastic subgradients is advantageous when computing a stochastic subgradient is computationally cheaper than computing a deterministic subgradient. This observation is supported by the fact that, in expectation, the stochastic subgradient method guarantees the same convergence rate as its deterministic version (Bubeck 2015, theorem 6.1). For the IO loss minimization Problem (14), computing a stochastic subgradient can be significantly cheaper than computing the full subgradient because of the finite sum structure of (14). Namely, by Danskin's theorem (Bertsekas 2008, section

B.5), we have that the subdifferential of the ASL with respect to θ is

$$\partial \ell_\theta(\hat{s},\hat{x}) = \operatorname{conv}\left\{ \phi(\hat{s},\hat{x}) - \phi(\hat{s},x^\star(\hat{s})) : x^\star(\hat{s}) \in \arg\max_{x\in\mathbb{X}(\hat{s})} \{d(\hat{x},x) - \langle \theta, \phi(\hat{s},x)\rangle\} \right\}. \tag{25}$$

Thus, to compute a subgradient of $\frac{1}{N}\sum_{i=1}^N \ell_\theta(\hat{s}_i,\hat{x}_i)$, we need to solve N maximization problems, one for each signal-response pair $\{(\hat{s}_i,\hat{x}_i)\}_{i=1}^N$. On the other hand, by sampling an index j uniformly from $[N]$, we have that $g_j(\theta) \in \partial \ell_\theta(\hat{s}_j,\hat{x}_j)$ is a stochastic subgradient of $\frac{1}{N}\sum_{i=1}^N \ell_\theta(\hat{s}_i,\hat{x}_i)$: that is, $\mathbb{E}_{j\sim[N]}[g_j(\theta)|\theta] = (1/N)\sum_{i=1}^N g_i(\theta)$. This is a standard method for computing stochastic (sub-)gradients for empirical risk minimization-type problems (e.g., Bubeck 2015, chapter 6). In summary, to compute an unbiased stochastic subgradient of (14), instead of N, we need to solve only *one* maximization problem.

5.3. Approximate Subgradients

As shown in (25), we need to solve the optimization problem $\max_{x\in\mathbb{X}(\hat{s})}\{d(\hat{x},x) - \langle \theta, \phi(\hat{s},x)\rangle\}$ in order to compute a subgradient of the ASL. However, in practice, it may be too costly to solve this optimization problem to optimality at each iteration of the algorithm. Thus, it would be useful if we could use an approximate solution to this problem instead of an optimal one. Turns out that, indeed, given an *ε-approximate solution* to the maximization problem, which is a feasible point x_ε such that

$$d(\hat{x},x_\varepsilon) - \langle \theta, \phi(\hat{s},x_\varepsilon)\rangle \geq \max_{x\in\mathbb{X}(\hat{s})}\{d(\hat{x},x) - \langle \theta, \phi(\hat{s},x)\rangle\} - \varepsilon,$$

we can construct an ε-subgradient of the ASL.

Lemma 5.1 (Approximate Solutions and $\boldsymbol{\varepsilon}$-Subgradients). *Let x_ε be a feasible, ε-suboptimal solution of $\max_{x\in\mathbb{X}(\hat{s})}\{d(\hat{x},x) - \langle \theta, \phi(\hat{s},x)\rangle\}$. Thus, the vector $g_\varepsilon(\theta) = \phi(\hat{s},\hat{x}) - \phi(\hat{s},x_\varepsilon)$ is an ε-subgradient of $\ell_\theta(\hat{s},\hat{x})$ with respect to θ: that is,*

$$\ell_\theta(\hat{s},\hat{x}) - \ell_\nu(\hat{s},\hat{x}) \leq \langle \phi(\hat{s},\hat{x}) - \phi(\hat{s},x_\varepsilon), \theta - \nu\rangle + \varepsilon.$$

In summary, we can use an approximate solution to the inner maximization problem of the ASL to construct ε-subgradients, and as proven in Proposition 5.1, these can be effectively used to compute an approximate solution of convex optimization problems. Finally, it is not difficult to show that by combining stochastic subgradients (i.e., by using only one random signal-response pair) and approximate subgradients (i.e., solving the respective maximization problem approximately), one gets a stochastic approximate subgradient (Definition 5.1). Putting these ideas together, Algorithm 5.1 shows the pseudocode of the SAMD algorithm applied to Problem (14), where we denote the gradient (or a subgradient) of $\mathcal{R}$ as $\nabla\mathcal{R}$. In line 3 of Algorithm 5.1, one example is sampled from the data set for each iteration of the algorithm. However, in practice, it may be advantageous to instead sample a batch of $1 \leq B \leq N$ examples and use this batch of data to compute the approximate stochastic subgradient. By changing the size B of the batch, we can control the trade-off between having more precise subgradients (large B) and faster subgradient computations (small B). This idea is explored in the numerical experiments of Section 6.4.

Algorithm 5.1 (SAMD Algorithm for (14))

1: **Input:** Step-size sequence $\{\eta_t\}_{t=1}^T$, κ, θ_1, d, ϕ, ω, $\nabla\mathcal{R}$ and data set $\{(\hat{s}_i,\hat{x}_i)\}_{i=1}^N$.
2: **for** $t = 1,\dots,T$ **do**
3: Sample j uniformly from $\{1,\dots,N\}$
4: Compute x_t, a (possibly approximate) solution of $\max_{x\in\mathbb{X}(\hat{s}_j)}\{d(\hat{x}_j,x) - \langle \theta_t, \phi(\hat{s}_j,x)\rangle\}$
5: Approximate stochastic subgradient: $\tilde{g}_t(\theta_t) = \kappa\nabla\mathcal{R}(\theta_t) + \phi(\hat{s}_j,\hat{x}_j) - \phi(\hat{s}_j,x_t)$
6: Mirror descent step: $\theta_{t+1} = \arg\min_{\theta\in\Theta}\{\eta_t\langle \tilde{g}_t(\theta_t), \theta\rangle + \mathcal{B}_\omega(\theta,\theta_t)\}$
7: **end for**
8: **Output:** $\{\theta_t\}_{t=1}^T$

We end this section by briefly discussing the case of *online IO*. In this scenario, instead of having a data set of signal-response data up front, we receive one signal-response pair at a time. After receiving each signal-response data pair, we must choose a cost vector θ_t using all the information gathered so far and evaluate the loss of this cost vector. The final performance of the algorithm is measured using *regret* performance metrics.

