INVERSE PARAMETRIC OPTIMIZATION FOR LEARNING UTILITY FUNCTIONS FROM OPTIMAL AND SATISFICING DECISIONS

Elaheh Hosseiniiraj

A thesis

in The Department

of

Mechanical, Industrial & Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Industrial Engineering) at

Concordia University

Montréal, Québec, Canada

June 2021

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Elaheh Hosseiniiraj**
Entitled:	**Inverse Parametric Optimization For Learning Utility Functions From Optimal and Satisficing Decisions**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy(Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Chun Wang	Chair
Dr. Irène Abi-Zeid	External Examiner
Dr. Nizar Bouguila	External to Program
Dr. Ivan Contreras	Examiner
Dr. Onur Kuzgunkaya	Examiner
Dr. Daria Terekhov	Supervisor

Approved

Dr. Ivan Contreras, Graduate Program Director

July 21, 2021

Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Inverse Parametric Optimization For Learning Utility Functions From Optimal and Satisficing Decisions

Elaheh Hosseiniiraj, Ph.D.
Concordia University, 2021

Inverse optimization is a method to determine optimization model parameters from observed decisions. Despite being a learning method, inverse optimization is not part of a data scientist's toolkit in practice, especially as many general-purpose machine learning packages are widely available as an alternative. In this dissertation, we examine and remedy two aspects of inverse optimization that prevent it from becoming more used by practitioners. These aspects include the alternative-based approach in inverse optimization modeling and the assumption that observations should be optimal.

In the first part of the dissertation, we position inverse optimization as a learning method in analogy to supervised machine learning. The first part of this dissertation provides a starting point toward identifying the characteristics that make inverse optimization more efficient compared to general out-of-the-box supervised machine learning approaches, focusing on the problem of imputing the objective function of a parametric convex optimization problem.

The second part of this dissertation provides an attribute-based perspective to inverse optimization modeling. Inverse attribute-based optimization imputes the importance of the decision attributes that result in minimally suboptimal decisions instead of imputing the importance of decisions. This perspective expands the range of inverse optimization applicability. We demonstrate that it facilitates the application of inverse optimization in assortment optimization, where changing product selections is a defining feature and accurate predictions of demand are essential.

Finally, in the third part of the dissertation, we expand inverse parametric optimization to a more general setting where the assumption that the observations are optimal is relaxed to requiring only feasibility. The proposed inverse satisfaction method can deal with both feasible and minimally suboptimal solutions. We mathematically prove that the inverse satisfaction method provides statistically consistent estimates of the unknown parameters and can learn from both optimal and feasible decisions.

Acknowledgments

First, I would like to offer my sincere gratitude to my supervisor, Dr. Daria Terekhov, for her support, patience and for providing me with this unique and rewarding learning experience. This journey would not have been possible without Dr. Terekhov's constructive comments and supervision on our weekly meetings. Thank you for making these years the most growing time for me.

Also, a warm and kind thank goes to the member of the Data-Driven Operations Research Lab and Concordia University staff for creating a friendly and motivating environment.

I also would like to thank my dear friends Shima Ghanei, Bahar Katoozian, and Mozhde Hemmati for their never-ending support and for all those lunch idea exchanges and coffee breaks when I needed them the most.

Last but not least, I would like to thank my parents Mariyam and Ali, for always believing in me, supporting me in every decision I have made, and encouraging me to do my best. Also, a big thank goes to my husband, Masoud, who has always been the greatest support and my biggest comfort during all these years.

Contribution of Authors

This dissertation contains the material of three papers that are already submitted or are close to submission to peer reviewed journals. Chapter 3 is the manuscript of the paper "Comparing Inverse Optimization and Machine Learning Methods for Imputing a Convex Objective Function" which has been submitted to the *Journal of Operational Research Society* and is under revision. Chapter 4 is an extended version of the manuscript of the paper "Inverse Attribute-based Optimization with an Application in Assortment Optimization" which has been submitted to *International Transactions in Operational Research* which is currently under the second round of review. Chapter 5 is the manuscript of the paper "Inverse Satisfaction" which is currently in preparation and will be submitted by Fall 2021.

All three manuscripts are co-authored with Dr. Daria Terekhov, who established research guidelines and reviewed the papers before submission. The author of this thesis acted as the principal researcher with the corresponding duties such as the development of formulations and algorithms, the programming of solution methods and the analysis of computational results along with writing drafts of the papers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inverse problems involve finding the parameters that define a forward model given observations of the forward model solutions(s). Inverse problems are important when the solution(s) of a forward model is available but the model itself is not fully known —a common setting in numerous disciplines such as geology, medical imaging, and decision science (Tarantola, 2005). Inverse optimization (IO) is a class of inverse problems whose forward model is an optimization model. IO aims to impute missing optimization model parameters given observations of minimally sub-optimal observations ,i.e., observations of optimum or near optimum solutions of an optimization problem. To do so, inverse optimization uses some form of optimality conditions to leverage the prior knowledge about the data (i.e., the knowledge that the data was generated by an optimization process) and to ensure that the observations become minimally sub-optimal solutions of the imputed forward optimization problem (FOP). A review of inverse optimization methods is presented in Chapter 2.

In this dissertation, we study a particular type of inverse optimization problem, inverse parametric optimization. Inverse parametric optimization is the problem of learning unknown parameters of a parametric forward optimization problem. A parametric (forward) optimization problem (also referred to as a multi-parametric programming problem) is a family of optimization problems parametrized by an independent parameter; this problem involves finding a function that would map values of the independent parameter to optimal solutions (Hempel et al., 2014; Pistikopoulos et al., 2011). In this sense, inverse parametric optimization fits within the broader conceptual framework of "learning" which is the capability to acquire knowledge by extracting patterns from raw data (Goodfellow et al., 2016); we are acquiring knowledge about an optimization process that generated the data we observe. The parametric setting is especially advantageous in practice as it is capable of reflecting the situations where decisions depend on external parameters, e.g., customers make decisions regarding the number of items to buy by solving a utility optimization problem where the price of items is an external parameter.

1.1 Motivating Examples

Inverse optimization applications occur in settings where an optimization process generates decisions, decisions are observable, but the parameters of the decision generation problem are unknown or could vary. While early developments of inverse optimization methodologies assume that those decisions should be optimal, recently the inverse optimization methodology is expanded to settings where the decisions are near to optimal. Examples of the optimization process that generated data are automated decision support systems where a designed optimization problem generates decisions; experts that can usually make good decisions in their area of expertise; and agents/users whose decisions maximize their utility (or give them an acceptable level of utility). Given observations of decisions, inverse optimization can be used to tune the parameters of an automated decision system and find parameter values to make a decision optimal. For the experts and optimizing agents/users example, the decision making problem is usually vague as experts and users cannot explicitly reveal the value of some of their decision making model parameters. Therefore, inverse optimization can be used to impute such parameters from observations of their (near to) optimal decisions. Other applications of inverse optimization studied in the literature include shortest path, assignment and minimum cut problems (Ahuja and Orlin, 2001), health related decision making where decisions by health experts are available but the expert decision making parameters are unavailable (Chan et al., 2019), finance (Bertsimas et al., 2012; Beil and Wein, 2003) and market bidding in electricity trading (Saez-Gallego and Morales, 2017).

This dissertation is mainly motivated by the problem of learning the decision making parameters of optimizing agents/users. Learning such parameters enables a third party (e.g., firms and recommender systems) to predict future decisions. Predicting future decisions of users has significant benefits for service providers as they can improve their services and offerings by tailoring them to the user's interest. It also benefits the users as they can get customized services. The inverse optimization models we propose leverage the idea of attributes from multi-attribute decision-making (MADM) and this dissertation provide frameworks and methods to impute the weights of decision attributes.

1.2 The Practice of Inverse Optimization

In this dissertation, we position inverse optimization as a learning method and aim to address two main questions: 1) when, or even whether, investing in developing IO methods is worthwhile? 2) how can we make inverse optimization more practical as a learning method?

To address the first question, we start by establishing an analogy between inverse optimization and machine learning (ML). Viewing an inverse (parametric) optimization problem as a learning problem brings up topics that have previously not been explored in the literature and are of practical interest. The first one is the comparison of ML and IO performance on data generated by an optimization process. The second topic is identifying the characteristics that would make a problem challenging for classic machine learning methods and require a more specialized methods of inverse optimization. In general, problem difficulty from a learning perspective is not well understood for problems where data is generated by an optimization process. It is unclear exactly which problems can be solved with machine learning tools and which problems need inverse optimization methods. We experimentally address these questions and bring together ideas and concepts from machine learning and operations research.

To address the second question, we noticed that there are settings in practice where previously-developed inverse optimization models are not fully applicable. One such setting is when we use inverse optimization to fit an optimization model that needs to predict the future decisions of optimizing agents. The majority of inverse optimization literature implicitly assumes an "alternative-based" approach, i.e., they impute objective function coefficients corresponding to a set of previously seen decisions which makes inverse optimization incapable of predicting future *unseen* decisions. We propose an "attribute-based" perspective in forward and inverse modeling to address this limitation of inverse optimization. Instead of imputing the weights of decisions, attribute-based IO imputes the weights of attributes of those decisions; thus, when the set of decisions changes, weights imputed from past data can be utilized to parameterize an optimization problem given a new set of alternatives, as long as both past and current alternatives can be expressed in terms of the same attributes.

Another setting in which inverse optimization cannot be applied is when the assumption of the optimality of observations is not satisfied. This setting is common in practice as many non-expert human agents cannot make optimal or near to optimal decisions due to cognitive and time limitations and opt to make only satisficing decisions (Simon, 1979). Even though an optimization process is not involved in such a setting, decisions are usually subject to constraints and the decision making problem is a feasibility problem instead of an optimization problem. To make inverse optimization applicable for such a setting, we relax the optimality assumption and propose an inverse satisfaction method. Inverse satisfaction expands the applicability of inverse optimization to a broader set of problems in practice.

This dissertation contributes to building a bridge between IO methodology and IO practice. We extend the applicability of inverse optimization to more general settings by demonstrating characteristics where inverse optimization could be a strong alternative to classical machine learning methods, applying inverse optimization to attribute-based formulations, and developing inverse optimization-like formulations for satisficing decisions.

1.3 Contributions

The main contributions of this dissertation are as follows:

1. We position inverse parametric optimization as a learning problem and experimentally compare the performance of a classic IO method with three out-of-the-box machine learning (ML) methods, namely Gaussian process regression, random forest, and support vector regression.

2. We identify the characteristics that would make a problem challenging for inverse optimization and machine learning methods and conduct experiments on random parametric optimization problems (POPs). To the best of our knowledge, no previous papers have compared IO and ML methods on this class of problems. Our experiments with randomly-generated POPs demonstrate that the choice of an ML or IO approach should depend on (i) the size of the training set, (ii) the nature of the dependence of the optimization problem on external parameters, (iii) the level of confidence with regards to the correctness optimization prior, (iv) the number of critical regions in the solution space of the POP.

3. We propose a new perspective on inverse optimization based on attributes. Inverse attribute-based optimization aims to impute the weights of attributes that lead to an optimal decision instead of imputing the decision's weight. This perspective expands the range of IO applicability and facilitates the application of IO in assortment optimization, where changing product selections is a defining feature, and accurate predictions of demand are essential.

4. We relax the assumption of near-optimality of observations in inverse optimization and propose an inverse satisfaction method to jointly estimate the weight of the satisficer's decision attributes and their acceptable threshold for decisions. We experimentally show that our proposed model can learn from both optimizer and satisficer decisions compared to previous models in inverse optimization literature that are capable of learning only from optimizers.

5. We mathematically prove that the proposed inverse satisfaction model is estimation consistent, meaning that with enough number of observations, the estimation of weights and acceptable threshold converge to their true values.

1.4 Outline of the Dissertation

The structure of the dissertation is as follows. In Chapter 2 we review the literature of inverse optimization methods and related research in other disciplines.

In Chapter 3 we discuss the positioning of inverse parametric problems as a learning problem. In this chapter, we focus on the problem of imputing the objective function coefficients of a parametric convex optimization problem. We compare the predictive performance of three standard supervised machine learning algorithms, namely random forest (RF), support vector regression (SVR), and Gaussian process regression (GP) to the performance of the IO model of Keshavarz et al. (2011). To identify the strengths and weaknesses of machine learning and inverse optimization methods, we conduct experiments with multiple settings mainly based on the complexity of the data generation process and the number of observations.

In Chapter 4 we address one of the limitations of inverse optimization in practice which is its alternative-based approach in modeling which limits the applicability of inverse optimization for prediction purposes. In this chapter, we propose a new perspective on modeling based on decision attributes. We study the problem of imputing the weights of attributes instead of imputing the coefficients of the decision. The inverse attribute-based perspective enables inverse optimization to predict decisions that are not present in the training set as long as the set of decision attributes remains unchanged.

In Chapter 5, we further expand the applicability of inverse optimization. We relax one key but limiting assumption in inverse optimization, which is the assumption that the observations are optimal. In many situations, decisions are only acceptable decisions. However, the decision making process is usually subject to some constraints. This chapter proposes an inverse model for imputing the unknown parameters of a feasibility problem, given observations of its feasible solutions. We call this method inverse satisfaction and mathematically prove that the proposed inverse satisfaction method is statistically data consistent, i.e., its estimate of the feasibility model parameters converges to the actual values of the parameters in probability.

Finally, Chapter 6 concludes the dissertation with a summary of contributions and future work.

Chapter 2

Literature Review

This chapter briefly reviews the inverse optimization preliminaries and then reviews the inverse optimization developments in methodology and applications. At the end of the chapter, we address how this dissertation contributes to the current inverse optimization literature.

2.1 Preliminaries

Inverse optimization models vary based on their choice of the loss function. In the literature, loss functions are generally defined over either the decision space or objective space (Babier et al., 2021). However, in this review, we consider another classification of loss functions. We classify loss functions either on parameter space, decision space or objective space. We define the forward optimization problem as:

$$\mathbf{FOP}(\mathbf{c}) : \underset{\mathbf{x} \in \mathcal{S}}{\text{minimize}} \quad f(\mathbf{x}, \mathbf{c}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ are the decision variables, $\mathbf{c} \in \mathbb{R}^n$ are the coefficients of the objective function f, and $\mathcal{S}$ is the feasible region of the forward problem. Given an observed solution of problem (1) denoted $\hat{\mathbf{x}}$, we define the three classes of inverse optimization as follows.

Parameter space:

The corresponding classic inverse optimization problem, i.e., the problem of finding missing objective function coefficients, can be defined in parameter space as:

$$\min_{\mathbf{c}} \quad \left\{ \mathcal{L}(\mathbf{c}, \mathbf{d}) \quad \middle| \ \hat{\mathbf{x}} \in \arg\min_{\mathbf{x} \in \mathcal{X}} \ f(\mathbf{x}, \mathbf{c}) \right\} \tag{2}$$

where $\mathbf{d}$ is an estimate of unknown vector $\mathbf{c}$ and $\mathcal{L}$ is a loss function that measures the error between a known and an imputed parameter vector. As there might be multiple feasible $\mathbf{c}$ vectors, the function $\mathcal{L}$ is used to find a single solution closest to the given estimate of $\mathbf{c}$. Various forms of loss function $\mathcal{L}$ have been studied in the literature. Ahuja and Orlin (2001) use the L_1 and L_∞ norm. They show that if a forward problem is polynomially solvable for each linear cost function, then the corresponding inverse problems under L_1 and L_∞ are polynomially solvable too. Iyengar and Kang (2005) and Burton and Toint (1992) consider L_2 norm as the loss function. Hamming distance is also considered in some studies (e.g., Duin and Volgenant (2006)). While in some studies prior information about the value of $\mathbf{c}$ is available, some others assume situations when there is no prior information on the missing parameters (e.g., Chan et al. (2019)). In this case, $\mathbf{c}$ does not need to be close to a given prior but only has to be imputed to make the observation optimal for the forward solution.

Decision space:

Representation (2) of the inverse optimization problem denotes the early developments of the literature when the loss functions are mostly defined on the parameter space. Other studies define the loss function over the decision variables (e.g., Aswani et al. (2018, 2019b); Esfahani et al. (2018)). The corresponding inverse optimization problem with such loss function can be defined as:

$$\underset{\mathbf{c}}{\text{minimize}} \quad \left\{ \mathcal{L}(\hat{\mathbf{x}}, \mathbf{x}^{pred}) \;\middle|\; \mathbf{x}^{pred} \in \underset{\mathbf{x}\in\mathcal{X}}{\arg\min} \; f(\mathbf{x}, \mathbf{c}) \right\} \tag{3}$$

where $\mathbf{x}^{pred}$ is an optimal solution to the fitted forward optimization model and $\mathcal{L}$ is any given loss between $\hat{\mathbf{x}}$ and $\mathbf{x}^{pred}$. Some examples of the loss function in the literature are the average discrepancy between the measured data and the estimated data (Aswani et al., 2018) and the mismatch between the predicted and true optimal decisions (Esfahani et al., 2018) such as a 2-norm as:

$$\mathcal{L}(\hat{\mathbf{x}}, \mathbf{x}^{pred}) = \|\hat{\mathbf{x}} - \mathbf{x}^{pred}\|_2. \tag{4}$$

Objective space:

The loss function of inverse optimization can also measure the loss in objective space. In this case, formulation (3) can be modified in a way that the loss function minimizes some form of optimality condition. One commonly used loss in objective function space is the absolute duality gap which measures the objective function values incurred by observations $\hat{\mathbf{x}}$ and $\mathbf{x}^{pred}$ (Saez-Gallego and Morales, 2017; Chan et al., 2019). Another loss function used in objective space is the relative duality gap that measures the relative value instead of the absolute value of loss (Chan et al., 2014a; Babier et al., 2018; Chan et al., 2019). Other loss functions in objective space have been studied that consider the residulas of some form of optimality conditions. For instance, Keshavarz et al. (2011) and Bertsimas et al. (2015) propose a loss function on residuals of optimality conditions including KKT conditions and variational inequality conditions, respectively.

2.2 Review of the Literature

Inverse optimization models in the literature can be generally categorized based on (i) number of observations: single or multiple observations, (ii) observations setting with regard to noise: noisy or noise-free, (iii) inverse continuous or inverse integer programs and (iv) non-parametric or parametric setting.

Early developments of inverse optimization are mainly focused on single and noise-free observations (i.e., observations for which there exist values of missing parameters that can make them optimal solutions of the forward optimization problem (Ahuja and Orlin, 2001; Iyengar and Kang, 2005; Burton and Toint, 1992)). However, recent studies in inverse optimization incorporate noise in the observations (Keshavarz et al., 2011; Aswani et al., 2018; Esfahani et al., 2018).

While most of the studies in inverse optimization are continuous programs, some investigate the integer setting in inverse optimization too (e.g., Schaefer (2009); Wang (2009); Bulut and Ralphs (2015); Moghaddass and Terekhov (2020)).

Until recently, the majority of the literature centered around the non-parametric setting. However, recently the inverse parametric optimization has drawn more attention in the literature (e.g., (Keshavarz et al., 2011; Aswani et al., 2018; Saez-Gallego and Morales, 2017)).

This dissertation focuses on the inverse continuous optimization problems with multiple noisy observations in a parametric setting. Hence, below we provide a review of the literature in this setting specifically.

2.2.1 Inverse Continuous Optimization with Noisy Observations

In inverse optimization models with a noise-free setting, it is assumed that there exist parameter values that make a given solution optimal; however, in practice it might be impossible to find such parameter values. This happens because of several reasons such as the error in data collecting, deviation of decision makers from optimality –phenomena referred to as bounded rationality–and the mismatch between the mathematical model and the actual decision making process (we refer readers to Esfahani et al. (2018) for a review on the possible sources of noise in inverse optimization). Accordingly, new inverse optimization studies have been done with noisy data and non-optimal candidate solutions to render meaningful solutions for noisy observations and make them ϵ_optimal. In this family of inverse optimization problems, usually a loss function measures the slack of some form of optimality conditions such as KKT conditions (Keshavarz et al., 2011), variational inequality (Bertsimas et al., 2015) and strong duality (Chan et al., 2019) and convexity for both the feasible region of the parametrized optimization model and the objective function is inherently assumed. Recently other studies have been done dealing with more complex systems with non-convex decision domain that only rely on access to an

LP oracle using learning algorithms with no convexity assumptions (Bärmann et al., 2017).