Remark 5.1 (Regret Bounds and Online IO). Bärmann et al. (2017) use an online multiplicative weights updates (MWU) algorithm to prove an $O(\sqrt{T})$ regret bound, and more recently, Besbes et al. (2023) use an online adaptation of the circumcenter concept to prove an $O(\log(T))$ regret bound. In line 3 of Algorithm 5.1, if instead of sampling a new signal-response example, we use the online data received at that iteration, Algorithm 5.1 can be readily adapted to online IO problems. Moreover, the convergence bound of Proposition 5.1 may also be straightforwardly converted to an $O(\sqrt{T})$ regret bound (or $O(\log(T))$ for the strongly convex case) (Orabona 2019, chapter 3). In particular, Algorithm 5.1 can be interpreted as a generalization of the MWU algorithm (Allen-Zhu and Orecchia 2014, appendix A.2).

6. Numerical Experiments

In this section, we numerically evaluate the approaches to inverse optimization proposed in this paper. All linear and quadratic programs were solved using Gurobi.

All semidefinite programs were solved using CVXPY with MOSEK as the solver. For all results presented in this section, we learn the expert's cost function using a training data set of expert signal-response data and evaluate its performance using a test data set (i.e., out-of-sample performance). Moreover, in every plot, we report the average performance value for 10 randomly generated true cost vectors as well as the 5th and 95th percentile bounds. In Online Appendix EC.4.2, the in-sample results are reported, complementing the ones from this section. All of our experiments are reproducible and are part of the examples in the InvOpt Python package (Zattoni Scroccaro 2023).

6.1. Consistent Data

In this section, we numerically evaluate the approaches discussed in Section 3 for IO problems with consistent data.

Decision Problem of the Expert. To generate its decisions, the expert solves the binary linear program

$$\begin{aligned} \min_{x} \quad & \langle \theta, x \rangle \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \{0,1\}^n, \end{aligned} \tag{26}$$

where $\theta \in \mathbb{R}^n_+$ is the cost vector, $A \in \mathbb{R}^{t \times n}$, $b \in \mathbb{R}^t$, and x is the decision vector. Although the decision Problem (26) is presented as a general binary linear program, many real-world problems can be modeled within this class of optimization problems. As a motivating example, we briefly discuss the problem of modeling consumer behavior and how it fits (26). In this problem, the expert is a consumer who given some contextual information, decides which products to buy from a set of n products. That is, each component of the decision vector $x \in \{0,1\}^n$ corresponds to one of the n products and equals one if the consumer buys it and zero otherwise. The signal with contextual information could be, for instance, $\hat{s} = (A, b)$, where each component of $A \in \mathbb{R}^n$ corresponds to the price of each product and $b \in \mathbb{R}$ corresponds to the total budget of the consumer. In this case, the constraint $Ax \leq b$ simply represents the budget constraint of the consumer. Finally, each component of the cost vector $\theta \in \mathbb{R}^n$ represents (the negative of) the utility the consumer assigns to each product, and by minimizing $\langle \theta, x \rangle$, the goal of the consumer is to buy the set of products that maximizes its utility while respecting its budget constraint. Thus, in this context, solving the IO problem translates to learning the utility function underlying the actions of a consumer agent.

Data Generation. To generate training and test data, we sample cost vectors uniformly from $\{\theta \in \mathbb{R}^n : 0 \leq \theta \leq 1\}$, and we sample $\hat{A}$ uniformly from $\{A \in \mathbb{R}^{t \times n} : -1 \leq A \leq 0\}$ and $\hat{b}$ uniformly from $\{b \in \mathbb{R}^t : -1 \leq b \leq 0\}$. To make sure that the problem instances are feasible, we check if the sum of each row of $\hat{A}$ is larger than the respective component of $\hat{b}$, which ensures that $x = (1, \ldots, 1)$ is a feasible solution of (26). Given a signal $\hat{s} = (\hat{A}, \hat{b})$, we generate a response $\hat{x}$ by solving (26) (i.e., $\hat{x} \in \arg\min_{x \in \mathbb{X}(\hat{s})} \langle \theta, x \rangle$). To evaluate the IO approaches, we generate 10 random cost vectors θ_{true}. For each of these cost vectors, we generate a training data set $\hat{\mathcal{D}}_{\text{train}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ and a test data set $\hat{\mathcal{D}}_{\text{test}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$, with $N = 100$, $n = 6$, and $t = 4$.

IO Approaches. We compare seven approaches to choose a vector from the set of consistent cost vectors (4).

- **Feasibility:** We use (5) with $\|\theta\|_1 = 1$, $s = (A, b)$, $\mathbb{X}(s) = \{x \in \{0,1\}^n : Ax \leq b\}$, $\phi(x, s) = x$, augmented with the constraint $\theta \in \Theta = \{\theta \in \mathbb{R}^n : \theta \geq 0\}$.
- **Incenter:** We use (13) with $\mathcal{R}(\theta) = \frac{1}{2}\|\theta\|_2^2$, $s = (A, b)$, $\mathbb{X}(s) = \{x \in \{0,1\}^n : Ax \leq b\}$, $\phi(x, s) = x$, $d(\hat{x}_i, x_i) = \|\hat{x}_i - x_i\|_2$, and $\Theta = \{\theta \in \mathbb{R}^n : \theta \geq 0\}$.
- **Circumcenter:** We solve the following optimization problem:

$$\begin{aligned} \min_{\|\theta\|_2 = 1} \quad & r \\ \text{s.t.} \quad & \max_{\substack{\tilde{\theta} \in \mathbb{C} \\ \|\tilde{\theta}\|_2 = 1}} \|\theta - \tilde{\theta}\|_2^2 \leq r, \end{aligned} \tag{27}$$

which is an epigraph reformulation of the circumcenter problem (see the Proof of Theorem 3.1 in the Online Appendix). To solve this optimization problem, we substitute the max constraint by the constraints $\|\theta_E - \tilde{\theta}\|_2^2 \leq r$, $\forall \theta_E \in E$, where E is the set of normalized extreme points of $\mathbb{C}$.

- **Ellipsoidal incenter:** We solve (12), the ellipsoidal version of the incenter concept. In our implementation, we use the convex constraint $\|\theta\| \leq 1$ instead of $\|\theta\| = 1$ because one can show that the constraint $\|\theta\| \leq 1$ will always be tight at an optimum when $\text{int}(\mathbb{C}) \neq \emptyset$.
- **Ellipsoidal circumcenter:** We solve

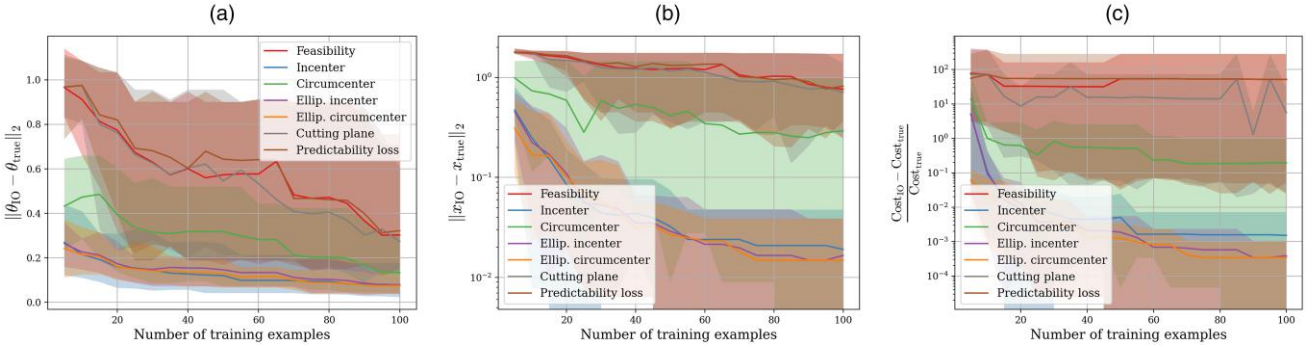
$$\begin{aligned} \max_{A \succcurlyeq 0, \theta} \quad & \text{logdet}(A) \\ \text{s.t.} \quad & \max_{\substack{\tilde{\theta} \in \mathbb{C} \\ \|\tilde{\theta}\|_2 = 1}} \|A(\tilde{\theta} - \theta)\|_2^2 \leq 1, \end{aligned}$$

which is an ellipsoidal generalization of the circumcenter reformulation (27).

- **Cutting plane:** We use the cutting-plane algorithm of Wang (2009).
- **Predictability loss:** We use the predictability loss of Aswani et al. (2018) (Online Appendix EC.1).

Results. Figure 3 shows the results for this scenario. To make the comparisons consistent, before evaluating the results, all cost vectors are normalized. For all the plots, the x axis refers to the number of training examples

Figure 3. (Color online) Out-of-Sample Results for the Consistent Data Scenario



Notes. (a) Difference between the true cost vector (θ_{true}) and the one learned using IO (θ_{IO}). (b) Average error between the decision generated by θ_{true} and θ_{IO}. (c) Relative difference between the cost of the decisions generated using θ_{true} and θ_{IO}. Ellip., ellipsoidal.

used to compute the results. The idea is to evaluate how efficient these approaches are with respect to the amount of data fed to them. Figure 3(a) shows the difference between the cost vector returned by the IO approaches (which we name θ_{IO}) and the cost vector used to generate the data (θ_{true}). As expected, the more data we feed to the approaches, the closer θ_{IO} tends to get from θ_{true}. Comparing them, the incenter, ellipsoidal incenter, and ellipsoidal circumcenter approaches have similar performance and clearly outperform the other approaches. Figure 3(b) shows the average difference between the optimal decision of (26) using θ_{IO} (which we name x_{IO}) and the decision in the test data set (which we name x_{true}). Again, we see that the incenter, ellipsoidal incenter, and ellipsoidal circumcenter approaches present the best performance. Figure 3(c) shows the normalized difference between the cost of the expert decisions and the cost of the decisions using θ_{IO}. More precisely, we define $\text{Cost}_{\text{IO}} := \sum_{i=1}^{N} \langle \theta_{\text{true}}, x_{\text{IO},i} \rangle$ and $\text{Cost}_{\text{true}} := \sum_{i=1}^{N} \langle \theta_{\text{true}}, \hat{x}_i \rangle$ and compare the relative difference between them. Notice that this difference will always be nonnegative by the optimality of $\hat{x}_i$. Once again, the incenter, ellipsoidal incenter, and ellipsoidal circumcenter approaches outperform the other approaches.

Moreover, Table 1 shows the time it took to generate the results of this section for each approach. As can be seen, even though the incenter, ellipsoidal incenter, and ellipsoidal circumcenter approaches show similar performance, the incenter approach is at least one order of magnitude faster to compute. Although these numbers depend on the actual implementation of each IO approach, such a difference in solving time is expected because the incenter problem can be formulated as a quadratic program, whereas the ellipsoidal approaches involve semidefinite constraints. Finally, regarding the main driver for improved out-of-sample performance of the incenter and ellipsoidal approaches, one possible explanation is that these approaches optimize for a vector in the interior of $\mathbb{C}$: that is, away from the boundaries of this set. This conclusion is based on the following observations; recall the intuition for the incenter vector provided in Remark 3.1, also visualized in Figure 1(b). That is, the idea behind the incenter is to find the vector farthest away from the boundaries of the set $\mathbb{C}$. Also, recall that the authors in Besbes et al. (2023) showed that circumcenter concept fails for online IO problems precisely because in the worst case, the circumcenter vector can lie exactly in the boundary of the set $\mathbb{C}$ (Besbes et al. 2023, figure 2), and the ellipsoidal circumcenter addresses this problem because its circumcenter lies inside the set $\mathbb{C}$ (Besbes et al. 2023, figure 3). Therefore, the incenter, ellipsoidal incenter, and ellipsoidal circumcenter in some sense optimize for vector in the interior of $\mathbb{C}$, and as shown in Figure 3, they achieve similar out-of-sample performance in this offline IO experiment (although the incenter vector is computationally cheaper to compute).

6.2. Inconsistent Data

In this section, we numerically evaluate the approaches discussed in Section 4 for IO problems with inconsistent data.

Decision Problem of the Expert. To generate its decisions, the expert solves the binary linear program (26), where $\theta \in \mathbb{R}^n$ is the cost vector, $A \in \mathbb{R}^{t \times n}$, $b \in \mathbb{R}^t$, and x is the decision vector. Notice that different from the

Table 1. Computational Time to Generate the Results of Figure 3

Approach	Feasibility	Incenter	Circumcenter	Ellipsoidal incenter	Ellipsoidal circumcenter	Cutting plane	Predictability loss
Time (seconds)	69	98	1,323	915	1,345	206	236

previous section, we do not assume that the cost vector θ is nonnegative.