While most of the noise-free setting papers have primarily focused on single point observation problems, both single point and multiple point settings have been considered in noisy settings. Some studies consider a single noisy observation which is feasible (e.g., Keshavarz et al. (2011); Chan et al. (2019); Bertsimas et al. (2012)) and some consider both feasible and infeasible observations (Boutilier et al. (2015); Chan et al. (2014a)) to the forward problem. Inverse optimization with noisy data has been mostly studied with multiple observations of the solutions of forward optimization problem (Troutt (1995); Troutt et al. (2005); Keshavarz et al. (2011); Aswani et al. (2018); Bertsimas et al. (2012); Esfahani et al. (2018)).

In most inverse optimization literature with noisy observations, even though observations are not candidate optimal, they are assumed to be a feasible point for the forward problem. Chan et al. (2014a) propose a generalized inverse optimization model in which observations could be on the boundary, inside the feasible region of the forward problem or even be an infeasible point in a multiobjective linear optimization context.

Recently a few studies have been done using robust and inverse optimization concepts either to solve robust optimization problems with a perspective of inverse optimization (Bertsimas et al. (2012); Chassein and Goerigk (2018)) or to use robust optimization [1] approaches in the course of solving inverse optimization problems (Esfahani et al., 2018; Ghobadi et al., 2018).

[1]Robust optimization is an area in operations research which addresses optimization problems with uncertain parameters (Bertsimas and Sim, 2004).

2.2.2 Inverse Parametric Optimization

This dissertation is based on the idea that the parametric setting of inverse optimization, i.e., learning model parameters of a parametric optimization problem (POP), can be viewed as a learning problem where the goal is to learn optimization model parameters from a sequence of external parameters and their corresponding parametric optimal decisions. The inverse parametric optimization problem has been studied by many authors (Keshavarz et al., 2011; Chow and Recker, 2012; Aswani et al., 2018; Esfahani et al., 2018; Saez-Gallego and Morales, 2017; Kovács, 2019; Tan et al., 2019, 2020).

Inverse parametric optimization was initially studied by Keshavarz et al. (2011) where an inverse optimization model was proposed to learn a convex objective function of a parametric optimization problem from pairs of external parameter and their corresponding optimal decisions. The parametric setting has been studied in the literature to learn from optimizing agent's decisions, i.e., optimal decisions with response to an external parameter in the environment (Chow and Recker, 2012; Aswani et al., 2018; Esfahani et al., 2018; Saez-Gallego and Morales, 2017; Kovács, 2019). Saez-Gallego and Morales (2017) propose an inverse parametric method to learn the electricity marginal utility parameters and the power consumption limits from the consumer's electricity demand parametrized by the electricity price and estimate their future electricity load under different levels of price. Aswani et al. (2018) propose an inverse parametric optimization method to learn an optimizing user utility function from decisions parametrized by temperature with application in energy and decisions. The similar problem of learning optimizing agent's utility is studied by Esfahani et al. (2018); in this study decisions are parametrized by price in the context of consumer behaviour. Recently, Tan et al. (2019, 2020) have studied the methodology and modeling aspects of inverse parametric optimization. They propose a deep learning method based on back propagation for learning parametric optimization problems (Tan et al., 2019) and a gradient-based framework for learning parametric linear programs from optimal decisions (Tan et al., 2020).

2.3 Connection to Other Disciplines

Inverse optimization as a problem of imputing model parameters from observations can be interpreted as a learning method. More specifically, inverse parametric optimization can be seen as a from of supervised learning method of machine learning literature (e.g., random forest and decision tree (Breiman, 1996, 2001), support vector regression (Drucker et al., 1997) and Gaussian process regression (Rasmussen, 2004)) [2] where the external parameters of the forward optimization problem are the input (independent variable, feature) and the observations of the decisions are output (dependent variable, target). More related work in the machine learning literature includes, most notably, structured prediction (Taskar et al., 2005). Despite this related machine learning-based work, a comparison of classic machine learning methods on general parametric inverse optimization problems has not previously been done in the literature.

From the perspective of learning from optimal decisions, inverse optimization has also connection to the literature around the utility elicitation problem (Keeney et al., 1993; Mármol et al., 1998; Boutilier, 2002; Chajewska et al., 2000; Abdellaoui, 2000) as the observations in this problem are assumed to be utility-optimizing. However, until recently, there has been little interest in the problem of finding preferences of users that solve constrained optimization problems and incorporating the prior

[2] We refer the readers to Section 4.5.1 for the definition of these methods with more detail.

knowledge that the user's decisions are solutions of an optimization problem. Inverse optimization as a method for imputing optimization model parameters is capable of using the prior knowledge that the user's decisions are obtained from solving an optimization problem.

Modeling user's decisions in the literature is usually represented as a multi-attribute decision making problem. Given that decisions are associated with some attributes, inverse optimization can be used to find the weight of attributes when observations of user's decisions are available. Though MADM literature has been more focused on finding the decisions given the attribute weights, some studies proposed methods to assign weights given observations of the decisions (Stillwell et al., 1981). From this perspective, inverse optimization can be an alternate to such weight assignment methods and solve the inverse problem of an MADM problem under the assumption of optimizing decision-makers.

2.4 Contributions of This Dissertation to the Literature

The inverse optimization methodology studied in this dissertation is similar to of Keshavarz et al. (2011); Aswani et al. (2018); Esfahani et al. (2018) given that they also studied inverse optimization in the parametric setting considering multiple noisy observations. In this dissertation, we first position the inverse parametric optimization problem in general as a form of learning. Though previous works have established an analogy between inverse optimization and regression (Chan et al., 2019), and some provide a comparison of an IO approach to several ML algorithms in the context of a particular application (Fernández-Blanco et al., 2019; Aswani et al., 2019a); comparing the performance of supervised machine learning methods with inverse parametric optimization for general problems such as POPs have not been previously addressed in the literature. In this dissertation, we identify the characteristics of a parametric optimization data/decision generation process that make the task of learning challenging for inverse optimization and machine learning methods. The problem characteristics studied in this dissertation can contribute valuable insights to the practitioners on why, whether and when they should opt for using a more specialized learning method such as inverse optimization instead of conventional machine learning methods.

Motivated by the practice of machine learning, this dissertation also proposes a new perspective in forward and inverse optimization modeling based on attributes that can further expand the applicability of inverse optimization. As mentioned earlier, inverse parametric optimization can be analogous to supervised machine learning (i.e., observable parameters to be considered as features and decisions as targets), therefore inverse parametric optimization can be used within a prediction framework to predict future decisions based on observations of parameters. However, to the best of our knowledge, all the studies in the literature have an "alternative-based" approach in inverse modeling (e.g., Keshavarz et al. (2011); Bertsimas et al. (2015); Aswani et al. (2018)), which limits the applicability of inverse optimization in settings when the set of decisions are subject to change with regard to the external parameters. This dissertation provides a new perspective that makes inverse parametric optimization more fitting for the parametric prediction problems.

One key assumption in inverse optimization is the optimality of the observations.

Even though recent studies relax the optimality assumption and incorporate noise in the observations (e.g., Chan et al. (2019)), the observations are still assumed to be $\epsilon-$optimal for the forward problem. However, the optimality assumption is not always guaranteed in many settings, especially where decision makers suffer from bounded rationality and make decisions to reach an acceptable level of utility instead of optimizing it (Simon, 1979). To further expand the applicability of inverse optimization, we relax the optimality assumption to only feasibility. We propose an inverse satisfaction model for imputing a feasibility problem's parameters, given observations of feasible solutions. To the best of our knowledge, this setting has not been studied before in the literature. Similar to the statistical inference perspective previously provided by Aswani et al. (2018) for minimally suboptimal observations, we also prove that the proposed inverse satisfaction model is statistically consistent in estimating the unknown parameters of the forward feasibility problem given feasible observations.

Chapter 3

Comparing Inverse Optimization and Machine Learning Methods for Imputing a Convex Objective Function

Inverse optimization aims to determine optimization model parameters from observed decisions. However, IO is not part of a data scientist's toolkit in practice, especially as many general-purpose machine learning packages are widely available as an alternative. When encountering IO, practitioners face the question of when, or even whether, investing in developing IO methods is worthwhile. This chapter provides a starting point toward answering these questions, focusing on the problem of imputing the objective function of a parametric convex optimization problem. We compare the predictive performance of three standard supervised machine learning (ML) algorithms (random forest, support vector regression and Gaussian process regression) to the performance of the IO model of Keshavarz et al. (2011). While the IO literature focuses on the development of methods tailored to particular problem classes, our goal is to evaluate general 'out-of-the-box' approaches. Our experiments suggest that determining whether to use an ML or IO approach requires considering (i) the training set size, (ii) the dependence of the optimization problem on external parameters, (iii) the level of confidence with regards to the correctness of the optimization prior, and (iv) the number of critical regions in the solution space.

3.1 Introduction

Inverse optimization imputes missing optimization model parameters from data that represents minimally sub-optimal solutions of that unknown optimization problem (e.g., past decisions of an optimizing agent). In the academic literature, IO has been shown successful in a variety of problems, including medical decision making (Chan et al., 2014a), electricity demand forecasting (Saez-Gallego and Morales, 2017), the household activity pattern problem (Chow and Recker, 2012) and economic lot-sizing (Egri et al., 2014). However, IO is rarely used by practitioners.

Previous work has established an analogy between IO and regression (Chan et al., 2019), and provided a statistical inference perspective on IO (Aswani et al., 2018). More generally, one can notice that *imputing parameter values from data* can be viewed as a machine learning problem (Tan et al., 2019), a perspective that we adopt and explore in this chapter. Viewing an inverse (parametric) optimization problem as a learning problem raises questions that have previously not been explored in the literature and that are of practical interest. First, how well would classic machine learning methods perform on such problems? Second, what characteristics would make a problem challenging for classic machine learning methods and would require the more specialized methods of IO? Our interactions with data science teams of two large companies have led us to believe that answering these questions is essential if a data scientist wants to consider adding IO to their toolkit. In this study, we have used synthetic data by generating instances of the forward optimization problem. Using synthetic data give us more control on generating instances with certain properties that are important in the postulated hypotheses about the behaviour of the inverse optimization method which helps us to assess these hypotheses.

While most of the IO literature develops specialized approaches, our goal is to compare fairly general methods, as we believe these are more likely to be used by practitioners. After establishing our perspective on inverse parametric optimization as a learning problem, we experimentally compare the performance of a classic IO method with three out-of-the-box machine learning (ML) methods, namely Gaussian process regression, random forest and support vector regression. We use the IO method of Keshavarz et al. (2011) as it is well suited and easy to apply to inverse parametric convex optimization problems. We show that the chosen ML methods can indeed perform well on a problem that has been well-studied in the IO literature. To identify the characteristics that would make an IO problem challenging for ML (highlighting the need for sophisticated IO models), we conduct experiments on random parametric optimization problems (POPs) generated using the parametric optimization toolbox of Oberdieck et al. (2016). To the best of our knowledge, no previous papers have compared IO and ML methods on this class of problems. Our experiments with randomly-generated POPs demonstrate that the choice of an ML or IO approach should depend on (i) the size of the training set, (ii) the nature of the dependence of the optimization problem on external parameters, (iii) the level of confidence with regards to the correctness optimization prior, (iv) the number of critical regions in the solution space of the POP.

3.2 Background and Related Work

To define an IO problem, we first need to postulate a *forward* optimization problem, i.e., the problem whose solutions, perhaps corrupted by noise, we observe. The initial focus of the IO literature was to impute coefficients of forward optimization problems that do not have any dependence on an external parameter (non-parametric). Burton and Toint (1992, 1994) study the inverse shortest path problem in which the goal is to find a minimal perturbation of the arc costs to make an observed set of paths optimal. Given an observed value for decision variables $\mathbf{x}$, Ahuja and Orlin (2001) find a $\mathbf{c}$ vector of minimal distance to a known vector $\hat{\mathbf{c}}$ that would make $\mathbf{x}$ optimal. Chan et al. (2014a) and Chan et al. (2019) study the inverse linear programming problem when the observed data is noisy and no prior $\hat{\mathbf{c}}$ is given. Tavaslıoğlu et al. (2018) characterize the inverse feasible region: the set of objectives that would make a given set of feasible solutions to a linear program optimal. Shahmoradi and Lee (2020) introduce a way to measure how sensitive the set of optimal cost vectors is to changes in a given data set.

A parametric optimization problem (also known as a multi-parametric programming problem) is a family of optimization problems parametrized by an independent parameter; the forward parametric problem involves finding a function that would map values of the independent parameter to optimal solutions (Pistikopoulos et al., 2011). As an example, consider the following parameteric linear program with independent parameters (u_1, u_2): $\min(c_1 + c_2u_1)x_1 + (c_3 + c_4u_2)x_2$ s.t. $a_1x_1 + a_2x_2 \geq b_1$, where instantiating u_1 and u_2 leads to a standard (non-parametric) optimization problem. There is substantial interest in the inverse parametric optimization problem (Keshavarz et al., 2011; Chow and Recker, 2012; Aswani et al., 2018; Esfahani et al., 2018; Saez-Gallego and Morales, 2017; Kovács, 2019; Tan et al., 2019, 2020). In such a problem, pairs $(\mathbf{u}_k, \mathbf{x}_k)$ where $\mathbf{u}_k$ is an independent parameter for observation k and $\mathbf{x}_k$ is an observed value for the decision variables $\mathbf{x}$ are given; the goal is to impute objective function and/or constraint coefficients that would make the observed points optimal or near-optimal solutions of the given parametric forward optimization problem.

Classic IO methods (Burton and Toint, 1992; Ahuja and Orlin, 2001; Keshavarz et al., 2011) rely on some form of optimality conditions, such as the Karush-Kuhn-Tucker (KKT) conditions. More recently, methods based on a combination of ML and IO have emerged (Tan et al., 2019, 2020; Babier et al., 2021). Aswani et al. (2019a) and Fernández-Blanco et al. (2019) include comparison of an IO approach to several ML algorithms in the context of a particular application. Related work in the ML literature includes, most notably, structured prediction (Taskar et al., 2005). Despite this related ML-based work, a comparison of classic methods on general parametric IO problems has not previously been done and the questions posed in the introduction have not been answered.

3.3 Problem Definition

We study the inverse of *parametric optimization problems* (POPs). POPs are mathematical programs where the objective function and constraints are functions of one or multiple external parameters (Pistikopoulos et al., 2011; Oberdieck et al., 2016). The inverse of POPs are inverse parametric optimization problems which are 'compatible' with an ML perspective (specifically supervised learning) as they use pairs of external parameters (features in machine learning literature) and decisions (targets in machine learning literature), unlike non-parametric optimization problems where the training data consists only of near optimal decisions. The *forward* parametric optimization problem (FOP) of interest to us is:

$$\begin{aligned} \mathbf{FOP}(\mathbf{u},\mathbf{c}) : \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{u},\mathbf{x},\mathbf{c}) \\ \text{s.t.} \quad & g(\mathbf{u},\mathbf{x}) \leq \mathbf{0}, \end{aligned} \tag{5}$$

where $\mathbf{x} \in \mathbb{R}^n$ are the decision variables, $\mathbf{u} \in \mathbb{R}^v$ are the independent parameters ('features' in ML parlance), and $\mathbf{c} \in \mathbb{R}^n$ are the coefficients of the objective function f. We assume f and g are differentiable and convex in $\mathbf{x}$ for each value of $\mathbf{u}$. We let $\mathcal{J}$ be the index set of variables with $|\mathcal{J}| = n$, and $\mathcal{I}$ be the index set of constraints with $|\mathcal{I}| = m$; we let $\mathbf{0}$ denote the column vector of zeros of the required dimension.

In an *inverse* parametric optimization problem, observations of pairs $\mathcal{D} = \{(\hat{\mathbf{x}}_1, \hat{\mathbf{u}}_1), \ldots, (\hat{\mathbf{x}}_K, \hat{\mathbf{u}}_K)\}$ are given; the goal is to find a $\mathbf{c}$ that would make $\hat{\mathbf{x}}_k$ as close to optimal as possible for the corresponding $\mathbf{FOP}(\hat{\mathbf{u}}_k, \mathbf{c})$, for each $k \in \mathcal{K} =$

$\{1, \ldots, K\}$:

$$\underset{\mathbf{c}}{\text{minimize}} \quad \left\{ \sum_{k \in \mathcal{K}} \mathcal{L}(\hat{\mathbf{x}}_k, \mathbf{x}_k^{pred}) \;\middle|\; \mathbf{x}_k^{pred} \in \underset{\mathbf{x} \in \mathcal{X}_{\hat{\mathbf{u}}_k}}{\arg\min} \; f(\hat{\mathbf{u}}_k, \mathbf{x}, \mathbf{c}) \right\} \tag{6}$$

where $\mathcal{L}$ is any given loss function, $\mathbf{x}_k^{pred}$ is an optimal solution to the fitted forward optimization model instantiated at $\mathbf{u} = \hat{\mathbf{u}}_k$, and $\mathcal{X}_{\hat{\mathbf{u}}_k} = \{\mathbf{x} \mid g(\hat{\mathbf{u}}_k, \mathbf{x}) \leq \mathbf{0}\}$ is the feasible region corresponding to $\hat{\mathbf{u}}_k$. There are various ways to reformulate (6); the particular model we use here is the model proposed by Keshavarz et al. (2011), which formulates the optimality criterion via KKT optimality conditions as:

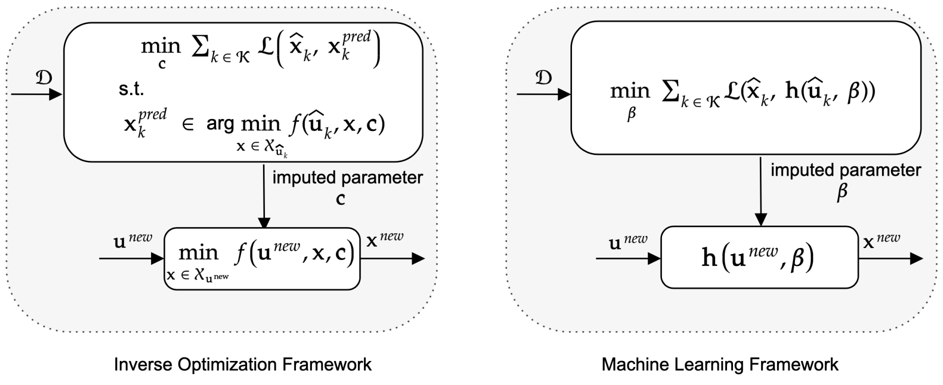
$$\begin{aligned}
\mathbf{IO}(\mathcal{D}) : \underset{\boldsymbol{\lambda}, \mathbf{c}}{\text{minimize}} \quad & \sum_{k \in \mathcal{K}} \phi(\mathbf{r}_k^{stat}, \mathbf{r}_k^{comp}) \\
\text{s.t.} \quad & r^{statj}_{\;\;k} = \frac{\partial f(\mathbf{u}, \mathbf{x}, \mathbf{c})}{\partial x_j} + \sum_{i=1}^{m} \lambda_i^k \frac{\partial g_i(\mathbf{u}, \mathbf{x})}{\partial x_j} && \forall j \in \mathcal{J}, k \in \mathcal{K} \\
& r^{compi}_{\;\;k} = -\lambda_k^i g_i(\mathbf{u}, \mathbf{x}) && \forall i \in \mathcal{I}, k \in \mathcal{K} \\
& \boldsymbol{\lambda} \geq \mathbf{0},
\end{aligned}$$

where ϕ is some norm, $\mathbf{r}^{stat}$ and $\mathbf{r}^{comp}$ represent the complementary slackness and stationarity residuals, respectively, of the KKT conditions; $\boldsymbol{\lambda}$ is the dual variable. The missing objective function coefficient vector $\mathbf{c}$ is a decision variable and pairs of $(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k)$ are the inputs.

The goal of IO could be explanatory, i.e., we aim to characterize the process that generated the data, or predictive, i.e., we want to fit a model that will predict the actions of an optimizing agent/process. That is, having imputed $\mathbf{c}$ and given $\mathbf{u}^{new}$ we could use $\mathbf{FOP}(\mathbf{u}^{new}, \mathbf{c})$ to predict $\mathbf{x}^{new}$. Fitting a model to data can also result in better prescriptive performance, i.e., if instead of predicting what an optimizing agent/process would do, we want to prescribe an optimal decision under new circumstances $\mathbf{u}^{new}$ (this latter perspective being more consistent with classical optimization).