Data Generation. To generate training and test data, we sample cost vectors uniformly from $\{\theta \in \mathbb{R}^n : -1 \leq \theta \leq 1\}$, and we sample $\hat{A}$ uniformly from $\{A \in \mathbb{R}^{t \times n} : -1 \leq A \leq 1\}$ and $\hat{b}$ uniformly from $\{b \in \mathbb{R}^t : -1 \leq b \leq 0\}$. After generating a signal $\hat{s} = (\hat{A}, \hat{b})$, we check if $\mathbb{X}(\hat{s})$ is nonempty to ensure that the problem instance is feasible. To generate the response vectors $\hat{x}$ for the training data sets, we solve Problem (26) with noise added to the cost vector (i.e., $\hat{x} \in \arg\min_{x \in \mathbb{X}(\hat{s})} \langle \theta + w, x \rangle$), where $w \in \mathbb{R}^n$ is a random vector with components sampled from a normal distribution with zero mean and standard deviation equal to 0.05. This means that different from the consistent data case, (most probably) there will be no single cost vector consistent with the entire training data set. To evaluate the IO approaches, we generate 10 random cost vectors θ_{true}. For each of these cost vectors, we generate a training data set $\hat{\mathcal{D}}_{\text{train}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ (by solving the noisy version of (26), with a different noise vector w for each signal-response pair) and a test data set $\hat{\mathcal{D}}_{\text{test}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ (noiseless), with $N = 100$, $n = 10$, and $t = 8$.

IO Approaches. We compare five IO approaches for this problem.

- **SL:** We use (14) with $\kappa = 0$, $\Theta = \mathbb{R}^n$, $s = (A, b)$, $\mathbb{X}(s) = \{x \in \{0,1\}^n : Ax \leq b\}$, $\phi(x, s) = x$, and the suboptimality loss $\ell_\theta(\hat{s}, \hat{x}) = \max_{x \in \mathbb{X}(\hat{s})} \{\langle \theta, \phi(\hat{s}, \hat{x}) - \phi(\hat{s}, x) \rangle\}$. To prevent the trivial solution $\theta = 0$, we add a norm equality constraint $\|\theta\|_\infty = 1$ and solve $2n$ linear programs, one for each facet of the ∞-norm unit sphere (Mohajerin Esfahani et al. 2018).
- **ASL:** We use (14) with the ASL, $\kappa = 0.001$, $\mathcal{R}(\theta) = \frac{1}{2}\|\theta\|_2^2$, $s = (A, b)$, $\mathbb{X}(s) = \{x \in \{0,1\}^n : Ax \leq b\}$, $\phi(x, s) = x$, $d(\hat{x}_i, x_i) = \|\hat{x}_i - x_i\|_2$, and $\Theta = \mathbb{R}^n$.
- **Ellipsoidal ASL:** We solve

$$\begin{aligned} \min_{A, \theta, \beta_1, \ldots, \beta_N} \quad & -\kappa \log\Big(\det(A)\Big) + \frac{1}{N}\sum_{i=1}^N \beta_i \\ \text{s.t.} \quad & \langle \theta, \phi(\hat{s}_i, \hat{x}_i) - \phi(\hat{s}_i, x_i) \rangle \\ & + \|A(\phi(\hat{s}_i, x_i) - \phi(\hat{s}_i, \hat{x}_i))\|_2 \leq \beta_i \\ & \qquad \forall x_i \in \mathbb{X}(\hat{s}_i), \quad \forall i \in [N] \\ & \|\theta\|_2 \leq 1, \quad A \succcurlyeq 0, \end{aligned} \tag{28}$$

which is an ellipsoidal generalization of (16) (see Remark 3.2).

- **Cutting plane:** We use the cutting-plane method of Bodur et al. (2022), which is an extension of the cutting-plane algorithm of Wang (2009) for IO problems with inconsistent data.
- **Predictability loss:** We use the predictability loss of Aswani et al. (2018) (Online Appendix EC.1).

Notice that we do not test circumcenter-based approaches because they are not defined for the inconsistent data case: that is, when $\mathbb{C} \setminus \{0\} = \emptyset$.

Results. Figure 4 shows the results this scenario. The discussion on the interpretation of each performance metric presented in Figure 4 mirrors the one of Figure 3. Importantly, the approach based on the ASL and its ellipsoidal version outperforms the other approaches. Moreover, as expected, because of the noise in the training data, one can see that θ_{IO} was not able to reproduce the behavior of the expert as well as in the consistent (i.e., noiseless) case. In Table 2, we show the time it took to generate the results of this section for each approach. As can be seen, even though the ASL and the ellipsoidal ASL show similar performance, the ASL approach is more computationally efficient. Again, this difference is because the ellipsoidal ASL involves semidefinite constraints (Equation (28)), which is not the case for the standard ASL (Equation (16)).

6.3. Mixed-Integer Feasible Set

In this section, we numerically evaluate the approach from Section 4.2 for IO problems with mixed-integer feasible sets.

Decision Problem of the Expert. To generate its decisions, the expert solves a mixed-integer linear program of the form

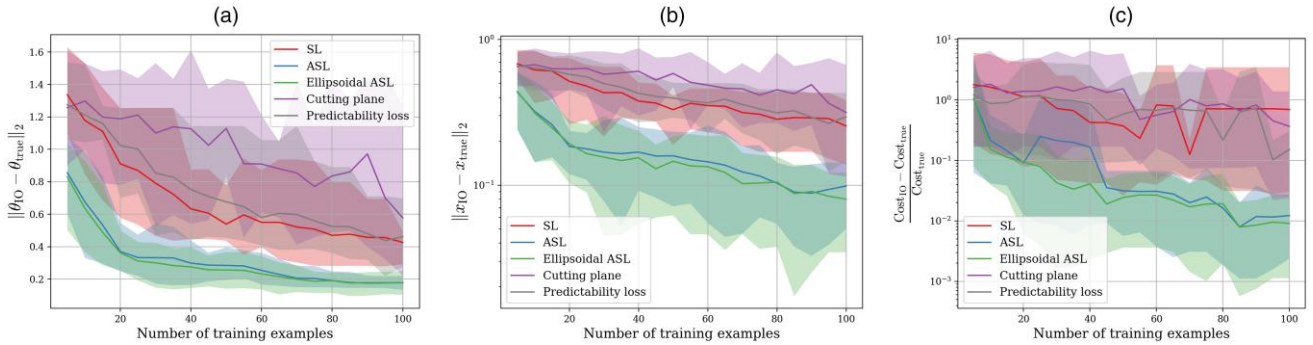
$$\begin{aligned} \min_{y, z} \quad & \langle q_y, y \rangle + \langle q_z, z \rangle \\ \text{s.t.} \quad & Ay + Bz \leq c \\ & 0 \leq y \leq 1, z \in \{0,1\}^v, \end{aligned} \tag{29}$$

where $\theta := (q_y, q_z) \in \mathbb{R}_+^{u+v}$ is the cost vector $A \in \mathbb{R}^{t \times u}$, $B \in \mathbb{R}^{t \times v}$, and $b \in \mathbb{R}^t$.

Data Generation. To generate training and test data, we sample the cost vector uniformly from $\{\theta \in \mathbb{R}^{u+v} : 0 \leq \theta \leq 1\}$, and we sample $\hat{A}$ uniformly from $\{A \in \mathbb{R}^{t \times u} : -1 \leq A \leq 0\}$, $\hat{B}$ uniformly from $\{B \in \mathbb{R}^{t \times v} : -1 \leq B \leq 0\}$, and $\hat{c}$ uniformly from $\{c \in \mathbb{R}^t : -2 \leq c \leq 0\}$. To make sure the problem instances are feasible, we checked if the sum of each row of $[\hat{A}\ \hat{B}]$ is smaller than the respective component of $\hat{c}$. Given a tuple $(\hat{A}, \hat{B}, \hat{c})$, we generate a response $\hat{x} = (\hat{y}, \hat{z})$ by solving Problem (29). To evaluate the IO approaches, we generate 10 random cost vectors θ_{true}. For each of these cost vectors, we generate a training data set $\hat{\mathcal{D}}_{\text{train}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ and a test data set $\hat{\mathcal{D}}_{\text{test}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$, with $N = 100$, $u = v = 6$, and $t = 4$.

IO Approaches. We compare six IO approaches for this problem.

Figure 4. (Color online) Out-of-Sample Results for the Inconsistent Data Scenario



Notes. (a) Difference between the true cost vector (θ_{true}) and the one learned using IO (θ_{IO}). (b) Average error between the decision generated by θ_{true} and θ_{IO}. (c) Relative difference between the cost of the decisions generated using θ_{true} and θ_{IO}.

- **SL:** We use (21) with no regularization $\mathcal{R}$ and no distance function (i.e., $\kappa = 0$ and $d(\hat{x}_i, x_i) = 0$). To avoid the trivial solution $\theta = 0$, we add the constraint $\|\theta\|_1 = 1$. We call this approach SL because it is the result of performing the reformulation steps of Corollary 4.1 using the suboptimality loss instead of the ASL.
- **ASL-yz:** We use (21) with $\theta = (q_y, q_z)$, $s = (A, B, c, 0)$ (A and c have to be augmented to account for the constraints $0 \le y \le 1$), $\mathbb{Z}(w) = \{0,1\}^v$, $Q_{yy} = 0$, $\phi_1(w,z) = 1$, $\phi_2(w,z) = z$, $\kappa = 0$, $\Theta = \{\theta \in \mathbb{R}^{u+v} : \theta \ge 0\}$, and $d_z(\hat{z}_i, z_i) = \|\hat{z}_i - z_i\|_2$.
- **ASL-z:** Same as ASL-yz but with $h_k = 0 \ \forall k \in [2u]$ (see discussion in the paragraph after Corollary 4.1).
- **Circumcenter:** We solve (27).
- **Cutting plane:** We use the cutting-plane algorithm of Wang (2009).
- **Predictability loss:** We use the predictability loss of Aswani et al. (2018) (Online Appendix EC.1).

Results. Figure 5 shows the results for this scenario. The discussions on the interpretations of the results of this section mirror the ones of Figure 3 from Section 6.1. Once again, the approach based on the ASL outperforms the other approaches on all three performance metrics. In particular, the ASL-z and ASL-yz show similar performance, with the ASL-yz being slightly better overall and when there are very few training examples. However, because the ASL-yz optimization problem has $2N$ times more constraints than the ASL-z optimization problem, it is much more costly to solve, as can be seen from the computational times in Table 3.

Table 2. Computational Time to Generate the Results of Figure 4

Approach	SL	ASL	Ellipsoidal ASL	Cutting plane	Predictability loss
Time (seconds)	230	241	351	440	6,192

6.4. Stochastic Approximate Mirror Descent

In this section, we numerically evaluate the SAMD algorithm proposed in Section 5.

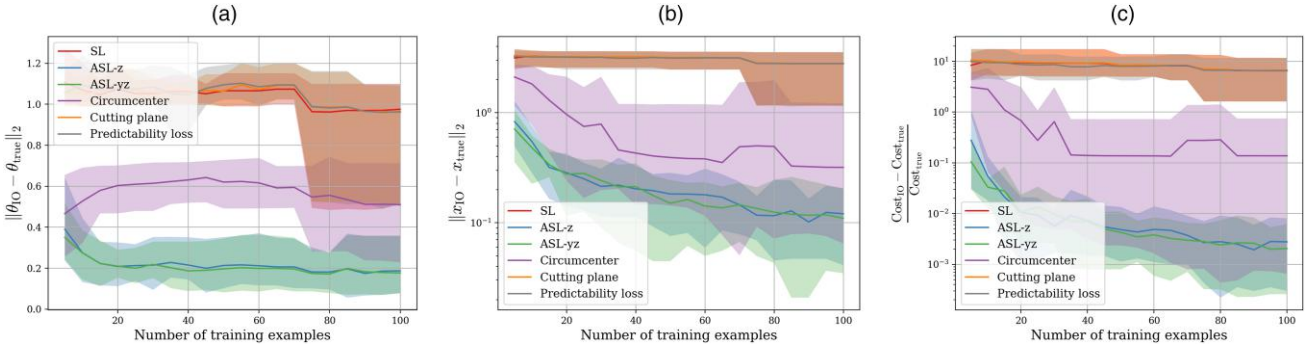
Decision Problem of the Expert. To generate its decisions, the expert solves the binary linear program (26), where $\theta \in \mathbb{R}^n_+$ is the cost vector, $A \in \mathbb{R}^{t \times n}$, $b \in \mathbb{R}^t$, and x is the decision vector. We can write this optimization problem as $\min_{x \in \mathbb{X}(s)} F(s,x)$ by defining $s := (A, b)$, $F(s,x) := \langle \theta, x \rangle$, and $\mathbb{X}(s) := \{x \in \{0,1\}^n : Ax \ge b\}$.