3.4 Inverse Optimization Problem as a Learning Problem

Learning is the capability to acquire knowledge by extracting patterns from raw data (Goodfellow et al., 2016); in this sense, IO fits within the broader conceptual framework of learning, i.e., we are acquiring knowledge about an optimization process that generated the data we observe. Applied to the IO problem described in Section 3.3, supervised ML would aim to find a vector-valued function $\mathbf{h}(\mathbf{u}, \boldsymbol{\beta})$ from $\mathbf{u}$ (input variable or feature) to $\mathbf{x}$ (target variable) such that the loss function $\mathcal{L}$ is minimized (Russell and Norvig, 2016). Once $\mathbf{h}(\mathbf{u}, \boldsymbol{\beta})$ is imputed, it can be used for prediction given new feature observations $\mathbf{u}^{new}$. Figure 1 illustrates this observation: in the left panel, we shown that IO takes pairs of $(\hat{\mathbf{u}}, \hat{\mathbf{x}})$ (i.e., data) as input, imputes the value of $\mathbf{c}$ (i.e., knowledge) and then, given $\mathbf{u}^{new}$ and an imputed $\mathbf{c}$, makes a prediction $\mathbf{x}^{new}$; in the right panel, we show that the same happens in a typical supervised ML framework, with the difference being the actual models and methods used to impute the unknown parameters and, consequently, the learned model that is used for prediction.



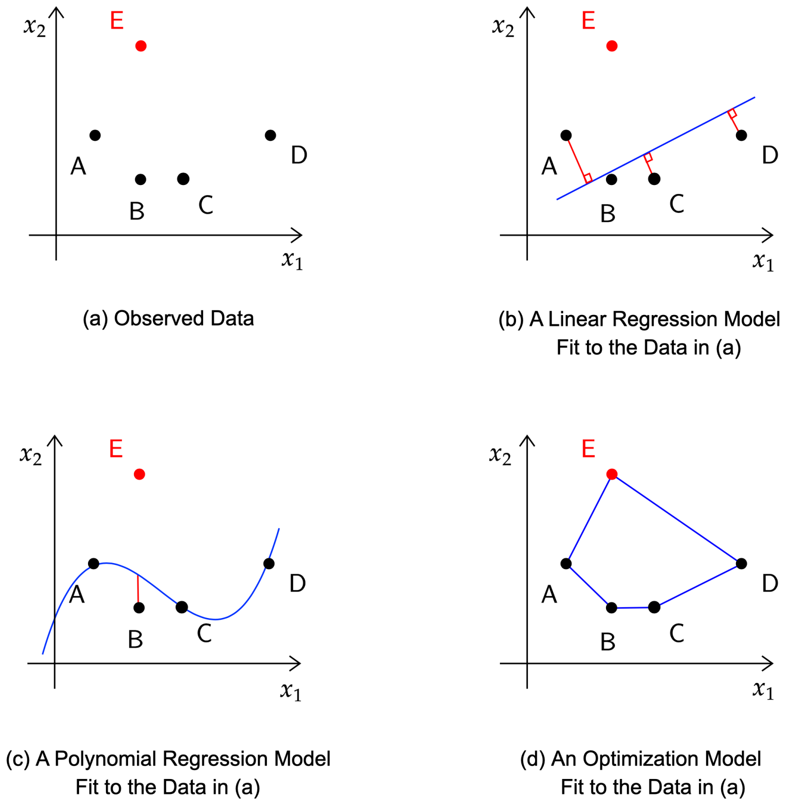
$\mathcal{D}$
$\min_{\mathbf{c}} \sum_{k\in\mathcal{K}} \mathcal{L}\left(\hat{\mathbf{x}}_k, \mathbf{x}_k^{pred}\right)$
s.t.
$\mathbf{x}_k^{pred} \in \arg\min_{\mathbf{x}\in\mathcal{X}_{\hat{\mathbf{u}}_k}} f(\hat{\mathbf{u}}_k, \mathbf{x}, \mathbf{c})$
imputed parameter
$\mathbf{c}$
$\mathbf{u}^{new}$
$\min_{\mathbf{x}\in\mathcal{X}_{\mathbf{u}^{new}}} f(\mathbf{u}^{new}, \mathbf{x}, \mathbf{c})$
$\mathbf{x}^{new}$
Inverse Optimization Framework
$\mathcal{D}$
$\min_{\beta} \sum_{k\in\mathcal{K}} \mathcal{L}(\hat{\mathbf{x}}_k, \mathbf{h}(\hat{\mathbf{u}}_k, \beta))$
imputed parameter
β
$\mathbf{u}^{new}$
$\mathbf{h}(\mathbf{u}^{new}, \beta)$
$\mathbf{x}^{new}$
Machine Learning Framework

Figure 1: IO and ML perspectives applied to an IO problem.

The fundamental assumption in IO is that the observations of decisions $\hat{\mathbf{x}}$ are near-optimal or optimal solutions to an optimization problem, whereas general ML methods do not require such an assumption. Based on the description above, an IO problem can be seen as part of the same conceptual framework as machine learning – the main difference being that in IO data is assumed to come from an optimization process. Viewed another way, we can say that IO assumes a strong prior that the input data is coming from an optimization process, and IO methods leverage this prior knowledge.

Consider Figure 2 (a); suppose that points A to D are given data and our goal is to predict the next observation; point E is an unseen test point for which we want to evaluate the quality of prediction. Given no additional information, a natural idea would be to fit a curve using simple linear or polynomial regression, as shown in panels (b) and (c), respectively. In this particular example, the training error for these models is low but the test error is high, which would normally signal over-fitting to the training data. However, in this case the problem is not over-fitting — rather, the issue is that the data is *not* coming from a linear or a polynomial relationship between x_1 and x_2 and also not from a linear or polynomial relationship between a feature u and each of the x_i. In reality, A, B, C and D are optimal solutions to the parametric optimization problem $P(u)$: max $(-6+5u)x_1 - (3+10u)x_2$ subject to the convex hull of (A, B, C, D, E), shown in Figure 2 (d), with $P(-0.96 \leq u \leq 0.16) = A$, $P(-0.16 \leq u \leq 1.25) = B$, $P(1.25 \leq u \leq 17.7) = C$, $P(17.7 \leq u \leq 20) = D$ and $P(-20 \leq u \leq -0.96) = E$. This example was generated using the Wolfram Parametric Linear Programming app (Bunduchi and Mandric, 2011).

Just like we aim to fit a linear regression model when we conjecture the relationship is linear, or a polynomial one when we conjecture it is polynomial, we can fit an optimization model when we think the data is coming from an optimization problem.



(a) Observed Data

(b) A Linear Regression Model Fit to the Data in (a)

(c) A Polynomial Regression Model Fit to the Data in (a)

(d) An Optimization Model Fit to the Data in (a)

Figure 2: A data set and the fitted optimization model (points A to D are the training set, point E is the test set).

Prior knowledge of the process that generated the data can have a substantial impact on the ability to fit a good model; the above example illustrates specifically the effect of incorporating an optimization model prior. Next, we compare the predictive performance of IO methods, which assume an optimization prior, and ML methods, which do not, when data is generated from parametric convex optimization problems.

3.5 Experiments

3.5.1 Preliminaries

Our goal is to evaluate 'out-of-the-box' approaches that can be easily implemented in practice: a KKT-based IO model (Keshavarz et al., 2011), random forest (Breiman, 2001), support vector regression (Drucker et al., 1997) and Gaussian process regression (Rasmussen, 2004). We use the scikit-learn (Pedregosa et al., 2011) implementation of the ML methods. Keshavarz et al. (2011)'s KKT-based IO model is solved using *Concert Technology* of IBM ILOG CPLEX v.12.7.0. All experiments are run on an HP server with 20 Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz processors and 512 GB RAM under Linux environment.

To show the value of prior knowledge for IO, we consider two types of IO models. In *IO-perfect*, we have perfect information about the *form* of the unknown objective function (but do not know its parameters). In *IO-imperfect*, the objective function is misspecified, and so differs from the true one due to modeling error. Pairs $(\hat{\mathbf{u}}_k, \hat{\mathbf{x}}_k)$ are generated using Algorithm 1 and divided into a training set and a test set. In Algorithm 1, K refers to the number of observations and $\mathbf{x}_k^{true}$ refers to the optimum decisions of FOP_{true}; from the perspective of data generation, these optimal solutions become observations, denoted $\hat{\mathbf{x}}_k$ in the set of data $\mathcal{D}$. We use leave-one-out cross-validation on the training set for the ML methods: we pick one point as our validation set, fit a model to the remaining data and evaluate the error on the single held-out point. A validation error estimate is obtained by repeating this procedure for each of the training points and averaging the results. Doing so with several hyperparameter settings enables us to choose the hyperparameter setting with the lowest average cross-validation error. All of the ML experimental results presented below show the error on the test set for the best hyper-parameter setting found during the leave-one-out cross-validation procedure.

We measure the deviation between the predicted $\mathbf{x}$ and $\mathbf{x}^{true}$ (the true values of the decisins in the test set) in the test set (of size M) using mean relative error (MRE):

$$MRE = \frac{1}{M}\sum_{m=1}^{M}\frac{\|\mathbf{x}-\mathbf{x}^{true}\|_2}{\|\mathbf{x}^{true}\|_2}. \tag{7}$$

3.5.2 Experiment 1: Utility Function Estimation

Consider imputing a customer's unknown utility function given observations of their past purchases (Keshavarz et al., 2011; Bärmann et al., 2017). We assume that customers solve an internal optimization problem in the course of purchasing goods to maximize their utility function. This is the same problem as the one addressed

Algorithm 1 Data Generation

```
for $k \leftarrow 1$ to $K$ do
    generate $\mathbf{u}_k$ from uniform distribution
    solve $\mathbf{FOP}_{true}(\mathbf{u}_k, \mathbf{c}^{true})$ and get $\mathbf{x}_k^{true}$
    $k \leftarrow k+1$
end for
$\mathcal{D} \leftarrow \{(\hat{\mathbf{x}}_1, \hat{\mathbf{u}}_1), (\hat{\mathbf{x}}_2, \hat{\mathbf{u}}_2), ..., (\hat{\mathbf{x}}_K, \hat{\mathbf{u}}_K)\}$
```

by Keshavarz et al. (2011):

$$\begin{aligned} UFOP(\mathbf{p}) : \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{p}^T\mathbf{x} - U(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{p} \in \mathbb{R}^n$ are the vectors of consumer demand and the associated price[1], respectively. Problem (8) is a relatively simple representation of the consumer's utility optimizing problem when the objective function is minimizing the total spending minus their gained utility. This model can be used to predict the the consumer's demand under different values of price. In this section, the external parameter of the POP (previously indicated by $\mathbf{u}$) is parameter $\mathbf{p}$. Similarly to Keshavarz et al. (2011), we assume that the function $U(\mathbf{x})$ is a concave utility function which is non-decreasing over $[0, \mathbf{x}^{\max}]$, $\mathbf{x}^{\max}$ being the maximum demand level found from past observations. Since $U(\mathbf{x})$ is concave, the objective function of (8) is convex.

We generate $p_i \sim Uniform[5, 20]$ i.e., the parameters are generated from the uniform distribution. The true utility function is $U_{true}(\mathbf{x}) = \sum_{i=1}^{n} \sqrt{x_i}$. We use $U_{perfect}(\mathbf{x}) = c_i\sqrt{x_i}$ in implementing *IO-perfect* (i.e., the model is correctly specified so that all observations can be made optimal solutions for their corresponding p_i). For *IO-imperfect*, we use $U_{imperfect}(\mathbf{x}) = \sum_{i=1}^{n} qx_i^2 + 2rx_i$. Since this function is different from $U_{perfect}$, the inverse optimization model with this function has imperfect information. We chose a concave quadratic function for the $U_{imperfect}$.

[1]We do not consider any specific currency for the parameter $\mathbf{p}$ in this problem. As we do not have specific constraints on the utility function (e.g., upper and lower bound on the value of the utility function), the currency of the price $\mathbf{p}$ does not change the value of unknown parameters of the utility function as the relative magnitude of $\mathbf{p}$ and the utility function unknown parameters would be the same regardless of the currency.

Based on the results of our hyper-parameter search during leave-one-out cross-validation, we use the scikit-learn implementation of random forest with `n_estimators=50`, `max_depth=None`, `max_feature=None`, support vector regression with `c=0.1`, `gamma="auto"`, `kernel=RBF` and Gaussian process with `kernel=RBF`, `alpha=1e-10`, `normalized_y=False`, `optimizer="fmin_1_bfgs_b"` and `n_restarts_optimizer=10`.

Figure 3 shows how the relative test error of *IO-perfect*, *IO-imperfect*, RF, SVR and GP changes when we increase the training size. Whereas RF, SVR and GP performance improves, the performance of *IO-imperfect* and *IO-perfect* is not affected. GP achieves performance similar to that of *IO-imperfect* with only 20 observations, while RF and SVR get close to *IO-imperfect* at 60 and 100 points, respectively. The ML methods perform well because the underlying optimization process (resulting from model (8)) is constrained by only non-negativity constraints and is parametric only in the objective function – the relationship between the input and output of this process can be captured (learned) without knowledge of the underlying optimization model. In the next experiment, we aim to identify IO problems which would be more difficult for these ML methods and would require knowledge of the underlying optimization structure to make good predictions.

3.5.3 Experiment 2: Randomly-Generated Parametric Optimization Problems

In this experiment, we compare the performance of inverse optimization and machine learning methods using near optimal solutions of general parametric optimization problems. Using the POP toolbox (Oberdieck et al., 2016), we generate POPs of the

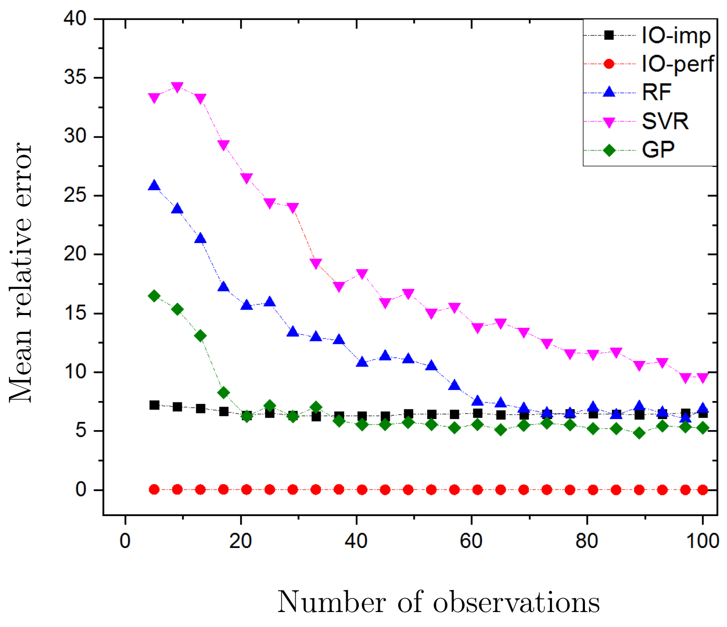


Figure 3: Comparing *IO-imperfect*, *IO-perfect*, RF, SVR, and GP for utility function estimation; each point is an average over 15 randomly-generated problems, $p_i \sim Uniform[5, 20]$. Mean relative error is calculated by formulation (7).

following form:

$$\begin{aligned} TPOP(\mathbf{u}) : \min_{\mathbf{x}} \quad & (\mathbf{Qx} + \mathbf{Hu} + \mathbf{c})^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \leq \mathbf{b} + \mathbf{Fu} \\ & \mathbf{x} \in \mathbb{R}^n \\ & \mathbf{u} \in \mathbb{U} := \{\mathbf{u} \in \mathbb{R}^q \mid \mathbf{CR}_A \mathbf{u} \leq \mathbf{CR}_b\}, \end{aligned} \tag{9}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n} \succ 0$, $\mathbf{H} \in \mathbb{R}^{n \times q}$ and $\mathbf{c} \in \mathbb{R}^{n \times 1}$ and $\mathbf{CR}$ represents critical regions. In the inequality constraints, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^{m \times 1}$ and $\mathbf{F} \in \mathbb{R}^{m \times q}$. The last constraint of (9) restricts the parameter space, i.e., the admissible values of external parameter (feature) $\mathbf{u}$, where $\mathbf{CR}_A \in \mathbb{R}^{2q \times q}$ and $\mathbf{CR}_b \in \mathbb{R}^{2q \times 1}$ allow for representations of constraints on $\mathbf{u}$ (in particular, lower and upper bounds – hence the number of rows

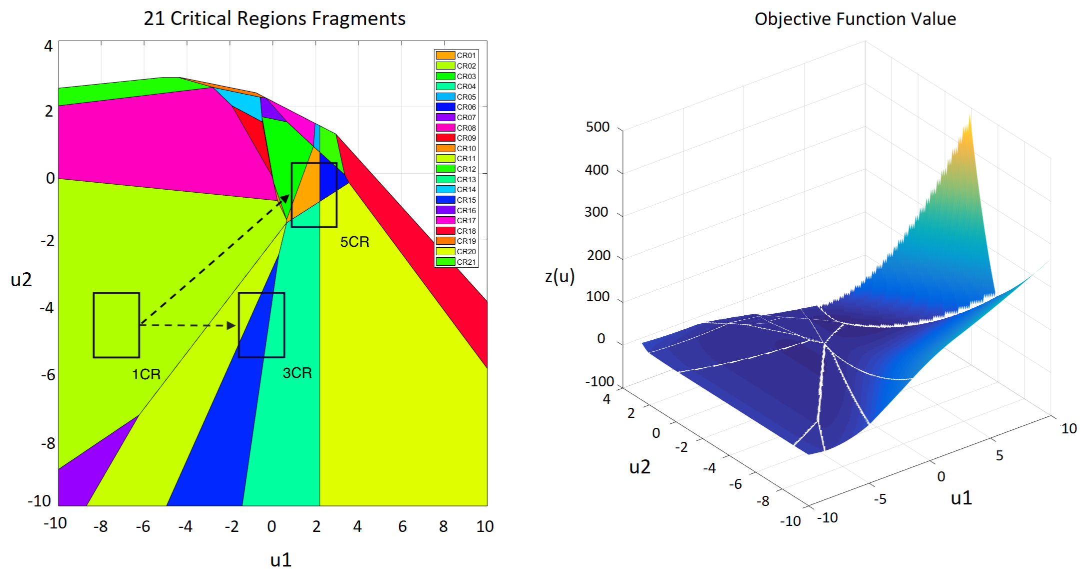


Figure 4: A POP instance defined over two parameters u_1 and u_2 generated using the POP toolbox of Oberdieck et al. (2016). The left panel shows 21 critical regions of the parameter space, while the right panel shows the piecewise quadratic optimal objective function $z(\mathbf{u})$. Each critical region on the left corresponds to a quadratic 'piece' of $z(\mathbf{u})$ on the right.

is $2q$), and q denotes the dimensionality of $\mathbf{u}$. We assume $q = 2$ in our experiments.

To study (forward) POPs, it is standard to view the space of feasible parameters $\mathbf{u}$ as being partitioned into polyhedrons called *critical regions*, each of which is uniquely defined by a set of optimal active constraints (Ahmadi-Moshkenani et al., 2018; Oberdieck et al., 2016). It is known that if $\mathbf{Q}$ is positive definite and $\mathbb{U}$ is convex, the optimal objective value of TPOP($\mathbf{u}$), $z(\mathbf{u})$ is continuous and piecewise quadratic, while if the problem is reduced to a multi-parametric linear program, then the optimal objective function $z(\mathbf{u})$ is continuous, convex, and piecewise affine (see Oberdieck et al. (2016) and the references therein). Figure 4 shows an example of one of the instances we generate using the POP toolbox of Oberdieck et al. (2016) with 21 critical regions and a two-dimensional $\mathbf{u}$; the critical regions are shown in the left panel while the corresponding optimal value function $z(\mathbf{u})$ is shown on the right.

As the objective function $z(\mathbf{u})$ is piecewise affine over the space of external parameter $\mathbf{u}$, having more critical regions means having more pieces of $z(\mathbf{u})$ and as it is not known where one piece of function $z(\mathbf{u})$ transitions to another, the performance of the learning methods can be affected. Therefore, in this section, we conduct experiments varying the number of critical regions. To be able to control the number of critical regions in our experiments, we manually pre-process the instances generated from the toolbox to define problems over smaller regions of the parameter space with a particular number of critical regions. For instance, from the example in Figure 4, we can generate a one-region, a three-region and a five-region problems by selecting the regions indicated by the black rectangle borders. Throughout this process, care is taken to select intervals where each critical region has close to a "fair share" of the region; in preliminary experiments, it was noted that the difficulty of learning over a parameter space interval that covers, say, five regions but is dominated by one particular region is similar to the case of one critical region (more discussion on the effect of critical regions on learning performance appears later in this section).

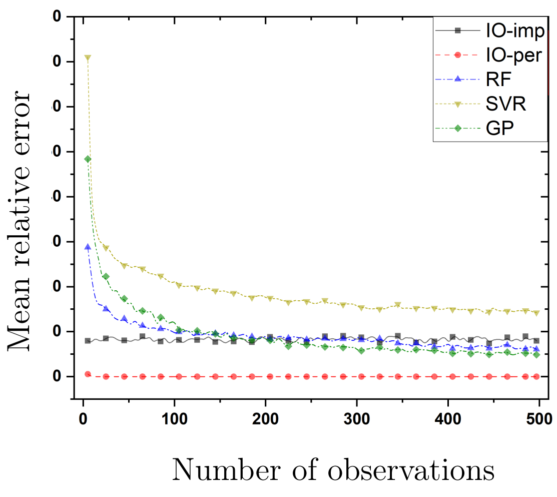
Varying the Training Set Size (Data Efficiency)

We compare the predictive performance of *IO-perfect*, *IO-imperfect*, SVR, RF and GP varying the training size from 1 to 500 on 20 randomly generated POPs of type (9) with two parameters u_1 and u_2, with 1 and 3 critical regions. For *IO-imperfect* we use the mean of selected intervals of $\mathbf{u}$ in the $(\mathbf{Hu})^T\mathbf{x}$ term in the objective function instead of a parametric term.

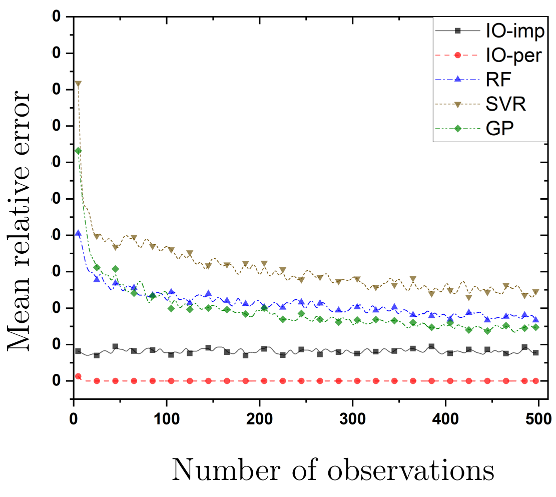
Figures 5(a) and 5(b) illustrate the predictive (i.e., test set) performance of these methods (averaged over 20 instances) given data sets from one critical region and three critical regions, respectively. As expected, ML performance improves with increasing training set size in both cases. On the contrary, IO methods are not affected: *IO-perfect* has nearly zero error while *IO-imperfect* incurs around 8% error, regardless of the training set size. Since IO is able to achieve low prediction error (in terms of predicting $\mathbf{x}$) with a much smaller training size, we see that IO methods with correctly specified priors i.e., the correct parametric form of the objective function, can be more data efficient than ML algorithms for problems where data is generated by an optimization process.

Because the IO approach we chose solves an optimization model, no matter how many points are given, *IO-perfect* is able to recover the parameters that make the observations optimal while *IO-imperfect* will maintain the same level of misspecification. Insensitivity to the training set size is consistent with Aswani et al. (2018)'s results, which showed that Keshavarz et al. (2011)'s method is in general not risk consistent, i.e., it is not guaranteed to asymptotically provide the best possible predictions of the decisions. However, we emphasize that despite being risk inconsistent, given the right prior, the Keshavarz et al. (2011) approach performs well, and can be the method of choice due to its data efficiency.

The test error for ML methods increases for all training set sizes as we move from one critical region (Figure 5(a)) to three critical regions (Figure 5(b)). Comparing Figures 5(a) and 5(b), the curves for GP and RF in 5(a) intersect *IO-imperfect* after 200 points, but in 5(b) they require more points to be comparable to *IO-imperfect*. This observation suggests that the number of critical regions in the parameter space may be a factor that makes these problems more difficult for standard ML approaches, motivating the next experimental comparison.



(a) One Critical Region



(b) Three Critical Regions

Figure 5: Comparing *IO-perfect*, *IO-imperfect*, SVR, RF and GP varying training sample size and critical regions.

Increasing the Number of Critical Regions

Figure 6 shows the average predictive performance of *IO-imperfect* and the three ML methods over 20 randomly generated POPs given a training set of 200 observations. For each of the 20 problem instances, we choose one interval of $\mathbf{u}$ with one critical region, one with three critical regions and one with five, keeping the chosen range of both u_1 and u_2 equal to two, as in Figure 4. We can see that *IO-imperfect* is not sensitive to the number of critical regions while for all employed ML methods the test error increases with the number of critical regions. As mentioned above, every critical region is a polyhedron of $\mathbf{u}$ values over which the optimal objective function $z(\mathbf{u})$ is quadratic (a quadratic 'piece' of the overall piecewise quadratic function) while the optimizer function $\mathbf{x}^*(\mathbf{u})$ i.e., the optimal solution of the POP, is piecewise-affine (Oberdieck et al., 2016). It is not surprising that some of the ML methods can perform well with 200 points over one critical region as they only have to learn one quadratic

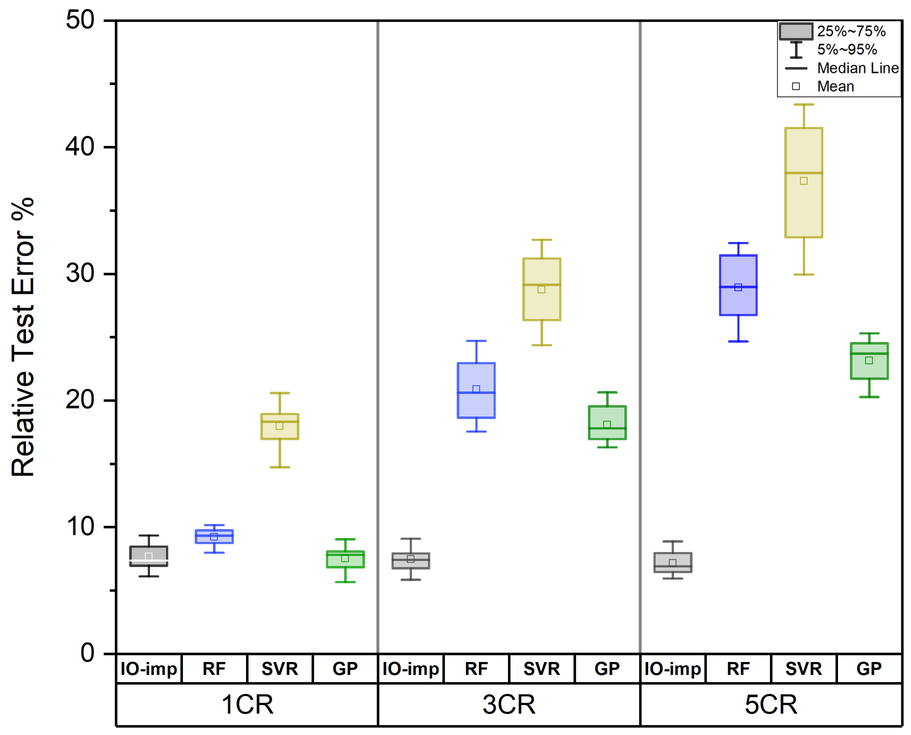


Figure 6: Comparing *IO-imperfect*, SVR, RF and GP as the number of critical regions increases; training set size = 200, test set size = 200.

function. When the number of critical regions increases, the task for the chosen ML methods becomes more challenging: they have to learn multiple quadratic functions without any prior knowledge about when the transition from one region to another will occur. We do not rule out the possibility of creating new, custom learning algorithms to overcome this issue but doing so is beyond the scope of this chapter.

Varying the Functional Dependence of the Objective Function on $\mathbf{u}$

Next, we investigate how the relationship between the TPOP objective function and the feature $\mathbf{u}$ affects the performance of ML and IO. We investigate this question in

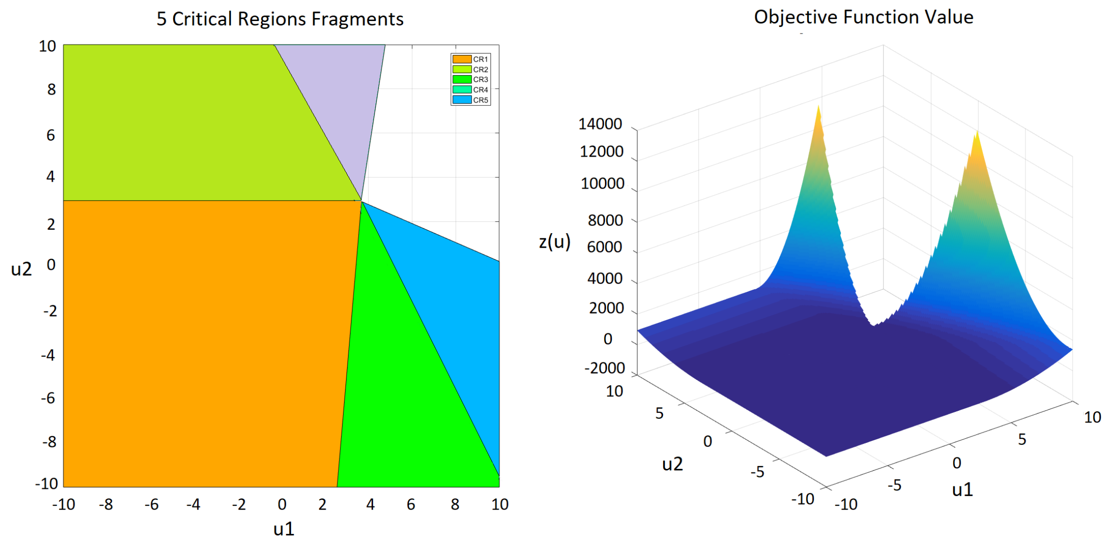


Figure 7: Problem (10) critical regions (left) and parametric objective function (right) generated using the POP Toolbox of Oberdieck et al. (2016).

the context of one randomly generated POP (see Figure 7):

$$\begin{aligned} z(\mathbf{u}) : \underset{\mathbf{x}}{\text{minimize}} \quad & 1.3040x_1^2 + (1+u_1)x_1 + 19.4545x_2^2 + (u_2 - u_1 + 1)x_2 \\ \text{s.t.} \quad & 0.2294x_1 \leq 2.5237 - 0.9733u_2 \\ & 0.1890x_2 \leq 4.2679 - 0.8658u_1 - 0.4634u_2 \\ & 0.5436x_1 - 0.5889x_2 \leq 2.8088 - 0.4757u_1 - 0.3624u_2 \\ & 0.2210x_1 \leq 3.2535 - 0.9753u_1. \end{aligned} \tag{10}$$

We consider three objective functions for problem (10) defined through functions $\Psi_1(\mathbf{u})$ and $\Psi_2(\mathbf{u})$, which set the coefficients of x_1 and x_2:

$$f = 1.3040x_1^2 + \Psi_1(\mathbf{u})x_1 + 19.4545x_2^2 + \Psi_2(\mathbf{u})x_2. \tag{11}$$

We start with a linear $\Psi(\mathbf{u})$ followed by changing it to a multi-variate rational function and subsequently increasing the degree of the polynomials forming the rational function to create a relationship that could lead to a more challenging learning problem.

As shown in Table 1, the test error of GP, SVR and RF increases as we move from the 'simple' (linear) dependence on $\mathbf{u}$ to making the coefficient of $\mathbf{x}_1$ and $\mathbf{x}_2$ multi-variate rational functions of $\mathbf{u}$. Since ML methods aim to find a mapping from $\mathbf{u}$ to $\mathbf{x}$, a more complex $\Psi(\mathbf{u})$ indeed makes learning more difficult. On the other hand, IO is not sensitive to the nature of $\Psi(\mathbf{u})$ as this function is just the coefficient of $\mathbf{x}$ in the IO mathematical model.

Investigating the Impact of Correctness of Prior Knowledge

Above, we show that *IO-perfect* and *IO-imperfect* are insensitive to problem characteristics which significantly impact the performance of ML (training set size, number of critical regions and the nature of the dependence on $\mathbf{u}$). However, a difference between *IO-perfect* and *IO-imperfect* was observable. To investigate the impact of the correctness of objective function prior, we consider five objective function choices for problem (10) which encode different degrees of correctness (or mis-specification) of a prior.

In Table 2, given $\mathcal{D}$, we aim to impute c_1 and c_2. The first objective function is the exact objective function of problem (10), i.e., the *IO-perfect* method. The subsequent functions deviate from the true function in the coefficients of x_1 and x_2 (but the goal is still to impute c_1 and c_2 from the data generated by the true objective function of problem (10)). Besides parametric objective functions (i.e., objective functions defined over $\mathbf{u}$), we also consider two non-parametric choices. In the first

Table 1: Comparing the predictive performance of ML and IO varying the form of the true objective function, with $\hat{u}_1 \sim U(4, 6)$ and $\hat{u}_2 \sim U(-6, -4)$. Recall that K is training set size. The test set size is set to be equal to the training set size.

True Objective Function	Method	MRE % ($K = 100$)	MRE % ($K = 200$)
$1.3040x_1^2 + (1 + u_1)x_1 + 19.4545x_2^2 + (u_2 - u_1 + 1)x_2$	*IO-perfect*	~ 0	~ 0
	IO-imperfect	3.85	3.79
	GP	9.75	6.03
	RF	6.75	5.74
	SVR	19.03	18.04
$1.3040x_1^2 + \frac{1+u_1}{1-u_1}x_1 + 19.4545x_2^2 + \frac{(u_2-u_1+1)^3}{(u_2-1)^2}x_2$	*IO-perfect*	~ 0	~ 0
	IO-imperfect	7.80	7.11
	GP	10.85	8.59
	RF	9.77	7.07
	SVR	24.94	23.30
$1.3040x_1^2 + \frac{u_1}{(1-2u_1)^4}x_1 + 19.4545x_2^2 + \frac{(u_2-u_1)^3}{(3u_2-5)^5}x_2$	*IO-perfect*	~ 0	~ 0
	IO-imperfect	7.78	7.62
	GP	15.28	10.29
	RF	22.24	17.75
	SVR	38.76	34.80

non-parametric objective, we use the mean of $1 + \hat{u}_1$ and $\hat{u}_2 - \hat{u}_1 + 1$ from the given training set $\hat{u}_1$ and $\hat{u}_2$; in the second non-parametric function, we completely eliminate the linear terms.

We see that even slight differences in the chosen objective function structure increase IO test error, although the differences are smaller in the category of parametric

Table 2: Different parametrizations of the objective function of problem (10) and the corresponding test error, given $\hat{u}_1 \sim U(4,6)$ and $\hat{u}_2 \sim U(-6,-4)$.

	Objective Function	MRE % ($K = 100$)
Parametric	$c_1x_1^2 + (1+u_1)x_1 + c_2x_2 + (u_2 - u_1 + 1)x_2$	~ 0
	$c_1x_1^2 + x_1 + c_2x_2 + (u_2 - u_1)x_2$	0.38
	$c_1x_1^2 + x_1 + c_2x_2 - u_1x_2$	3.56
Non-Parametric	$c_1x_1^2 + 6x_1 + c_2x_2 - 9x_2$	3.88
	$c_1x_1^2 + c_2x_2$	52.28

functions as compared to the non-parametric functions because the parametric category uses the values of the external parameter while the non-parametric category uses the mean of the interval of the external parameter which is not always an accurate estimate. The non-parametric mean-based function also does not lead to a significant increase, but this could change if a larger range of $\hat{u}$ is considered. Removing any encoding of the parametric parts of the objective function drastically increases the prediction error from 3.88% to 52.28%. These observations confirm the sensitivity of IO to the correctness of objective function prior.

3.5.4 Insights

Our experiments with randomly-generated POPs illustrate the importance of four characteristics as determinants of the difficulty of objective function learning: (i) size of the training set, (ii) nature of the dependence of the optimization problem on the external parameters, (iii) level of confidence with regards to the correctness of the optimization prior, (iv) number of critical regions in the parameter space of a POP.

Increasing the size of the training set favours ML methods but not necessarily the IO methods. The contrast in performance between the problem in experiment

Table 3: Problem Characteristics and Suggested Methods.

	Low/Small	High/Large
Size of Training Set	IO	ML
Dependence on External Parameter	ML	IO
Confidence in Correctness of Prior	ML	IO
Number of Critical Regions	ML	IO

1 and the randomly-generated POPs suggests that ML methods are more likely to be successful on problems with no or few (parametric) constraints. Increasing the complexity of the relationship between $\mathbf{u}$ and the objective function also makes the problem more difficult for ML. On the other hand, the performance of IO is strongly dependent on the correctness of the objective function prior.

Perhaps the most important characteristic determining the difficulty of the objective function learning tasks is the number of critical regions. Over 20 randomly-generated POPs, we found a substantial increase in the relative test error for all three ML methods as the number of critical regions increased. In fact, we conjecture that the underlying reason why adding parametric constraints and/or increasing the complexity of the dependence on $\mathbf{u}$ made learning more difficult is because doing so also increased the number of critical regions.

Given these observations, a summary of recommendations of when to use ML or IO for learning the coefficients of a convex POP is needed. We summarize our recommendations in Table 3. In each combination of a criterion and its value (low/high or small/large), we state whether we expect the classic ML methods or an IO method to be a better choice (or at least a better starting point) for solving the learning problem, while holding other criteria fixed. For example, when training set size is small, we expect IO to be a better option due to its data efficiency.

The relative performance of the methods considered may not be the same for other types of learning problems where data was generated by an optimization problem (e.g., learning constraints, learning in discrete optimization problems). However, we conjecture that the analysis of the underlying structure of the value function or the optimizer function in terms of the external parameters $\mathbf{u}$ (i.e., analysis of the problem in terms the critical regions) will be important in both gaining additional understanding of the challenges of learning from optimization data and developing more sophisticated learning methods for such problems.

3.6 Conclusion

In this chapter, we view inverse optimization as a problem of learning from decisions that are made through an unknown optimization process. We specifically focus on the problem of learning a convex objective function of a parametric optimization problem. We experimentally compare the predictive performance of an inverse optimization method with perfect and imperfect priors with three well-known machine learning algorithms: support vector regression, random forest and Gaussian processes. While we show that some inverse optimization problems can be tackled through classic machine learning approaches, we highlight the need for sophisticated inverse optimization models for problems where at least one of the following characteristics holds: (i) the size of the training set is small, (ii) both the constraints and the objective function of the problem in question are dependent on an external parameter (feature), particularly when that dependence is non-linear, (iii) we have high confidence that our knowledge of the parametric nature of the constraints and objective is correct, and (iv) the number of critical regions of the POP is large with no one region dominating. We believe that these observations provide practitioners with guidance on when to consider employing inverse optimization instead of, or in addition to, classical machine learning when learning the unknown parameters of a parametric optimization problem objective function.

Chapter 4

Inverse Attribute-based Optimization with an Application in Assortment Optimization

Many applications of inverse optimization arise in settings where the goal is to predict the future actions of an optimizing agent (e.g., an optimizing customer's future purchases). The majority of papers in this area implicitly assume an *alternative-based* modelling approach: the forward model finds an optimal set of actions (decisions) from among a given set of alternatives, while the inverse model imputes objective function coefficients corresponding to these alternatives. Since the imputed weights correspond only to alternatives existing in the training set, alternative-based modelling is limited to applications where the set of options does not change when a prediction is needed. In this chapter, we apply an *attribute-based* perspective, which allows IO to impute the weights of attributes that lead to an optimal decision instead of imputing the weight of the decision itself. This perspective expands the range of IO applicability; we demonstrate that it facilitates the application of IO in assortment optimization, where changing product selections is a defining feature and accurate predictions of demand are important. We compare IAO with rank-based and machine learning methods. We show that since inverse optimization encodes the utility optimizing behaviour of the consumer into the preference learning process, it results in lower assortment regret for the store and a lower utility gap for the consumers.