Data Generation. To generate training and test data, we sample cost vectors uniformly from $\{\theta \in \mathbb{R}^n : 0 \le \theta \le 1\}$, and we sample $\hat{A}$ uniformly from $\{A \in \mathbb{R}^{t \times n} : -1 \le A \le 0\}$ and $\hat{b}$ uniformly from $\left\{b \in \mathbb{R}^t : -\frac{n}{3} \le b \le 0\right\}$. To make sure that the problem instances are feasible, we checked if the sum of each row of $\hat{A}$ is larger than the respective component of $\hat{b}$. Given a signal $\hat{s}$, we generate a response $\hat{x}$ by solving (26) (i.e., $\hat{x} \in \arg\min_{x \in \mathbb{X}(\hat{s})} \langle \theta, x \rangle$). To evaluate the IO approaches, we generate 10 random cost vectors θ_{true}. For each of these cost vectors, we generate a training data set $\hat{\mathcal{D}}_{\text{train}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$ and a test data set $\hat{\mathcal{D}}_{\text{test}} = \{(\hat{s}_i, \hat{x}_i)\}_{i=1}^N$, with $N = 50$, $n = 20$, and $t = 15$.

IO Approaches. We tackle this IO problem using (14) with $\kappa = 0.01$, $\mathcal{R}(\theta) = \|\theta\|_1$, $\Theta = \{\theta \in \mathbb{R}^n : \theta \ge 0\}$, and the ASL with $s = (A, b)$, $\mathbb{X}(s) = \{x \in \{0,1\}^n : Ax \le b\}$, $\phi(s,x) = x$ and $d(\hat{x}, x) = \|x - \hat{x}\|_1$. When testing algorithms with exponentiated updates (i.e., mirror descent updates with $\omega(\theta) = \sum_{i=1}^n \theta_i \log(\theta_i)$), we solve the reformulation (23), with $\tilde{\kappa}$ chosen such that (14) and (23) have the same optimal solution. We compare eight algorithms, which result from all possible combinations of standard or mirror descent updates, deterministic or stochastic subgradients, and exact or approximate subgradients.

(i) **Subgradient method (SM):** Algorithm 5.1 with $\omega(\theta) = \frac{1}{2}\|\theta\|_2^2$ and exact subgradients computed using the entire data set.

Figure 5. (Color online) Out-of-Sample Results for the Mixed-Integer Feasible Set Scenario



Notes. (a) Difference between the true cost vector (θ_{true}) and the one learned using IO (θ_{IO}). (b) Average error between the decision generated by θ_{true} and θ_{IO}. (c) Relative difference between the cost of the decisions generated using θ_{true} and θ_{IO}.

(ii) **Mirror descent (MD)** with exponentiated updates: Algorithm 5.1 with $\omega(\theta) = \sum_{i=1}^{n} \theta_i \log(\theta_i)$ and exact subgradients computed using the entire data set.

(iii) **Stochastic subgradient method (SSM):** Algorithm 5.1 with $\omega(\theta) = \frac{1}{2}\|\theta\|_2^2$ and exact stochastic subgradients.

(iv) **Approximated subgradient method (ASM):** Algorithm 5.1 with $\omega(\theta) = \frac{1}{2}\|\theta\|_2^2$ and approximate subgradients computed using the entire data set.

(v) **Stochastic mirror descent (SMD)** with exponentiated updates: Algorithm 5.1 with $\omega(\theta) = \sum_{i=1}^{n} \theta_i \log(\theta_i)$ and exact stochastic subgradients.

(vi) **Approximate mirror descent (AMD)** with exponentiated updates: Algorithm 5.1 with $\omega(\theta) = \sum_{i=1}^{n} \theta_i \log(\theta_i)$ and approximate subgradients computed using the entire data set.

(vii) **Stochastic approximate subgradient method (SASM):** Algorithm 5.1 with $\omega(\theta) = \frac{1}{2}\|\theta\|_2^2$ and approximate stochastic subgradients.

(viii) **Stochastic approximate mirror descent (SAMD)** with exponentiated updates: Algorithm 5.1 with $\omega(\theta) = \sum_{i=1}^{n} \theta_i \log(\theta_i)$ and approximate stochastic subgradients.

For all algorithms, we use $\eta_t = 1/(\|\tilde{g}_{\varepsilon_t}(\theta_t)\|_* \sqrt{t})$. To compute approximate subgradients for the ASM and SAMD algorithms, we give the solver (in our case, Gurobi) a time limit of 0.03 seconds to solve the optimization problem in line 4 of Algorithm 5.1. If the solver is not able to find an optimal solution within this time limit, it returns the best feasible solution found.

Results. To compare the performance of the algorithms, we report their results in terms of running time instead of the number of iterations. Figure 6(a) shows the convergence of the proposed algorithms in terms of the training loss gap. More precisely, defining $f(\theta) := \kappa \mathcal{R}(\theta) + \frac{1}{N}\sum_{i=1}^{N} \ell_\theta(\hat{s}_i, \hat{x}_i)$, the training loss gap of some θ_t is defined as $f(\theta_t) - \min_{\theta \in \Theta} f(\theta)$. From this plot, it is clear that each modification of the standard subgradient method (i.e., mirror descent updates, stochastic subgradients, and approximate subgradients) contributes to improving the convergence speed of the algorithms, with the fastest convergence achieved by the combination of all of these improvements (i.e., the SAMD algorithm). In Figure 6(b), we show the convergence of the SAMD using different batch sizes to compute stochastic subgradients. That is, instead of using only one sampled example at each iteration (line 3 of Algorithm 5.1), we sample a batch of B examples and use this batch of data to compute the stochastic subgradient, which we call SAMD $-$ B. From Figure 6(b), we can see that even though smaller batches lead to more variance in the converge of the algorithm (as expected), they also lead to a faster empirical convergence for this experiment.

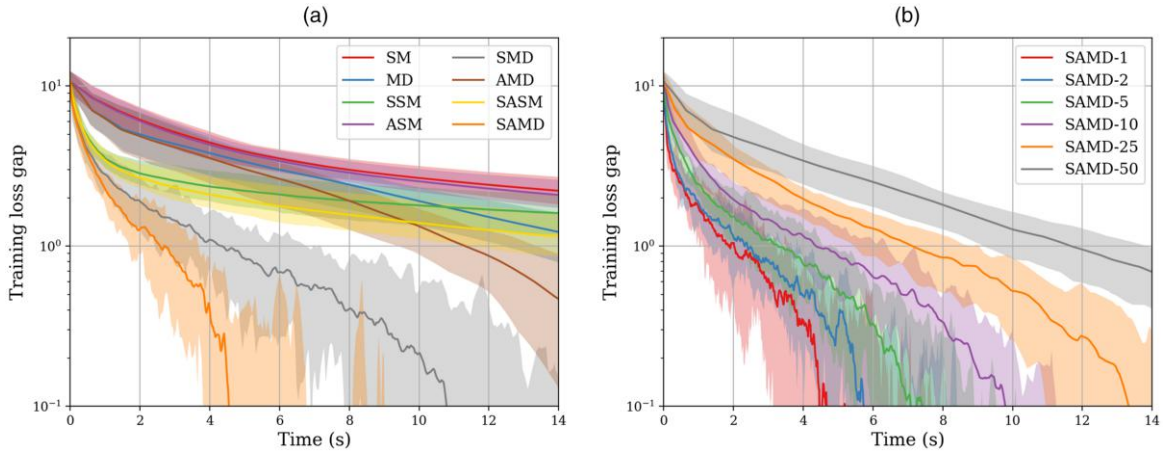
7. Further Discussions and Potential Applications