4.1 Introduction

Multi-attribute decision-making problems arise in situations where decisions/alternatives can be described through attributes (Yoon and Hwang, 1995; Yeh, 2002). A practical challenge with many existing MADM methods is the quantification of attribute importance to represent decision maker preferences. It is known that asking the decision maker to make exemplary decisions could be a better strategy than asking them to specify preferences or weights (Greco et al., 2002). While methodologies such as rough sets theory exist for extracting information from exemplary decisions, MADM methods usually do not distinguish the setting where the decision maker is an optimizer, which warrants special treatment.

Inverse optimization is a method that imputes missing optimization model parameters from data that represents optimal or near-optimal decisions of the underlying unknown optimization problem. IO has been successfully applied in many areas, including finance (Bertsimas et al., 2015), health care (Chan et al., 2019), electricity demand forecasting (Saez-Gallego and Morales, 2017) and the household activity pattern problem (Chow and Recker, 2012). The majority of the IO literature looks at imputing missing objective function parameters. In single-objective optimization problems, these parameters are coefficients associated with particular decision variables, frequently interpreted to represent the importance, value or cost of the decisions or alternatives represented by those decision variables. In inverse multi-objective optimization (Ajayi et al., 2020; Chan and Lee, 2018; Dong and Zeng, 2020) the goal is typically to infer the weights of different sub-objectives from observed decisions.

Motivated by an application in assortment optimization, this chapter is based on the observation that current IO modelling approaches are limited to applications where the set of decisions (or alternatives) stays the same; the weights imputed from past data represent decision maker preferences among a previous set of alternatives – if this set changes, the imputed weights become unusable. To remedy this issue, we propose to apply IO not to alternative-based models but, borrowing from MADM, to their attribute-based reformulations. We refer to this application of IO to attribute-based forward problems as *inverse attribute-based optimization* (IAO). Instead of imputing the weights of decisions, IAO imputes the weights of attributes of those decisions; thus, when the set of decisions changes, weights imputed from past data can be utilized to parameterize an optimization problem given a new set of alternatives, as long as both past and current alternatives can be expressed in terms of the same attributes.

Combining IO with an attribute-based perspective expands the applicability of IO in practice. One application to which we could not previously apply IO is assortment optimization. Assortment optimization deals with finding the best assortment (usually for a retail environment) given budget constraints (costs of buying products, transportation, inventory) and space constraints (Farias et al., 2017). To make assortment decisions, the firm must accurately predict future consumer demand (e.g., for the next season), which is a challenging task due to the variety of the offered products and the relatively small number of purchases by individual consumers. In this chapter, we use IAO precisely for this prediction task, under the assumption that consumers are optimizers. Since classical alternative-based modelling does not allow for predicting customer choices given a new set of items, we compare IAO to other methods that *are* applicable in attribute-based settings but *are not inverse optimization* models. In this way, we capture the value of specifically *inverse attribute-based optimization* as opposed to other attribute-based approaches.

Importantly, potential applications outside of customer behaviour are numerous. In transportation, prior IO work imputes costs of specific arcs in a given network; given these imputed costs, it is possible to predict the behaviour of a driver under new conditions (e.g., time of day or weather) on the *same* graph because the set of alternatives (arc choices) remains the same (Bertsimas et al., 2015; Bärmann et al., 2017; Tan et al., 2020).[1] However, the learned arc costs are not usable on a *new* graph, where the set of arc choices is different. The attribute-based perspective we propose would overcome this issue and is important, for instance, for autonomous vehicles (Caballero et al., 2021), which requires models that generalize to new graphs. Similarly, in diet planning, which has been of interest in IO (Ghobadi et al., 2018; Shahmoradi and Lee, 2020; Ghobadi and Mahmoudzadeh, 2020a), it is important to be able to recommend to a client new meals or recipes that are consistent with their diet restrictions and food preferences; yet the recommendations based on current, alternative-based, methods for learning the objective function are limited to the same food items. Expressing each food item in terms of attributes such as their vitamin,

[1]We note that one particular example of a cost function used by Tan et al. (2020) in their minimum-cost multi-commodity flow experiment is consistent with the attribute-based perspective, since there is a weight assigned to an arc's length. However, the importance of this modelling choice and its impact on generalizing to new graphs is not discussed.

mineral or calorie content would greatly enhance the applicability of inversely-optimized plans. In addition, while in this chapter we focus on the "batch" setting, where we are learning parameters from a given set of past observations, we see room for future study in combining our perspective with the *online* setting, where observations arrive over time. The online setting has been receiving an increasing amount of attention in the IO literature (Bärmann et al., 2017; Dong et al., 2018; Bärmann et al., 2020) but has focused on problems where only the set of constraints changes over time; our attribute-based perspective facilitates future application of IO to online problems where the set of alternatives may change as well.

Contributions

The main contributions of this chapter are as follows. We propose a new perspective on inverse optimization based on attributes. Inverse attribute-based optimization aims to impute the weights of attributes that lead to an optimal decision instead of imputing the weight of the decision itself. This perspective expands the range of IO applicability; we demonstrate that it facilitates the application of IO in assortment optimization, where changing product selections is a defining feature and accurate predictions of demand are important. We compare the performance of the proposed inverse attribute-based optimization method with a range of methods, including rank-based weight assigning methods and three supervised learning algorithms. We show that since inverse optimization encodes the utility optimizing behaviour of the consumer into the preference learning process, it results in lower assortment regret for the store as well as higher consumer satisfaction.

This chapter is organized as follows. Section 4.2 reviews the relevant literature. Section 4.3 explains the forward and inverse attribute-based approach presented in this chapter. Section 4.4 illustrates the application of the inverse attribute-based approach in the context of customer behaviour and the subsequent value of the predictions for assortment optimization. Section 4.5 presents the experimental results and Section 4.6 concludes the chapter.

4.2 Literature Review

Below, we survey work from several areas of research related to our approach. First, we highlight the most closely-related work in inverse optimization, which has been dominated by alternative-based modelling. Next, we contrast our work with multi-objective and inverse multi-objective optimization. Finally, we mention related work in multi-criteria decision-making.

We do not provide a literature review for assortment optimization; for further reading on assortment optimization, we refer the reader to the papers of Kök and Fisher (2007) and Kök et al. (2008).

4.2.1 Inverse Optimization

Inverse optimization finds unknown parameters of an optimization problem such that the observed solutions of a *forward problem* become optimal under the imputed configuration. Numerous inverse optimization methods have been developed considering various assumptions mostly about the forward optimization problem type and the characteristics of the observations (Ahuja and Orlin, 2001; Iyengar and Kang, 2005; Schaefer, 2009; Wang, 2009; Chan et al., 2019; Babier et al., 2021). There has recently been an increasing interest in parametric problems (Keshavarz et al., 2011; Bertsimas et al., 2012; Aswani et al., 2018; Esfahani et al., 2018; Tan et al., 2019, 2020).

Esfahani et al. (2018), Aswani et al. (2018) and Keshavarz et al. (2011) consider the problem of imputing a decision maker utility function, which is the problem we use for illustrative purposes. Esfahani et al. (2018) propose an inverse optimization method to learn the preferences of an agent who solves a parametric optimization problem depending on an exogenous signal. They consider that the observer has imperfect information and study three sources of error in data which include modelling error, measurement error and bounded rationality. They formalize this problem as a distributionally robust optimization program minimizing the worst case risk that the predicted decision differs from the agent's actual response. Aswani et al. (2018) study a similar problem. They propose an inverse optimization method for estimating an agent utility function capturing the trade-off between maintaining a comfortable indoor temperature versus the air conditioning energy costs. Keshavarz et al. (2011) propose a general framework of inverse optimization for imputing convex objective functions based on KKT optimality conditions. They study the problem of imputing a consumer's utility function while the consumer solves a convex parametric optimization problem with price being the hyper-parameter (i.e., the parameter that changes the optimization problem) and observations being consumer's demand level for different price levels. To the best of our knowledge, all studies of decision maker behaviour in the inverse optimization literature focus on imputing the preferences of decisions; we are not aware of any study in inverse optimization that considers attributes of the optimal decisions.

Recent studies have also proposed to view IO from a machine learning (ML) perspective (Tan et al., 2020). Chapter 3 showed that learning decision-making model parameters can be viewed from both an inverse optimization and a supervised learning perspective. They showed that when a more complex optimization process generates the data, inverse optimization is more data-efficient than supervised learning methods. However, they also showed that it is possible for classical machine learning methods to be applied to inverse optimization models such as utility function estimation. Since the machine learning concept of a feature can encapsulate the notion of a decisional attribute, classical supervised learning methods are applicable to the type of problems we study in this chapter. Thus, in the current chapter, similarly to Chapter 3 we compare the inverse attribute-based approach with Gaussian process regression, random forest and a decision tree regression (Breiman, 1996, 2001; Rasmussen, 2004).

4.2.2 Multi-Objective Optimization

It is important to contrast our attribute-based perspective with the perspective taken by multi-objective (or multi-criteria) optimization (Ehrgott and Gandibleux, 2000; Ehrgott, 2005). The main difference is in fact in the forward problem: in a multi-objective formulation, each sub-objective represents the value of a criterion (which could indeed have the same interpretation as an attribute) calculated over a subset of the decision variables, while our definition of a multi-attribute problem is a single-objective formulation where the objective coefficients of each decision variable are a function of various attributes; in short, in a multi-objective formulation, we aggregate the total value of an attribute *over all available options* while in our attribute-based formulation, we aggregate the value *over all attributes for each option*. These two formulations may be equivalent under some assumptions but will be different in general.

Consequently, the goal of inverse multi-objective optimization has been on imputing the weights of different sub-objectives (i.e., the relative importance of an attribute calculated over all options), while our aim is to impute the weight of an attribute with respect to a particular option.

Inverse multi-objective optimization has focused on imputing the weights of sub-objectives under different assumptions on Pareto optimality or feasibility of observations and availability of a prior weight vector (Roland et al., 2013; Chan and Lee, 2018; Naghavi et al., 2019; Ajayi et al., 2020; Dong and Zeng, 2020; Gebken and Peitz, 2020). If an attribute-based forward problem can be reformulated as a multi-objective problem, then, under the appropriate assumptions, the methods of inverse multi-objective optimization can be applied to solving inverse attribute-based optimization problems as well.

4.2.3 Multi-Criteria Decision-Making

The problem of evaluating the weights of decision attributes has been intensively studied in multi-criteria decision-making (Greco et al., 2016). Eisenführ et al. (2009) provides an overview of weight elicitation procedures for additive models. To select the best *multi-attribute alternative* and impute the weight of attributes based on past decisions, a variety of approaches based on no information or partial information about the rank of the attributes are found in the literature. Some of the most widely-used methods include 1) assigning equal weights (Dawes and Corrigan, 1974), 2) ranking attributes based on importance then selecting one of the rank weight formulas such as rank sum, rank reciprocal and rank exponent (Stillwell et al., 1981), and 3) using partial information as constraints in an optimization model with a particular objective function to find the set of weights (Barron, 1988).

The use of attributes has been widely studied in the areas of multi attribute decision making and multi-attribute utility theory (MUAT) (Wallenius et al., 2008; Dyer, 2005). However, the perspective of the study in this chapter differs from those studies in that we use attributes for the purpose of solving inverse optimization problems. Although this study is inspired by the concept of attributes in these areas, we did not compare our proposed inverse optimization method with the state of the art in MADM and MUAT (Siskos et al., 2016).

4.3 Attribute-Based Forward and Inverse Optimization

We consider a multi-attribute optimization problem where the decision maker (a user, consumer, system or any process capable of making decisions) is an optimizer, i.e., its decisions are results of solving an optimization problem. Below, we formulate both the forward and inverse optimization problems through an attribute-based perspective.

4.3.1 Attribute-Based Forward Optimization

Assume that we are studying an optimizing decision maker (DM) who is consistent in their preferences when making decisions over alternatives that are characterized by multiple attributes. The index set of alternatives/decisions $\mathcal{I}(t)$ and the index set of constraints $\mathcal{M}(t)$ change as a function of external parameter $t \in \mathcal{T}$ where $\mathcal{T}$ is a countable set (although this parameter can be a vector, we will assume, for simplicity, a scalar t). We define a family of forward attribute-based optimization problems of selecting the best set of alternatives from the set $\mathcal{I}(t)$ as:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i \in \mathcal{I}(t)} \mathbf{c}_i(\boldsymbol{\nu}_i, \mathbf{w})^T \mathbf{h}(x_i) \\ \text{subject to} \quad & g_m(\mathbf{x}) \leq 0 \qquad\qquad \forall m \in \mathcal{M}(t), \end{aligned} \tag{12}$$

where $\mathbf{x} = [x_1 \quad \ldots \quad x_{|\mathcal{I}(t)|}]$ is a vector of decision variables x_i corresponding to alternative i. Alternative i is described in terms of $|\mathcal{J}|$ attributes via the vector of attributes $\boldsymbol{\nu}_i = [\nu_{i1} \;\; \nu_{i2} \;\; \ldots \;\; \nu_{i|\mathcal{J}|}]$, where ν_{ij} represents the "quantity" of attribute j in alternative i. In many applications, ν_{ij} will be 0 or 1, i.e., alternative i will either possess or not possess attribute j. The vector-valued function $\mathbf{c}_i(\boldsymbol{\nu}_i, \mathbf{w})$ calculates the total attribute value corresponding to each option i based on the vector $\boldsymbol{\nu}_i$ and the weights $\mathbf{w}$. The vector-valued function $\mathbf{h}(x_i)$ consists of known basis functions. This forward formulation can be seen as an extension of the representation used by Keshavarz et al. (2011) where the forward objective function was assumed to be a weighted combination of basis functions; the main difference is that in our case, each basis function is a function of the attributes $\boldsymbol{\nu}_i$. For the purposes of this chapter, all functions in (12) are assumed convex and differentiable, although the idea of weights expressed in terms of attributes does not depend on this assumption. We also note that special cases of formulation (12) can be equivalent to a weighted sum of multiple objectives, in which case inverse multi-objective methods would be applicable.

Effectively, the attribute-based formulation (12) allows the set of alternatives to change from one t to another, while the objective function, expressed in terms of $\boldsymbol{\nu}$ and $\mathbf{w}$, stays the same (assuming the decisions are dependent on the same set of attributes in the same way). This characteristic of the formulation allows us to model situations where the set of options changes, e.g., over time.

4.3.2 Inverse Attribute-Based Optimization

The inverse optimization problem is to learn unknown parameters $\mathbf{w}$ given observations of $\mathbf{x}$, $\boldsymbol{\nu}_i$ and $\mathcal{I}(t)$; we will denote observations of $\mathbf{x}$ by a hat, e.g., $\hat{\mathbf{x}}$. The parametric structure of problem (12) allows us to use the imputed $\mathbf{w}$ for predicting future decisions given a new set of alternatives $\mathcal{I}(t^{\text{new}})$. We adapt the IO model of Keshavarz et al. (2011) to the attribute-based setting as shown in Equation (13). Instead of imputing the "weight" corresponding to each basis function (as was done by Keshavarz et al. (2011)), we are imputing the weight of each attribute.

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\lambda}} \quad & \sum_{k\in\mathcal{K}} \phi(\mathbf{r}^{stat^k}, \mathbf{r}^{compls^k}) \\
\text{s.t.} \quad & r^{stat^k}_{\ i} = \frac{\partial \mathbf{c}_i(\boldsymbol{\nu}_i, \mathbf{w})'\mathbf{h}(x_i)}{\partial x_i}\Big|_{\mathbf{x}=\hat{\mathbf{x}}^k} + \sum_{m=1}^{|\mathcal{M}(t)|} \lambda^k_m \frac{\partial g_m(\mathbf{x})}{\partial x_i}\Big|_{\mathbf{x}=\hat{\mathbf{x}}^k} && \forall i \in \mathcal{I}(t), k\in\mathcal{K} \\
& r^{compls^k}_{\ m} = -\lambda^k_m (g_m(\hat{\mathbf{x}}^k)) && \forall m \in \mathcal{M}(t), k\in\mathcal{K} \\
& \mathbf{w}\in\mathcal{W}, \quad \boldsymbol{\lambda} \geq \mathbf{0},
\end{aligned} \tag{13}$$

where $\hat{\mathbf{x}}^k$ is an observed decision vector, $k\in\mathcal{K}$, and $\mathcal{K}$ is the index set of observations; ϕ is a non-negative convex penalty function, and $\boldsymbol{\lambda}$ is the dual variable associated with KKT complementary slackness and stationarity residuals which are $\mathbf{r}^{compls}$ and $\mathbf{r}^{stat}$, respectively. This model will compute $\mathbf{w}$ such that past decisions become minimally sub-optimal solutions to model (12). A minimally sub-optimal decision in terms of the KKT optimality conditions is a decision for which the stationary and complementary slackness residuals are minimized (zero if exactly optimal). Due to the attribute-based nature of the model, given a new $\mathcal{I}(t)$, we are then able to predict DM's decisions by solving (12) with the imputed $\mathbf{w}$ and $\mathcal{I}(t^{\text{new}})$.

In multi-attribute decision-making problems, we may have prior information about the unknown weights of the attributes (e.g., pairwise comparison, ranking information, interval numbers, non-negativity, etc.). If we have such information, this information can be represented via the set $\mathcal{W}$ in the last line of problem (13); note that such a representation of assumptions on prior knowledge is also used by Chan et al. (2014b) and Keshavarz et al. (2011).

Classical IO models, such as the original model of Keshavarz et al. (2011), are used when the set of decisions (e.g., product choices) is static. In the next section, we will demonstrate that an attribute-based perspective facilitates the application of inverse optimization in assortment optimization, where product offerings, and hence the set of decisions a customer can make, change over time. In general, any existing inverse optimization method can be applied to an attribute-based forward formulation; we chose to demonstrate the value of the attribute-based perspective via the approach of Keshavarz et al. (2011) as it is easily applicable to parametric convex optimization problems.

4.4 Application in Assortment Optimization

Given past decisions of a consumer whom we assume to make near-optimal decisions (maximizing their utility function), we are interested in predicting their future purchases. Having this information enables firms and stores to plan for a better allocation of offerings (e.g., shelf items, advertisements, promotions) and deliver personalized offers and experiences in brick-and-mortar and online businesses. We specifically aim to predict decisions of users who aim to (implicitly) solve an optimization problem for

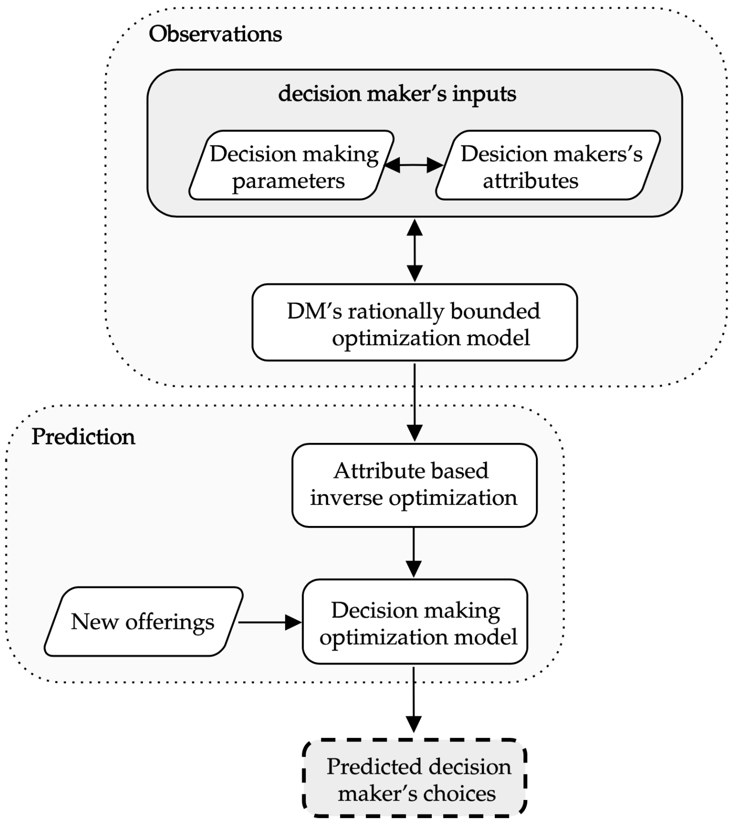

Observations
decision maker's inputs
Decision making parameters
Desicion makers's attributes
DM's rationally bounded optimization model
Prediction
Attribute based inverse optimization
New offerings
Decision making optimization model
Predicted decision maker's choices

Figure 8: Using inverse attribute-based optimization for applications in consumer behaviour given point of sale (POS) transaction data.

their decisions but may suffer from bounded rationality.

Figure 8 illustrates the data-driven perspective of this study. We assume that consumers make decisions based on a predefined set of attributes related to their alternatives. While consumers may know the attributes they are seeking, they may not be able to quantify the importance of one attribute over another. Given a consumer's past decisions, inverse attribute-based optimization imputes the weights, i.e., the importance of attributes. We then use the imputed weights in the forward optimization model representing the consumer's decision-making process to predict their future decisions. We can then use this information in various downstream uses

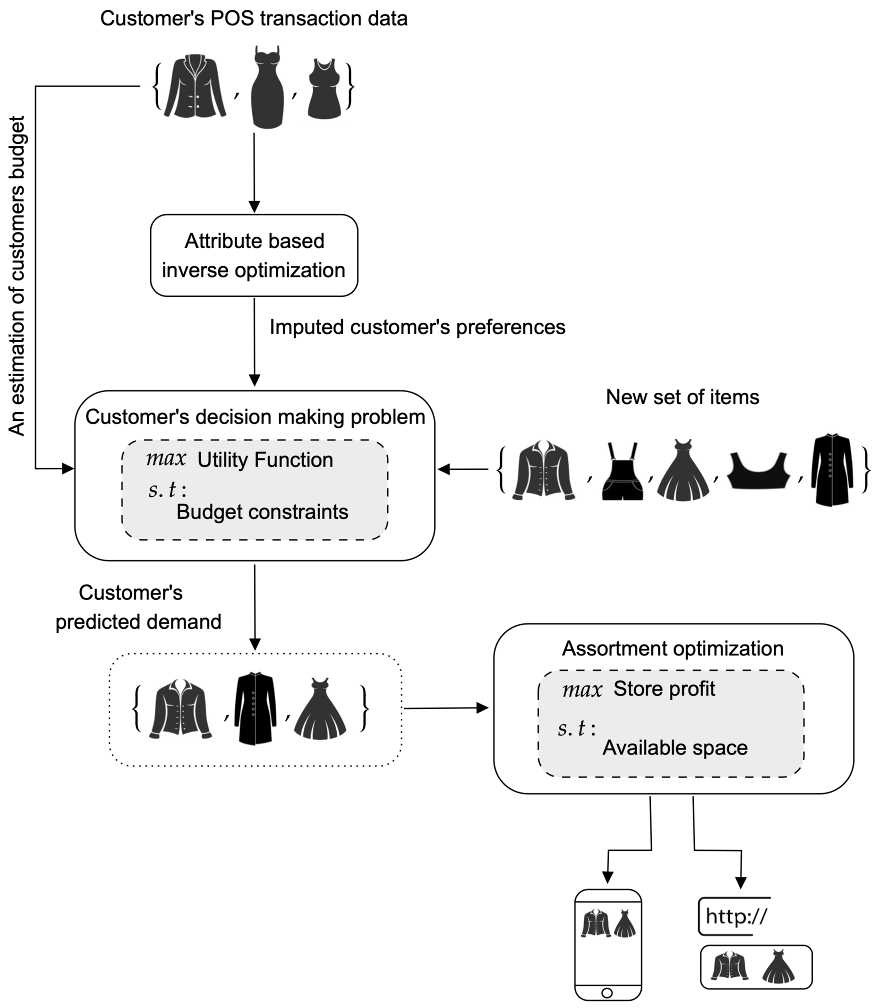

Customer's POS transaction data
An estimation of customers budget
Attribute based inverse optimization
Imputed customer's preferences
Customer's decision making problem
max Utility Function
s.t:
Budget constraints
New set of items
Customer's predicted demand
Assortment optimization
max Store profit
s.t:
Available space
http://

Figure 9: Using inverse attribute-based optimization for applications in online and in-app advertising for apparel industry.

such as recommending products to customers or planning better assortments.

4.4.1 An Illustrative Example: Online and In-App Advertising for Apparel Industry

Considering an apparel store, Figure 9 demonstrates how inverse attribute-based optimization can be used to offer consumers the best possible items matched to their taste. The store has access to the purchase history of each consumer through transaction data which represents customer decisions from each time they have shopped at the store. We assume that the set of items changes each time the customers shop, but the set of attributes stays the same. In this example, we define a relevant set of attributes in clothing shopping (e.g., size, colour, fabric pattern); inverse attribute-based optimization can then be used to impute the importance of these attributes. Having the importance of the attributes and an estimation of the consumer's budget from previous purchases enables us to solve a model that is an approximation of the consumer's decision-making problem. An attribute-based approach helps us understand how attributes contribute to the decisions and consequently to solve the consumer decision-making problem for a *new set of items*.

After predicting consumer preferences among the items in inventory, stores can offer these options to consumers. However, mobile applications and web banner displays have limited space in which only a few items could fit. Therefore, an assortment optimization problem needs to be solved to select the best items to put in the advertisement to maximize store profit.

4.4.2 Customer Forward Problem

We assume that an optimizing consumer solves an internal optimization problem maximizing their utility function while minimizing their total spending. The decisions are usually constrained by budget and non-negativity of the quantity of items bought by the consumer. In particular, we study an extension of the utility optimizing problem considered by Keshavarz et al. (2011) in problem (8):

$$\begin{aligned}
\textbf{trueFOP}(t, \mathbf{w}) : \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{i \in \mathcal{I}(t)} [p_i x_i - \sum_{j \in J} w_j \nu_{ij} (q x_i^2 + 2 r x_i)] \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}(t)} p_i x_i \leq B \qquad (14) \\
& \mathbf{x} \geq \mathbf{0},
\end{aligned}$$

where x_i is the share of item i acquired by the consumer, p_i is the price of each unit of item i, and B is the consumer's budget. $\sum_{j \in J} w_j \nu_{ij} (q x_i^2 + 2 r x_i)$ is an attribute-based quadratic utility function which captures the attribute-based utility of purchasing x_i units of product i, and where q and r are scaling parameters which are set to -0.01 and 1 in our experiments. Problem (14) is an attribute-based forward optimization model which is a special case of formulation (12) with $\mathbf{c}_i(\boldsymbol{\nu}_i, \mathbf{w}) = [(p_i - \sum_{j \in J} w_j \boldsymbol{\nu}_{ij} 2r) \quad - \sum_{j \in J} w_j \boldsymbol{\nu}_{ij} q] \; \forall i$ and $\mathbf{h}(x_i) = [x_i \quad x_i^2] \; \forall i$.

Formulation (14) represents the problem that would be solved by a truly rational and optimizing decision maker, i.e., a utility maximizer. In practice, however, customers may be affected by bounded rationality (Aswani et al., 2018; Esfahani et al., 2018). Since we did not have access to real consumer data, below we present a variant of (14) that we use for the purpose of generating synthetic data that is realistic, i.e., that is representing an optimizing consumer affected by bounded rationality. This formulation is based on the bounded rationality loss definition from the paper by

Esfahani et al. (2018):

$$\begin{aligned} \mathbf{FOP}(t, \mathbf{w}) : \min_{\mathbf{x}} \quad & \max \left\{ \sum_{i \in \mathcal{I}(t)} (p_i x_i - \sum_{j \in J} w_j \nu_{ij} (q x_i^2 + 2 r x_i)) - z^\star - \sigma, \quad 0 \right\} \\ \text{subject to} \quad & \sum_{i \in \mathcal{I}(t)} p_i x_i \leq B \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{15}$$

where σ is the degree of bounded rationality (which we assume to be fixed for each consumer) and $z^\star$ is the optimal value of problem (14). To be able to solve this problem more efficiently and generate suboptimal decisions we linearize model (15) using standard techniques and solve the linearized version.

We also note that in practice, we would not know the true objective function (or utility function) but rather would postulate a particular model, fit it and evaluate its goodness-of-fit; thus, in reality, the fitting an inverse optimization model should be embedded in a model selection framework (Chan et al., 2019). While in particular cases in practice, information about the true objective function might be available, in most of the problem such information is not available. The inverse optimization method proposed in this chapter assumes that only an approximation of the true objective function is available for inverse modeling.

4.4.3 Assortment Optimization

Given predictions of future consumer demand obtained by solving problem (14), we are able to solve an assortment optimization problem to offer products that are specifically tailored to consumer preferences. Doing so benefits both the consumer and the store, helping the consumer obtain a set of products matched to their taste and helping the store improve its sales and profit by stocking items that eventually will be bought.

Current practice for assortment optimization consists of the following steps: (a) selecting the items the firm should carry for the next season; (b) predicting consumer demand for the next season; and (c) deciding the corresponding order quantities regarding the firm's constraints (e.g., inventory, dollar budget). Addressing these steps varies from firm to firm. In practice, firms usually make decisions for (a) and (b) without a model and mostly based on past observations and expert judgment and intuition-based analysis (Farias et al., 2017). However, recently there has been an interest in developing predictive models. Finding the weights of attributes rather than alternatives enables inverse attribute-based optimization to provide necessary information to respond to (a) and (b) jointly by solving consumer decision-making model with imputed parameters coming from inverse optimization. After that, the predicted demand can be used to solve an assortment optimization problem to respond to (c).

Similar to Farias et al. (2017), we use the following model as the assortment optimization model:

$$\begin{aligned}
\mathbf{AO}(\mathbf{d},\mathbf{a}) : \max_{\mathbf{y}} \quad & \sum_{i\in\mathcal{I}(t)} e_i y_i & & \\
\text{subject to} \quad & \sum_{i\in\mathcal{I}(t)} b_i y_i \le Cap & & \text{[Budget Constraint]} \\
& y_i \le d_i \qquad \forall i \in \mathcal{I}(t) & & \text{[Demand Constraint]} \\
& y_i \le a_i \qquad \forall i \in \mathcal{I}(t) & & \text{[Inventory Constraint]} \\
& \mathbf{y} \ge \mathbf{0} \ \ \& \ \ \textit{integer}, & &
\end{aligned} \tag{16}$$

where y_i is the recommended number of item i that should be placed in the shelves, d_i

is the predicted consumer demand for item i from solving (14) with weights imputed by solving (13), a_i is the available inventory of item i, e_i is the revenue from selling each item i, b_i be the space item i occupies in the store and Cap the capacity of the store (could also be monetary capacity like budget).

4.5 Experimental Results

In this section, we present a set of experimental results that illustrates that attribute-based modelling combined with inverse optimization enables the prediction of optimizing customer behaviour in the context of assortment optimization, which was not possible with alternative-based modelling. Thus, these results can be seen as a proof-of-concept for attribute-based modelling in IO. Since direct comparison to alternative-based approaches is not possible, we compare our method to other methods that *are* applicable in attribute-based settings but are not IO models/methods. In this way, we capture the value of specifically *inverse* attribute-based *optimization* as opposed to other attribute-based approaches.

First, we evaluate the ability of IAO to predict future demand (including of items not present in the training set). Second, since the true value in this application is captured through the subsequent usage of the demand forecast, we evaluate the monetary value of the store's assortment regret as well as the utility gained by the consumer. We compare our proposed approach to three machine learning methods (Gaussian process regression, random forest and classification and regression tree (CART)) and four weight assignment methods from Stillwell et al. (1981) (equal weights, rank sum, rank reciprocal and rank exponent). This comparison allows us to capture the value of using specifically IO to predict the behaviour of optimizing agents when the set of alternatives is subject to change.

4.5.1 Comparison Methods

In this section we provide details of the weight assignment methods and the chosen machine learning methods.

Rank based weight assignment methods

Given J attributes, let's assume that the decision maker identified the weights of attributes as $w_1 \geq w_2 \geq ... \geq w_J$ and we have $\sum_{j=1}^{J} w_j = 1$ and $w_j \geq 0$. Stillwell et al. (1981) proposed two weight assignment methods, rank sum (RS) and rank reciprocal (RR). We chose these methods because when we are given with observations from near optimal decisions of users, we do not have prior knowledge about the distribution of the weights of the attributes. Therefore, we picked three methods that represent three different weight distribution. We note that when the prior knowledge about the distribution of the weights is available, the weight assignment methods should be based on the provided distribution. The rank based methods assign weights to the attributes based on the importance rank of the attributes. The importance rank of attributes, θ_j, is obtained by multiplying the user's matrix of decisions (i.e., the quantity of each item bought by the customer) by the items matrix (i.e., the matrix of items attribute values). The attribute j with the highest rank is considered to have $\theta_j = 1$, the attribute j with the second highest rank is assigned $\theta_j = 2$, etc. The rank sum approach assigns weights as follows:

$$w_j^{RS} = \frac{J - \theta_j + 1}{\sum_{j=1}^{J}(J - \theta_j + 1)}. \tag{17}$$

In RR, instead of considering the actual rank of the attributes, the inverted rank is formulated into the ranking method as:

$$w_j^{RR} = \frac{\frac{1}{\theta_j}}{\sum_{i=1}^{J} \frac{1}{\theta_i}}. \tag{18}$$

Another weight assigning method proposed by Stillwell et al. (1981) is rank exponent (REx) which is a generalization of the RS method as:

$$w_j^{REx} = \frac{(J - \theta_j + 1)^z}{\sum_{j=1}^{J} (J - \theta_j + 1)^z}, \tag{19}$$

where parameter $0 \leq z \leq 1$ reduces to equal weight when $z = 0$ and RS weights when $z = 1$. In our experiments, we consider $z = 0.3$.

Finally, we also use rank equal (REq) weights due to this method's simplicity, with weights defined as:

$$w_j^{REq} = \frac{1}{J}. \tag{20}$$

Machine learning methods

Random Forest:

Bootstrap aggregation or bagging is a method in machine learning that aggregates multiple learning models in order to reduce the prediction variance (Breiman, 1996). Given inputs $U = \{\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^K\}$ and targets $X = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^K\}$, bootstrap repeatedly selects a random subset of training data with replacement and fits the same learning model with the bootstrap-sampled data. In regression problems, it averages the results of the prediction. Random forests (Breiman, 2001) is a modified version of bootstrap that consists of multiple de-correlated regression trees. In regression problems random forest averages the prediction of all trees in the forest for the final prediction. The prediction function is $\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{p}^{\text{new}})$, where B is the total number of trees in the forest and f_b is a regression tree on the bth bootstrap sample.

Gaussian process regression:

Given inputs $U = \{\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^K\}$ and targets $X = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^K\}$, Gaussian Process Regression aims to learn a distribution of function $f(\mathbf{p}^k) + \epsilon^k = \mathbf{x}^k$ as $p(f)$ such that $f \sim \mathcal{GP}(\mu, \sigma)$ and $\epsilon_k \sim \mathcal{GP}(0, \sigma')$. Gaussian process regression defines a covariance of a prior distribution on the target function and uses the training data to define a likelihood function. It defines a posterior distribution over the target and uses its mean for the prediction (Rasmussen, 2004).

Decision tree regression:

As one of the most used algorithms in machine learning introduced by Breiman (1996), a decision tree can be used both for classification and regression problems. A decision tree builds a tree like prediction model that splits the input data through multiple rounds according to a certain cutting values in the data features. This creates multiple subsets with the last subset being called leaf nodes. To predict the outcome in each leaf node, the average outcome of the training data in this node is used.

4.5.2 Experimental Setup

We use the scikit-learn (Pedregosa et al., 2011) implementation of random forest, decision tree regression and Gaussian process regression. All optimization models are solved using *Concert Technology* of IBM ILOG CPLEX v.12.7.0. All experiments were run on an HP server with 20 Intel(R)Xeon(R) CPU E5-2687W v3 @ 3.10GHz processors and 512 GB RAM under Linux environment.

Following machine learning practices, we consider two sets of data: a training set and a test set. We use leave-one-out cross-validation on the training set for the machine learning methods. To do so, we pick only one point as our validation set from the whole data. We then build a model on all the remaining data and evaluate the single held out point error. A validation error estimate is obtained by repeating this procedure for each of the training points available and averaging the results. Doing so with several hyperparameter settings enables us to choose the hyperparameter setting with the lowest average cross-validation error. The experimental results presented below show the error on the test set for the best hyper-parameter setting found during the leave-one-out cross-validation process.

In this study we have used synthetic data to be able to easily assess the validity of the proposed approach and to be able to control the degree of optimality in the data. To do so, all the parameters of the problem (14) are drawn from uniform distributions. Algorithm (2) explains the training data generation process where $\hat{\mathbf{p}}^k$ and $\hat{\mathbf{x}}^k$ are the kth instances of prices and the corresponding decisions in the training data set. We generate random values for the parameters of the problem (14) from uniform distribution to get a global optimum decision and use this solution in problem (15) where $\sigma = z^\star \times \mathcal{S}$ where $\mathcal{S} \sim \text{Uniform}(0.1, 0.2)$. In fact, we consider $10\% - 20\%$ deviation for problem (15) solution from the global optimum value $z^\star$ due to the customer's bounded rationality. The quadratic concave utility function used for solving problems (14) and (15) is $U(\mathbf{x}) = -0.01\mathbf{x}^2 + 2\mathbf{x}$ (see Section 4.4.2) which satisfies the concavity and non-decreasing characteristics of a classic utility function. As we

Algorithm 2 Simulating optimal observations

```
1: generate $\boldsymbol{\nu} \sim Uniform[2,10]$ and $\mathbf{w}_{real} \sim Uniform[0,1]$.
2: $B \leftarrow 350$                                                ▷ Initialization
3: for $k \leftarrow 1$ to $K$ do
4:     generate $\hat{\mathbf{p}}^k \sim Uniform[5,25]$
5:     $z^\star \leftarrow trueFOP(\mathbf{w}_{real}, \boldsymbol{\nu}, \hat{\mathbf{p}}^k, B)$        ▷ Training Instances Generation
6:     solve $FOP(\mathbf{w}_{real}, z^\star, B)$ and get $\hat{\mathbf{x}}^k$
7:     $k \leftarrow k+1$
8: end for
9: $\mathcal{D} \leftarrow \{(\hat{\mathbf{x}}^1, \hat{\mathbf{p}}^1), (\hat{\mathbf{x}}^2, \hat{\mathbf{p}}^2), ..., (\hat{\mathbf{x}}^K, \hat{\mathbf{p}}^K)\}$
```

have assumed that the data represents an agent's rationally bounded decisions, we use problem (14) optimum solutions and solve problem (15) to mimic the behaviour of a customer affected by bounded rationality.

The same process is used with new value for the parameters to generate a test set for testing the performance of the proposed inverse attribute-based optimization. We define $\mathcal{G} = [\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, ..., \tilde{\mathbf{x}}^N]$ and $\mathcal{O} = [\tilde{\mathbf{p}}^1, \tilde{\mathbf{p}}^2, ..., \tilde{\mathbf{p}}^N]$, the optimal decisions and the corresponding prices, respectively, in the test data set where N is the size of the test set, which is 200 in our experiments.

Algorithm (3) explains the weight elicitation, demand prediction and assortment

Algorithm 3 Weight elicitation via IAO and weight assignments methods and assortment optimization

```
   $\mathbf{w} \leftarrow w\_elicitation\_method(\mathcal{D})$                  ▷ Imputing Unknown Weights
2: for $m \leftarrow 1$ to $M$ do
       solve $trueFOP(\mathbf{w}, \boldsymbol{\nu}^{new}, \tilde{\mathbf{p}}^m)$ and get $\mathbf{x}^m_{pred}$                  ▷ Prediction
4:     $m \leftarrow m+1$
   end for
6: $\mathbf{d} \leftarrow [\mathbf{x}^1_{pred}, \mathbf{x}^2_{pred}, ..., \mathbf{x}^M_{pred}]$
   $\mathbf{y} \leftarrow assort(\mathbf{d})$                                    ▷ Assortment Optimization
8: $regret \leftarrow R(\mathbf{y}, \mathcal{G})$
```

Algorithm 4 Weight elicitation via machine learning methods and assortment optimization

```
1: $ML\_method(\mathcal{D}).fit$                       ▷ ML training session
2: $\mathbf{d} \leftarrow ML\_method(\mathcal{O}).predict$    ▷ ML predicting session
3: $\mathbf{y} \leftarrow assort(\mathbf{d})$                ▷ Assortment Optimization
4: $regret \leftarrow R(\mathbf{y}, \mathcal{G})$
```

optimization process using inverse attribute-based optimization and weight assigning method. $w\_elicitation\_method$ represents inverse attribute-based optimization, rank sum, rank reciprocal, rank exponent and equal weights. After finding the value of unknown parameter $\mathbf{w}$ using one of these methods, the imputed $\mathbf{w}$ can be used to solve problem (14) to predict the upcoming demand for new items. Solving the assortment optimization problem considering new item demand, an optimum assortment decision is specified which can be assessed with the real demand by a regret function.