In this work, we present new approaches to tackle inverse optimization problems. Based on the geometry of the set of consistent cost vectors, we propose the use of an incenter vector of this set as the solution to the IO problem. Moreover, we propose a new loss function for IO problems: the augmented suboptimality loss. This loss can be interpreted as a relaxation of the incenter approach to tackle problems with inconsistent data. Moreover, this loss can be used to derive IO approaches tailored for problems with mixed-integer feasible sets and can be optimized directly using first-order methods. For the

Table 3. Computational Time to Generate the Results of Figure 5

Approach	SL	ASL-z	ASL-yz	Circumcenter	Cutting plane	Predictability loss
Time (seconds)	607	694	7,060	5,954	981	1,336

Figure 6. (Color online) Convergence Times of Different First-Order Optimization Algorithms and for Different Batch Sizes



Notes. (a) Convergence of the first-order optimization algorithms. (b) Convergence of the SAMD algorithms for different sizes of batches. AMD, approximate mirror descent; MD, mirror descent; SASM, stochastic approximate subgradient method; SM, subgradient method; SMD, stochastic mirror descent; SSM, stochastic subgradient method.

latter case, we propose new algorithms that combine stochastic, approximate, and mirror descent updates, all of which can be used to exploit the structure of IO losses. Finally, we numerically evaluate the approaches proposed in this paper and show that they can outperform state-of-the-art methods in a variety of scenarios.

The work presented in this paper opens several avenues for future research. For instance, one could try different ways to choose a vector from the set of consistent costs: for example, using ellipsoidal cones instead of circular cones. This idea was used in online IO algorithms (see Besbes et al. 2023, section 4.3), but perhaps it can also be leveraged for better empirical performance in the offline case. Also, one could try to adapt and further develop the incenter approaches presented in this paper to the online IO scenario, perhaps leveraging the geometry of the IO problem to prove tighter regret bounds in some specific scenarios, akin to Besbes et al. (2023). Another idea is to come up with different algorithms to tackle the optimization problems discussed in this paper (for instance, using cutting-plane/bundle methods), extending the related literature of structured prediction problems (Joachims et al. 2009, Wang 2009, Teo et al. 2010). Moreover, given the wealth of different first-order methods in the literature, we feel like designing specialized first-order (even proximal or second-order) algorithms tailored for inverse optimization problems is an underexplored research direction. Given the newly established bridge between IO and structured prediction problems (see Remark 4.2), we foresee that many interesting results can be achieved by combining and extending the literature of these two, mostly disjoint, communities. Finally, it would be interesting to extend the numerical experiments of this paper using real-world data: for instance, data from transportation/routing problems (e.g., as was recently done in Zattoni Scroccaro et al. (2024) in the context of the Amazon Challenge (Amazon.com, Inc. 2021)).

References

Ahmed S, Guan Y (2005) The inverse optimal value problem. *Math. Programming* 102:91–110.

Ahuja RK, Orlin JB (2001) Inverse optimization. *Oper. Res.* 49(5):771–783.

Akhtar SA, Kolarijani AS, Mohajerin Esfahani P (2021) Learning for control: An inverse optimization approach. *Proc. Amer. Control Conf., ACC 2021* (IEEE, Piscataway, NJ), 2193–2198.

Allen-Zhu Z, Orecchia L (2014) Linear coupling: An ultimate unification of gradient and mirror descent. Preprint, submitted July 6, https://arxiv.org/abs/1407.1537.

Amazon.com, Inc. (2021) Last-mile routing challenge team performance and leaderboard. Accessed August 26, 2024, https://routingchallenge.mit.edu/last-mile-routing-challenge-team-performance-and-leaderboard/.

Aswani A, Shen ZJ, Siddiq A (2018) Inverse optimization with noisy data. *Oper. Res.* 66(3):870–892.

Bärmann A, Pokutta S, Schneider O (2017) Emulating the expert: Inverse optimization through online learning. *Internat. Conf. Machine Learn.* (PMLR, New York), 400–410.

Bertsekas DP (2008) *Nonlinear Programming* (Athena Scientific, Belmont, MA).

Bertsekas DP (2015) *Convex Optimization Algorithms* (Athena Scientific, Belmont, MA).

Bertsimas D, Gupta V, Paschalidis IC (2012) Inverse optimization: A new perspective on the Black–Litterman model. *Oper. Res.* 60(6):1389–1403.

Bertsimas D, Gupta V, Paschalidis IC (2015) Data-driven estimation in equilibrium using inverse optimization. *Math. Programming* 153(2):595–633.

Besbes O, Fonseca Y, Lobel I (2023) Contextual inverse optimization: Offline and online learning. *Oper. Res.*, ePub ahead of print August 2, https://doi.org/10.1287/opre.2021.0369.

Bodur M, Chan TCY, Zhu IY (2022) Inverse mixed integer optimization: Polyhedral insights and trust region methods. *INFORMS J. Comput.* 34(3):1471–1488.

Boyd S, Vandenberghe L (2004) *Convex Optimization* (Cambridge University Press, Cambridge, UK).

Bubeck S (2015) *Convex Optimization: Algorithms and Complexity*, Foundations and Trends in Machine Learning (Now Publishers, Delft, Netherlands).

Burton D, Toint PL (1992) On an instance of the inverse shortest paths problem. *Math. Programming* 53:45–61.

Chan TCY, Lee T, Terekhov D (2019) Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Management Sci.* 65(3):1115–1135.

Chan TCY, Mahmood R, Zhu IY (2023) Inverse optimization: Theory and applications. *Oper. Res.*, ePub ahead of print December 19, https://doi.org/10.1287/opre.2022.0382.

Chan TCY, Craig T, Lee T, Sharpe MB (2014) Generalized inverse multiobjective optimization with application to cancer therapy. *Oper. Res.* 62(3):680–695.