Algorithm (4) explains a similar process as Algorithm (3) for the machine learning methods. The function ML_method represents Gaussian process regression, random forest or regression decision tree. Unlike the methods in Algorithm (3), machine learning methods do not directly impute the value of parameter $\mathbf{w}$. They instead impute the value of a surrogate parameter which fits their own learning model to the data. Moreover, unlike the Algorithm (3) methods which require solving another optimization problem to predict the demand (which later will be used to compute the regret), machine learning methods do not need to solve any optimization problem, instead these methods use their own prediction model.

We use the attribute weights elicited by inverse attribute-based optimization, Gaussian process, random forest, regression decision tree and rank sum, rank exponential, rank reciprocal and equal weights in the agent's bounded rationality optimization problem (15) to finally estimate the arriving demand for a new set of items. Thereafter, we use the estimated demand in the assortment optimization problem (16) to find the recommended quantity of each item to be stocked in the store.

4.5.3 Results

Figure 10 shows the accuracy of the proposed IAO method (labelled as "att-IO" in the figure), Gaussian process, random forest, regression decision tree (CART) and the weight assignment methods including equal weights, rank sum, rank reciprocal and rank exponent in predicting the upcoming demand. We calculate the prediction error as MRE $= \frac{1}{N}\sum_{n=1}^{N}\frac{\|\mathbf{x}^n-\tilde{\mathbf{x}}^n\|_2}{\|\tilde{\mathbf{x}}^n\|_2}$ where N is the size of test set, which may include items that have not been seen in the training set; $\mathbf{x}^n$ and $\tilde{\mathbf{x}}^n$ are the predicted demand and the actual demand value of nth item.

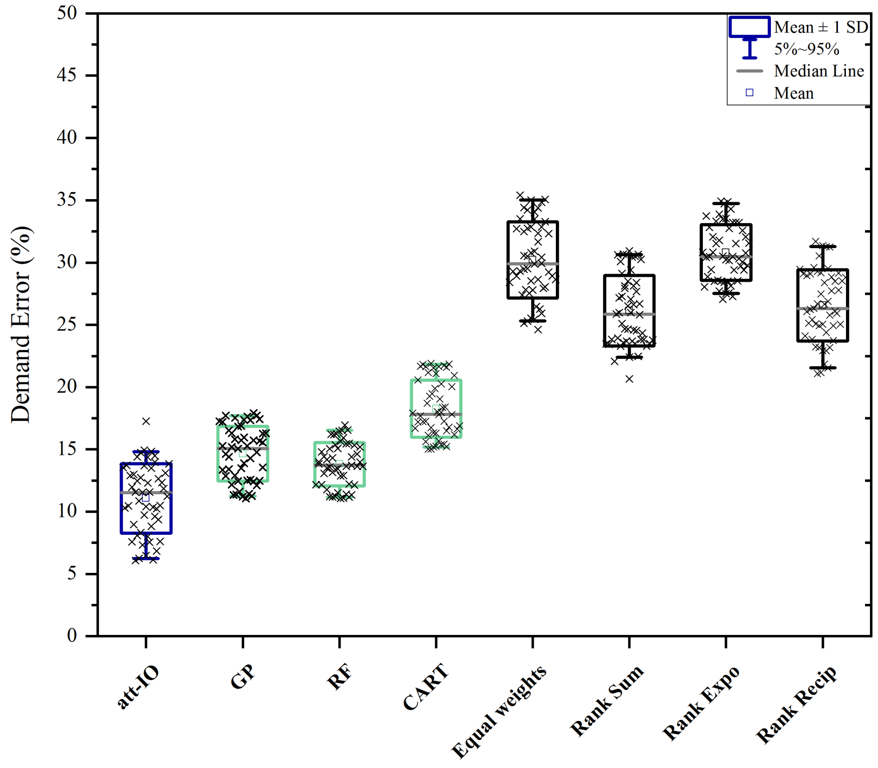


Figure 10: Demand prediction error for $\sigma \sim Uniform[10, 20]\%$ deviation from $z^\star$.

We can see that the prediction accuracy of inverse attribute-based optimization with around 12% mean relative prediction error is the lowest among the evaluated methods. This is due to the fact that IAO considers the consumer utility maximizing behaviour while other methods do not. We see that IAO and machine learning approaches have lower mean relative error as well as lower variance compared to the weight assignment methods.

We use the estimated demand in the assortment optimization problem (16) to find the recommended amount of each item to be stocked in the store. To compare the performance of the mentioned weight assignment methods in the assortment optimization problem, we define the following regret function:

$$R(t, \mathbf{y}, \tilde{\mathbf{x}}) = \sum_{i \in \mathcal{I}(t)} \Big[o_i(\max\{y_i - \tilde{x}_i, 0\}) + n_i(\max \tilde{x}_i - y_i, 0\}) \Big], \tag{21}$$

where R is the regret at time t, $\tilde{x}_i$ and y_i are the real demand and the available quantity of item i in the store, respectively. The proposed regret function captures both under-stock and over-stock in the store due to the inaccuracy in demand prediction with parameters $\mathbf{o}$ and $\mathbf{n}$. From the perspective of the store, Figure 11 shows that the store has lower assortment regret (defined by Equation (21)) when using inverse attribute-based optimization method. Similarly to Figure 10, machine learning methods are close to inverse attribute-based optimization and weight assigning methods have the largest assortment regret values.

Next, we consider the advantage of using inverse attribute-based optimization, machine learning algorithms and weight assigning methods in increasing the utility obtained by the consumer. To do so, we calculate the utility gained by consumers when the offered assortments are planned based on demands predicted by these methods.

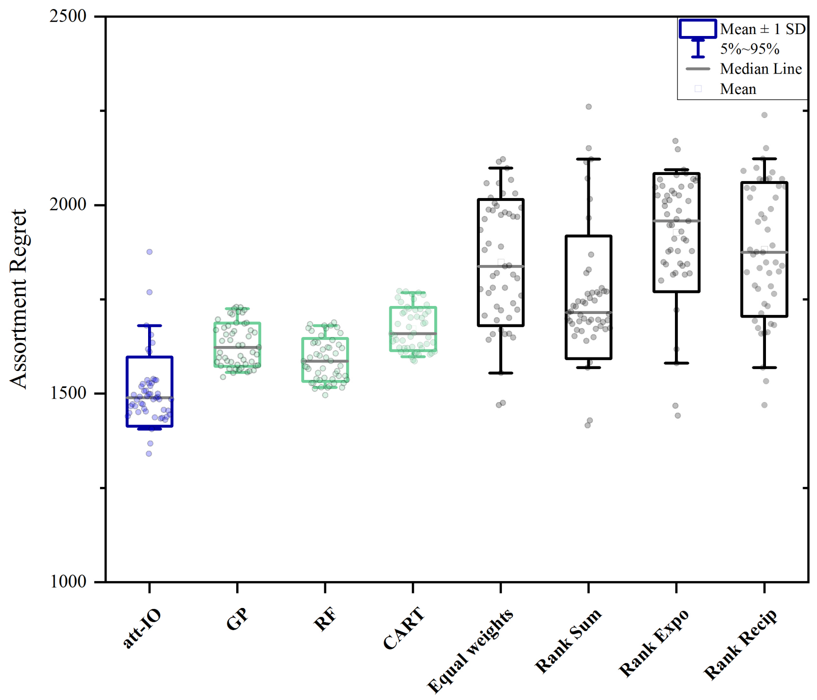


Figure 11: Store assortment regret for $\mathbf{o}$ and $\mathbf{n} \sim Uniform[5, 10]$.

Then, we solve the problem (15) with decision variable $\mathbf{x}$ bounded by the available items in the assortment and calculate the gained utility. We then calculate the utility when the quantity of items that can be bought by the consumer are not bounded by any assortment limits. We define the consumer's utility gap as $Utility\ Gap = \frac{|U^{\star} - U^{AO}|}{|U^{\star}|}$, where $U^{\star}$ is the global optimum utility and U^{AO} is the utility gained bounded by the offered assortment. Figure 12 demonstrates the consumer's utility gap for different demand estimation methods. We can see that IAO results in the lowest utility gap, followed by random forest, Gaussian process and decision tree regression. Similar to other figures, weight assigning methods underperform as compared to the IAO and machine learning methods. Comparing Figure 10 and 12, we can see that a small difference in demand prediction leads to a relatively bigger gap in consumer utility. For instance, an error in the demand prediction of about 5% between IAO and decision

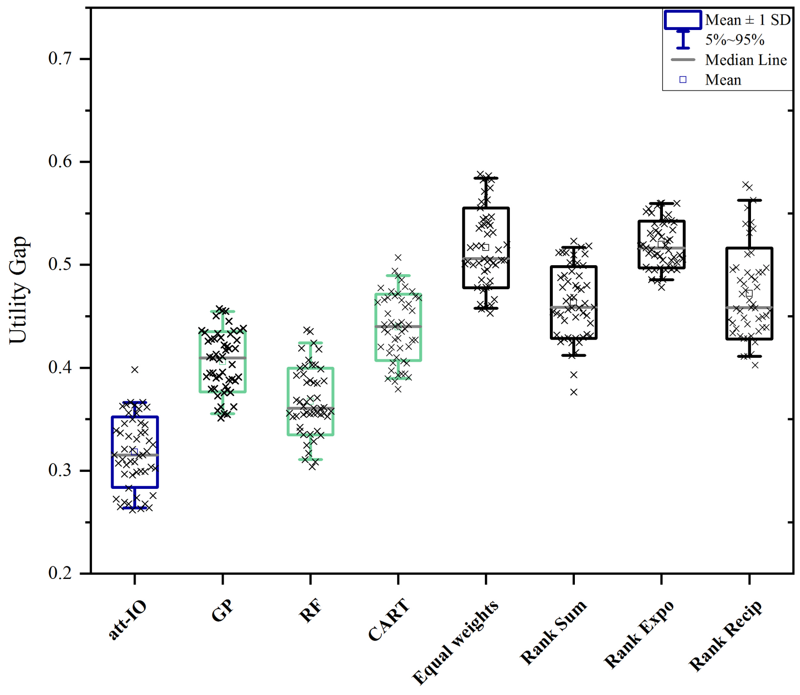


Figure 12: Consumer utility gap.

tree regression leads to a more than 10% difference in a consumer's utility gap.

Table 4 provides an overview of the computational times for IO and ML methods. We note that IAO does not have hyper-parameters that need tuning via cross validation, unlike the chosen ML methods. We also do not include the weight assignment methods in this table as training time was negligible and cross-validation was not required. As expected, the table demonstrates a trade-off between the IO and ML methods: IO is more time efficient during the training phase while ML is more time efficient

Table 4: Mean cross validation, training and prediction computational time (s) over 20 instances.

	att-IO	GP	RF	CART
Cross validation	NA	378.46	96.24	9.38
Training	<1	7.61	1.85	<1
Prediction	10.73	4.55	1.10	<1

during the prediction phase. The efficiency of IO training is mostly dependent on how efficiently the IO model can be solved – in our case it is convex and can be solved quickly. The efficiency of IO prediction is dependent on the complexity of the forward optimization model – in our experimental study, since the forward problem is convex, the run time is still reasonable; however, in general, the efficiency of the forward problem can hamper the use of the model for prediction purposes.

4.6 Conclusion

In this chapter, we consider the problem of predicting the behaviour of optimizers (e.g., optimizing consumers, systems) from observations of their choices (e.g., products, decisions) via inverse optimization. In contrast to the current usage of IO in alternative-based models, we introduce inverse attribute-based optimization, where each decision is described in terms of its attributes. This modelling approach expands the applicability of IO to settings where the set of alternatives (e.g., products, decisions) changes. We demonstrate the value of our approach in assortment optimization, an application where previous IO modelling could not be used due to the need to predict consumer behaviour given a new set of items. We compare inverse attribute-based optimization to other attribute-based methods, including rank-based methods and supervised machine learning, showing that using inverse attribute-based optimization leads to higher prediction accuracy, lower assortment regret for the store and a lower utility gap for the consumer. We believe that utilizing the attribute-based perspective can help make IO applicable in many more settings where there is a need to learn from observations of optimizing agents or systems.

Chapter 5

Inverse Satisfaction

Inverse optimization as a method for imputing optimization model parameters uses observations of minimally suboptimal decisions to learn such parameters. The suboptimality assumption of the observations is a key assumption in inverse optimization when the problem is to learn the optimizer's decision making problem. However, this assumption is not always a valid assumption, for instance, for satisficer decision makers. In this chapter, we relax the suboptimality assumption of the observations and propose an inverse method that imputes unknown parameters of a feasibility problem given observations of the feasible decisions of that problem. In analogy to inverse optimization, we call this problem inverse satisfaction. We mathematically prove that the proposed inverse satisfaction method is estimation consistent and show its estimation and prediction performance. We also compare the prediction performance of the proposed inverse satisfaction method with the inverse optimization method proposed by Keshavarz et al. (2011) and random forest from the machine learning literature on observations of optimizers and satisficers decisions.

5.1 Introduction

Inverse optimization is a method for imputing missing values of an optimization problem parameters given observations of that problem's minimally suboptimal decisions. IO methods have been developed for both optimal (Ahuja and Orlin, 2001) and suboptimal (Keshavarz et al., 2011; Bertsimas et al., 2012; Esfahani et al., 2018; Aswani et al., 2018) observations. Radiation therapy (Chan et al., 2019), auction mechanism (Beil and Wein, 2003) and electricity demand estimation (Saez-Gallego and Morales, 2017) are among some of the IO applications that have been studied in the literature.

Even though the applications of inverse optimization have been expanded through the development of models that incorporate noisy and near to optimal observations (Keshavarz et al., 2011; Aswani et al., 2018; Esfahani et al., 2018; Chan et al., 2019), there are still problems where the suboptimality assumption of the observations is not valid and classic inverse optimization methods cannot address such problems.

Motivated by the problem of learning user preferences from observations of their satisfactory (as opposed to optimal) decisions, we propose an inverse satisfaction model to impute parameters of user decision making problem given their *satisficing* decisions, i.e., decisions that satisfy an acceptable threshold rather than optimizing an objective function (Simon, 1979).

The assumption that decision makers (DM) make *rational* choices to gain the highest possible utility has been criticized because DMs generally possess limited cognition and effort to search through all possible alternatives (Simon, 1979). Although it is possible that DMs make optimal or near to optimal decisions, in practice DMs decision making paradigm may be similar to a heuristic search for finding a "good" choice rather than (explicitly or implicitly) examining all possible options in an optimization framework. They can have different rules for their search (e.g., lexicographic, elimination, conjunctive and disjunctive, etc.) (Gigerenzer and Todd, 1999), which can be incorporated into a mathematical representation of decision making problems.

As the current methods in the inverse optimization literature cannot necessarily capture a non-optimizer's behavior because such methods usually have optimality assumption about the observations, we propose an inverse satisfaction model to impute unknown parameters of a feasibility problem that represents a *satisficer*'s decision making problem. We relax the assumptions that the observations are optimal to a more realistic assumption of only feasibility. This assumption is different than the assumption that observations are noisy as in this assumption does not require near optimality of the observations and feasibility is the only condition. Doing so expands the applicability of the proposed method to settings where decision makers cannot necessarily find the optimal decision but still make feasible decisions, a setting with numerous applications in practice. Even though the proposed inverse satisfaction model assumes observations are feasible, optimal decisions can also be used as input as they are a special case of feasible decisions.

We assume that the satisficing decisions are defined over a pre-defined set of attributes to reach a certain acceptability level. In this setting, instead of optimizing an objective function (e.g., score function, utility function, etc.), the satisficing decisions should only satisfy an acceptable threshold (often referred to as aspiration level (Cyert et al., 1963)). The acceptable threshold is influenced by a user's socio-demographic characteristics (e.g., income) and similar to the importance of the attributes is user-specific. While the satisficing decisions are assumed to be available, the weight of the attributes attached to them and the user's acceptable threshold are usually unknown. Our proposed inverse satisfaction method imputes such weights jointly with the acceptable threshold given the user's satisficing decisions.

Chapter 4 has shown that the attribute-based modelling approach can expand the applicability of IO in practice. Therefore we use the attribute-based modeling in this chapter to impute the importance of attributes that make a decision satisficing. Such information can be further used in settings where decisions (satisficing or minimally suboptimal) change, but the underlying attributes of the decisions are the same. The connection of the inverse attribute based optimization to multi attribute decision making problems has been addressed in Chapter 4. In general, the forward model used in this chapter can be seen as an MADM model and the proposed inverse model imputes the weights of the decision attributes. While MADM studies have been more focused on solving the forward problems rather than solving the inverse problem, there have been methods for assigning weights to attributes (Stillwell et al., 1981; Keeney et al., 1993). Unlike the weight assignment methods in the literature, our proposed method takes into account that the decisions are the solutions of a feasibility problem.

Machine learning (ML) methods do a similar task of finding unknown parameters from observations. However, in problems usually addressed with machine learning, the observations are not coming from a constrained problem and observations are not usually assumed to be solutions to optimization or feasibility problems. While machine learning can be applied to the problem defined in this chapter, we conjecture that machine learning methods are not going to necessarily perform as well as methods that take the prior knowledge about the data generation problem into the consideration as the information about the data generation process is valuable prior knowledge.

Finding a function that separates two differently labeled sets of observations (i.e., satisficing and unsatisficing decisions) is a classification problem and methods such as support vector machines (SVM) can solve it. However, the problem defined in this chapter cannot be solved with SVM as the problem in this chapter has two levels of outputs (i.e., decisions regarding the number of items bought by the customer and whether the decisions are satisficing or unsatisficing). The first level corresponds to a regression problem and the second level corresponds to a classification problem. Therefore, SVM as a classification method cannot solve such problems as there is no mapping function from the inputs to the two leveled outputs. In fact, two different and independent mapping functions are needed for this problem, the first one to map the inputs to the actual decisions and a second mapping function to map decisions to their labels. Moreover, classification problems such as SVM do not provide information about the value of the acceptable threshold.

The main contributions of this chapter are:

1. We propose an inverse satisfaction method to jointly estimate the weight of the satisficer's decision attributes and their acceptable threshold given observations of their satisficed decisions. This setting has not been addressed previously.

2. We mathematically prove that the proposed inverse satisfaction model is estimation consistent, meaning with enough number of observations, the estimation of weights and acceptable threshold provided by the inverse satisfaction model converge to their true values.

3. We experimentally show that our proposed model is capable of learning from

both optimizers and satisficer's decisions compared to previous models in inverse optimization literature capable of learning only from optimizers.

The rest of the chapter is organized as follows. Section 5.2 provides a review on the literature. Section 5.3 defines the problem of interest and Section 5.4 mathematically define the forward and inverse problems. In Section 5.5 we mathematically prove the estimation consistency of the proposed inverse satisfaction method. In Section 5.6 we present and discuss the experimental results and finally Section 5.7 concludes this chapter.

5.2 Literature Review

Imputing optimization model parameters from observations of that model's optimal decisions was first introduced by the seminal paper of Burton and Toint (1992). In most of the studies in inverse optimization, unknown parameters are assumed to occur in the objective function and are imputed from optimal (Ahuja and Orlin, 2001; Iyengar and Kang, 2005) and near-optimal (Keshavarz et al., 2011; Chan et al., 2019; Aswani et al., 2018; Esfahani et al., 2018; Bärmann et al., 2017) decisions.

Multiple studies also investigated imputing unknown parameters occurring in the constraints of a linear program whether using a single observation (Chan and Kaw, 2020; Chow and Recker, 2012) or multiple observations (Tan et al., 2019; Ghobadi and Mahmoudzadeh, 2020b; Troutt et al., 2008). More related to our proposed framework are Tan et al. (2019) and Ghobadi and Mahmoudzadeh (2020b) where unknown parameters in both left and right hand sides of the constraints can be imputed. While these studies propose more general frameworks for the special case of inverse linear programming, the method proposed in this chapter does not have linearity assumption and is tailored to the specific problem of learning the behaviour of satisficing users.