Chan TCY, Eberg M, Forster K, Holloway C, Ieraci L, Shalaby Y, Yousefi N (2022) An inverse optimization approach to measuring clinical pathway concordance. *Management Sci.* 68(3):1882–1903.

Chen VX, Kılınç-Karzan F (2020) Online convex optimization perspective for learning from dynamically revealed preferences. Preprint, submitted August 24, https://arxiv.org/abs/2008.10460.

Chow JYJ, Recker WW (2012) Inverse optimization with endogenous arrival time constraints to calibrate the household activity pattern problem. *Transportation Res. Part B Methodological* 46(3):463–479.

Elmachtoub AN, Grigas P (2022) Smart "predict, then optimize." *Management Sci.* 68(1):9–26.

Faragó A, Szentesi Á, Szviatovszki B (2003) Inverse optimization in high-speed networks. *Discrete Appl. Math.* 129(1):83–98.

Ghobadi K, Mahmoudzadeh H (2021) Inferring linear feasible regions using inverse optimization. *Eur. J. Oper. Res.* 290(3):829–843.

Ghobadi K, Lee T, Mahmoudzadeh H, Terekhov D (2018) Robust inverse optimization. *Oper. Res. Lett.* 46(3):339–344.

Heuberger C (2004) Inverse combinatorial optimization: A survey on problems, methods, and results. *J. Combinatorial Optim.* 8:329–361.

Iyengar G, Kang W (2005) Inverse conic programming with applications. *Oper. Res. Lett.* 33(3):319–330.

Joachims T, Finley T, Yu CNJ (2009) Cutting-plane training of structural SVMs. *Machine Learn.* 77(1):27–59.

Juditsky A, Nemirovski A (2011) First order methods for nonsmooth convex large-scale optimization. I. General purpose methods. *Optim. Machine Learn.* 30(9):121–148.

Keshavarz A, Wang Y, Boyd S (2011) Imputing a convex objective function. *2011 IEEE Internat. Sympos. Intelligent Control* (IEEE, Piscataway, NJ), 613–619.

Lu H, Freund RM, Nesterov Y (2018) Relatively smooth convex optimization by first-order methods, and applications. *SIAM J. Optim.* 28(1):333–354.

Mohajerin Esfahani P, Shafieezadeh-Abadeh S, Hanasusanto GA, Kuhn D (2018) Data-driven inverse optimization with imperfect information. *Math. Programming* 167(1):191–234.

Nemirovski A (1996) Lecture notes: Interior point polynomial methods in convex programming. Accessed August 26, 2024, https://www2.isye.gatech.edu/~nemirovs/Lect_IPM.pdf.

Nemirovski A (2004) Prox-method with rate of convergence O(1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.* 15(1):229–251.

Nesterov Y (2007) Dual extrapolation and its applications to solving variational inequalities and related problems. *Math. Programming* 109(2–3):319–344.

Nowozin S, Lampert CH (2011) Structured learning and prediction in computer vision. *Foundations Trends Comput. Graphics Vision* 6(3–4):185–365.

Orabona F (2019) A modern introduction to online learning. Preprint, submitted December 31, https://arxiv.org/abs/1912.13213.

Ratliff ND, Bagnell JA, Zinkevich MA (2006) Maximum margin planning. *Proc. 23rd Internat. Conf. Machine Learn.* (ACM, New York), 729–736.

Ratliff ND, Bagnell JA, Zinkevich MA (2007) (Approximate) subgradient methods for structured prediction. *Proc. Eleventh Internat. Conf. Artificial Intelligence Statist.*, vol. 2 (PMLR, New York), 380–387.

Saez-Gallego J, Morales JM (2017) Short-term forecasting of price-responsive loads using inverse optimization. *IEEE Trans. Smart Grid* 9(5):4805–4814.

Schaefer AJ (2009) Inverse integer programming. *Optim. Lett.* 3:483–489.

Shor NZ (1985) *Minimization Methods for Non-Differentiable Functions*, Springer Series in Computational Mathematics (Springer, Cham, Switzerland).

Taskar B, Lacoste-Julien S, Jordan MI (2006) Structured prediction, dual extragradient and Bregman projections. *J. Machine Learn. Res.* 7(60):1627–1653.

Taskar B, Chatalbashev V, Koller D, Guestrin C (2005) Learning structured prediction models: A large margin approach. *Proc. 22nd Internat. Conf. Machine Learn.* (ACM, New York), 896–903.

Taşkesen B, Shafieezadeh-Abadeh S, Kuhn D (2023) Semi-discrete optimal transport: Hardness, regularization and numerical solution. *Math. Programming* 199:1033–1106.

Teo CH, Vishwanathan SVN, Smola A, Le QV (2010) Bundle methods for regularized risk minimization. *J. Machine Learn. Res.* 11(10):311–365.

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B Statist. Methodology* 58(1):267–288.

Tsochantaridis I, Joachims T, Hofmann T, Altun Y, Singer Y (2005) Large margin methods for structured and interdependent output variables. *J. Machine Learn. Res.* 6(9):1453–1484.

Wang L (2009) Cutting plane algorithms for the inverse mixed integer linear programming problem. *Oper. Res. Lett.* 37(2):114–116.

Zattoni Scroccaro P (2023) InvOpt: An open-source Python package to solve Inverse Optimization problems. https://github.com/pedroszattoni/invopt.

Zattoni Scroccaro P, van Beek P, Mohajerin Esfahani P, Atasoy B (2024) Inverse optimization for routing problems. *Transportation Sci.*, ePub ahead of print July 17, https://doi.org/10.1287/trsc.2023.0241.

Pedro Zattoni Scroccaro is a PhD researcher with the Delft University of Technology, Delft, Netherlands, and his research interests include inverse optimization and optimization algorithms, with applications to transportation systems.

Bilge Atasoy is an associate professor at the Delft University of Technology and is working toward adaptive transport and logistics systems. Her research lies at the intersection of operations research, behavioral modeling, and learning algorithms. She is the recipient of the European Research Council (ERC) Starting Grant and runs other national and international projects.

Peyman Mohajerin Esfahani is an associate professor at the Delft Center for Systems and Control. He was one of the three finalists for the Young Researcher Prize in Continuous Optimization awarded by the Mathematical Optimization Society in 2016 and a recipient of the 2016 George S. Axelby Outstanding Paper Award from the IEEE Control Systems Society. He received the European Research Council (ERC) Starting Grant and the INFORMS Frederick W. Lanchester Prize in 2020. He is the recipient of the 2022 European Control Award.