The unknown parameters of user's decision making problems are usually utility function parameters that are commonly referred to as *user preferences*. The preference elicitation process for a user could be based on observations from that specific user (e.g., Aswani et al. (2018); Horvitz et al. (2013); Lee et al. (2004)) or from observations of other similar users (e.g., as in collaborative filtering (Konstan et al., 1997)). In this chapter, we focus on the problem of imputing preferences of one or a group of homogeneous users (with the same preference distribution) from observations of their own decisions.

The preference elicitation methods usually do not have an explicit assumption regarding whether the user is a satisficer or an optimizer, though they often implicitly assume that users try to maximize their utility function (Lahaie and Parkes, 2004; Braziunas and Boutilier, 2006). These methods often require a considerable amount of data which are usually costly to acquire and not always available. Chajewska et al. (2000) and Boutilier et al. (2006) investigated the elicitation effort in data querying and the impact it has on the elicitation accuracy and showed that there is a trade-off between elicitation accuracy and the effort spent on acquiring data.

The problem of learning constraint parameters has been studied in other areas such constraint programming (CP) and constraint satisfaction problem (CSP) (Lallouet et al., 2010; Bessiere et al., 2016; Beldiceanu and Simonis, 2016). However, the problem of interest in these areas is learning global and logic constraints with discrete decision variables. In this chapter we are interested in learning constraints of a feasibility problem (not global constraints used in constraint programming) and we do not have any specific assumption about the integrality of the decision variables.

To the best of our knowledge, this work is the first in the literature that uses inverse optimization principles to learn parameters of constraints that define the behaviour of satisficing users. The proposed inverse satisfaction model has lower error in estimating the preferences and in learning satisficing user behavior than inverse optimization and machine learning methods.

5.3 Problem Definition

Rational decision makers, i.e., optimizers, always make decisions that maximize their utility function (Simon, 1979). We represent an optimizer decision-making process as solving the following optimization problem:

$$\underset{\mathbf{x}\in\mathcal{X}}{max} \quad f(\mathbf{x}) \tag{22}$$

where $\mathbf{x}$ is a vector of their decision, $f(\mathbf{x})$ is a real valued function measuring their satisfaction and $\mathcal{X}$ is the feasible region specified by the user's constraints. In contrast to optimizers, satisficer decision makers avoid engaging in solving an optimization problem as searching for the best decision among all possible choices is a time-consuming and/or resource-intensive process. Instead, they settle for an "acceptable" decision. Satisficer users behavior has been mathematically modeled with different approaches such as goal programming (Ignizio, 1976), chance constraint programming (Charnes and Cooper, 1963) and robust optimization (Jaillet et al., 2016). In this chapter, we use a deterministic feasibility problem to represent a satisficer's decision making problem:

$$\underset{\mathbf{x}\in\mathcal{X}}{max} \quad \left\{\mathbf{0} \mid \mathcal{C}\right\} \tag{23}$$

where the aim is to find a decision vector $\mathbf{x}$ that is a feasible solution with respect to a set of hard constraints $\mathcal{X}$ and *satisfices*, i.e., meeting an acceptable threshold, the constraints $\mathcal{C}$.

Figure 13 contrasts the decision-making problems of optimizers and satisficers. Figure 13 (a) illustrates an optimizer's decision making problem where the polyhedron $\mathcal{X}$ represents the feasible region of decisions specified by the decision making constraints and $\mathbf{x}^\star$ is the optimum solution. Figure 13 (b) shows the same problem but for satisficer decision makers where an acceptable solution $\hat{\mathbf{x}}$ is found with respect to a constraint $\mathcal{C}$.

Problems in Figures 13 (a) and 13 (b) are the optimizer and satisficer forward optimization problems, respectively. The inverse problem can therefore be defined as: given observations of decisions of these problems, what are the value of unknown parameters of models (22) and (23). We define the satisficer's inverse problem, i.e.,

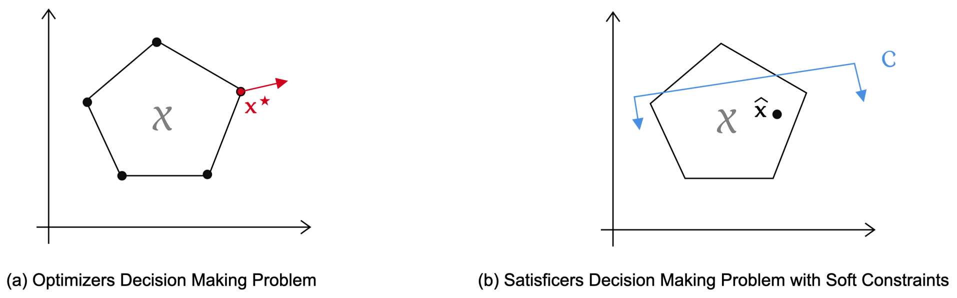
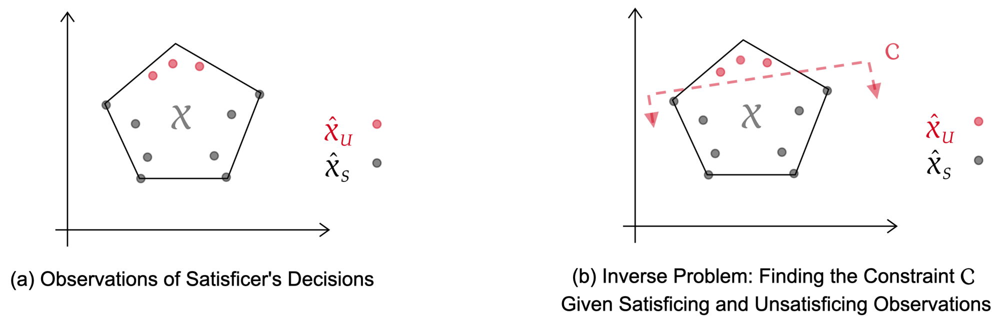

X
x*
C
X
x̂
(a) Optimizers Decision Making Problem
(b) Satisficers Decision Making Problem with Soft Constraints

Figure 13: Optimizer v.s. Satisficer Decision Making Problem



$\mathcal{X}$
$\hat{x}_u$
$\hat{x}_s$
(a) Observations of Satisficer's Decisions
C
(b) Inverse Problem: Finding the Constraint C Given Satisficing and Unsatisficing Observations

Figure 14: Satisficer's Inverse Problem

the inverse of problem shown in Figure 13 (a), as given the set of satisficed decisions $\hat{\mathcal{X}}_{\mathcal{S}}$ and the set of unsatisficed decisions $\hat{\mathcal{X}}_{\mathcal{U}}$ what is the constraint $\mathcal{C}$? In analogy to inverse optimization where the problem is to impute missing values of an optimization model parameters, we call this problem *inverse satisfaction* as this is the problem of imputing a satisfaction model parameters. Figure 14 shows the inverse satisfaction problem of the forward feasibility problem (23).

Unlike optimization problems whose solutions are optimal, any solution that satisfies the feasibility conditions is an acceptable solution for the satisfaction problems. However, users get different satisfaction (i.e., utility) values for different solutions. Though users cannot reveal their utility function parameters they can rank their choices based on the utility they get from them. Therefore, we assume that users can provide such ranking information. Figure 15 illustrates how the information and data obtained from users and observations of their decisions maps to the proposed inverse attribute based satisfaction method. We assume that we have access to the observations of both satisficing and unsatisficing offers by the user along with the ranking of the satisficing decisions obtained from a user's feedback. We use the inverse satisfaction method to impute the weights of decision attributes and use such

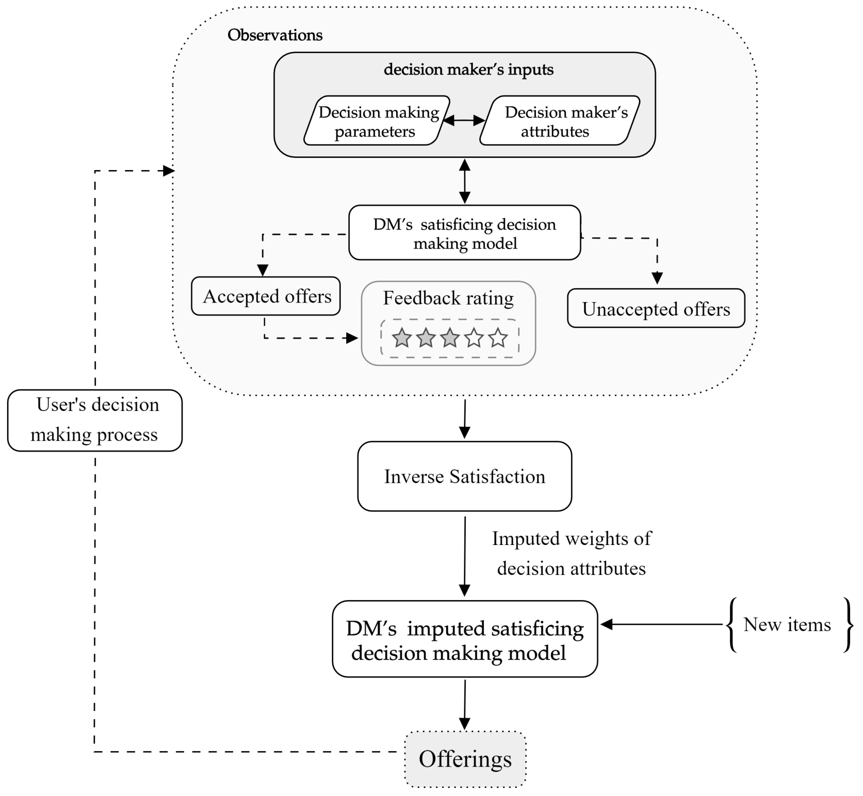

Observations
decision maker's inputs
Decision making parameters
Decision maker's attributes
DM's satisficing decision making model
Accepted offers
Feedback rating
Unaccepted offers
User's decision making process
Inverse Satisfaction
Imputed weights of decision attributes
DM's imputed satisficing decision making model
New items
Offerings

Figure 15: Problem Framework: data generation process, inverse satisfaction problem and the forward feasibility problem for new items.

parameters to solve the user's forward feasibility problem given a new set of items to suggest the best-matched items to the users in the next interaction with the user.

5.4 Modeling

In this chapter, we view two ends of the decision making strategies spectrum namely making random and optimized decisions and we assume that satisficing behavior lies between these two strategies. Mathematically this behavior can be interpreted as satisfying certain constraints. In this case, the user is not involved in optimizing functions of decisions such as maximizing utility functions or minimizing costs. Instead, an acceptable threshold for the decisions should be met in terms of the level of utility.

Below, we discuss the satisficing attribute based forward feasibility problem and the proposed inverse attribute based satisfaction model.

5.4.1 Satisficer Forward Problem

Assume that $\mathcal{I}$ is the set of alternatives and $\mathcal{J}$ is the set of attributes associated with the alternatives, consider the following problem:

$$\begin{aligned} FOP(\mathbf{w}, \boldsymbol{\nu}, \alpha) : \min_{\mathbf{x}} \quad & \mathbf{0}^T \mathbf{x} \\ \text{subject to:} \quad & g_m(\mathbf{x}) \leq 0 \qquad\qquad \forall m \in \mathcal{M} \\ & \sum_{i \in \mathcal{I}} f(x_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \geq \alpha \end{aligned} \tag{24}$$

where x_i is a decision variable corresponding to the recommended quantity of alternative i. Each alternative i can be described in terms of $|\mathcal{J}|$ attributes via the vector of attribute values $\boldsymbol{\nu}_i = [\nu_{i1}, \nu_{i2}, ..., \nu_{i|\mathcal{J}|}]$, where ν_{ij} represents the "quantity" of attribute j in alternative i. The function $c(\boldsymbol{\nu}_i, \mathbf{w})$ is a weighted aggregate function that calculates the total attribute value within each decision x_i based on the vector $\boldsymbol{\nu}_i$ and the attribute weights $\mathbf{w}$. This function is a general convex function. The function f captures the relation between the decision variables and the total attribute value and is a valuation function of decisions in attribute space. In the optimization problem (24), we can see that the objective function is a constant, meaning that no optimization process is involved in making decisions $\mathbf{x}$ and this problem is a feasibility problem. The functions $g_m(\mathbf{x})$ represent the decision making hard constraints such as customer's budget. The function $f(\mathbf{x}, c(\boldsymbol{\nu}, \mathbf{w}))$ represents the decision making criteria for which a threshold α should be met. In fact satisficers implicitly define an acceptable threshold in terms of their satisfaction represented by function $f(\mathbf{x}, c(\boldsymbol{\nu}, \mathbf{w}))$ while optimizers aim to optimize this function.

Problem (24) demonstrates the decision-making problem on a predefined set of items $\mathcal{I}$. However, the set $\mathcal{I}$ or the constraints may themselves be subject to change, dependent on an exogenous parameter t which changes the set of available alternatives as $\mathcal{I}(t)$ and constraints as $\mathcal{M}(t)$. A parametric version of the problem (24) is:

$$\begin{aligned} FOP(\mathbf{w}, \boldsymbol{\nu}, \alpha, t) : \min_{\mathbf{x}} \quad & \mathbf{0}^T \mathbf{x} \\ \text{subject to:} \quad & g_m(\mathbf{x}) \leq 0 \qquad \forall m \in \mathcal{M}(t) \\ & \sum_{i \in \mathcal{I}(t)} f(x_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \geq \alpha \end{aligned} \tag{25}$$

Formulation (25) allows the set of alternatives to change from one t to another. In practice, solving the forward problems (24) and (25) is not always possible as decision makers usually cannot specify the attribute weights $\mathbf{w}$ and information about the acceptable threshold α is not always attainable. On the other hand, in some applications (retail for instance) observations of the satisficing decisions and sometimes unsatisficing decisions are available. Therefore, we introduce inverse satisfaction to obtain the unknown parameters of problem (25) from observations of solutions to problem (25).

5.4.2 Satisficer Inverse Problem

In satisfaction problems any solution that satisfies the decision maker's constraints counts as a satisficing solution. However, users get a different level of satisfaction or utility for different solutions. Although users cannot reveal the weights of their decision attributes and their acceptable threshold, they usually can rank their choices based on the utility they get from them meaning that they can compare the satisfaction they get from making different decisions; which is called ordinal utility.

Recall that $\hat{\mathcal{X}}_{\mathcal{S}}$ is the set of satisficing decisions and $\hat{\mathcal{X}}_{\mathcal{U}}$ is the set of unsatisficing decisions. Let $\mathcal{D}^{true}$ be the set of observations with *true* labels of satisficing and unsatisficing defined based on the true forward feasibility problem (25) and $\mathcal{D}$ be the set of observations that are provided to the inverse satisfaction problem (32). We know that $\mathcal{D}^{true} = \hat{\mathcal{X}}_{\mathcal{S}}{}^{true} \cup \hat{\mathcal{X}}_{\mathcal{U}}{}^{true}$, $\mathcal{D} = \hat{\mathcal{X}}_{\mathcal{S}} \cup \hat{\mathcal{X}}_{\mathcal{U}}$ and $\hat{\mathcal{X}}_{\mathcal{S}}{}^{true} \cap \hat{\mathcal{X}}_{\mathcal{U}}{}^{true} = \emptyset$ and $\hat{\mathcal{X}}_{\mathcal{S}} \cap \hat{\mathcal{X}}_{\mathcal{U}} = \emptyset$. Considering $\mathcal{D}^{\bullet}$ and $\mathcal{D}^{\circ}$ as the sets of noisy and noise-free observations respectively, an observation $\hat{\mathbf{x}} \in \mathcal{D}^{\bullet}$ is a noisy observation if

$$\begin{cases} \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}}{}^{true} \quad \text{and} \quad \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}} \qquad \textit{or} \\ \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}}{}^{true} \quad \text{and} \quad \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}} \end{cases} \tag{26}$$

and observation $\hat{\mathbf{x}} \in \mathcal{D}^{\circ}$ is a noise-free observation if

$$\begin{cases} \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}}{}^{true} \quad \text{and} \quad \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}} \qquad \textit{or} \\ \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}}{}^{true} \quad \text{and} \quad \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}}. \end{cases} \tag{27}$$

In this chapter, we assume that the set of observations $\mathcal{D}$ only consist of noise-free observations (i.e., observations that are not mislabelled). Therefore, the sets of satisficing observations $\hat{\mathcal{X}}_{S}$ and unsatisficing observations $\hat{\mathcal{X}}_{U}$ are:

$$\hat{\mathcal{X}}_{\mathcal{S}} \subseteq \underset{\mathbf{x}}{\text{argmin}}\{\mathbf{0}^T\mathbf{x} \mid \sum_{i \in \mathcal{I}(t)} f(x_i, c(\boldsymbol{\nu}_i, \mathbf{w}_{true})) \geq \alpha_{true}\} \tag{28}$$

and

$$\hat{\mathcal{X}}_{\mathcal{U}} \subseteq \underset{\mathbf{x}}{\text{argmin}}\{\mathbf{0}^T\mathbf{x} \mid \sum_{i \in \mathcal{I}(t)} f(x_i, c(\boldsymbol{\nu}_i, \mathbf{w}_{true})) + \epsilon \leq \alpha_{true}\} \tag{29}$$

when $\mathbf{w}_{true}$ here is the parameter value that the users use to solve their forward problem to make their satisficing and unsatisficing decisions and ϵ is a small positive number. We define a set of preferential order of decisions based on the level of utility they provide with respect to the constraint $\sum_{i \in \mathcal{I}(t)} f(x_i, c(\mathbf{v}_i, \mathbf{w})) \geq \alpha$ defined as:

$$\begin{aligned} \mathcal{O}(\mathbf{w}) = \Big\{ (\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k) \ \mid \ & \sum_{i \in \mathcal{I}(t)} f(\hat{x}_i^r, c(\mathbf{v}_i^r, \mathbf{w})) \geq \\ & \qquad \sum_{i \in \mathcal{I}(t)} f(\hat{x}_i^k, c(\mathbf{v}_i^k, \mathbf{w})) \Big\}_{r,k \in \{1,\dots,|\hat{\mathcal{X}}_{\mathcal{S}}|\}, (r \neq k)} \end{aligned} \tag{30}$$

where $\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k \in \hat{\mathcal{X}}_{\mathcal{S}}$ and $\mathcal{O}(\mathbf{w}) \subseteq \hat{\mathcal{X}}_{\mathcal{S}} \times \hat{\mathcal{X}}_{\mathcal{S}}$. The ordered pair of $(\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k) \in \mathcal{O}(\mathbf{w})$ denotes that under the preference function f, the user gained the same or more satisfaction from decisions $\hat{\mathbf{x}}^r$ compared to decision $\hat{\mathbf{x}}^k$. We then define a loss function as:

$$\begin{aligned} l(\mathbf{w}) = \Big| \Big\{ (r,k) : \mathbb{1}\{(\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k) \in \mathcal{O}(\mathbf{w})\} \neq \mathbb{1}\{(\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k) \in \mathcal{O}(\mathbf{w}_{true})\} & \\ \forall r, k \in \{1, \dots, |\hat{\mathcal{X}}_{\mathcal{S}}|\}, (r \neq k) \Big\} \Big| & \end{aligned} \tag{31}$$

where $\mathbb{1}$ is an indicator function and $\mathbf{w}_{true}$ is the unknown true weights of the satisfaction constraint for which we already have the observed customer preferences. This function captures the number of pairs whose order with an imputed weight does not correspond to the real order with the true weights. We propose the following inverse attribute-based formulation for imputing the parameters of constraint $\mathcal{C}$:

$$\begin{aligned}
IS(\hat{\mathbf{x}}, \boldsymbol{\nu}) : \min_{\mathbf{w},\alpha} \quad & l(\mathbf{w})/2 \\
\text{subject to:} \quad & \sum_{i \in \mathcal{I}(t)} f(\hat{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \geq \alpha \quad \forall \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}} \\
& \sum_{i \in \mathcal{I}(t)} f(\hat{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) + \epsilon \leq \alpha \quad \forall \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}} \\
& \mathbf{w} \in \mathcal{W} \\
& \alpha \in \mathcal{A}
\end{aligned} \tag{32}$$

where $\hat{\mathbf{x}}$ is the vector of observed decisions and the sets $\mathcal{W}$ and $\mathcal{A}$ include any prior knowledge about the parameters $\mathbf{w}$ and α respectively (e.g., upper and lower bounds, normalized weights, ordinal ranking of weights, etc.).

Unlike inverse optimization methods that assume observations are minimally sub-optimal solutions of the forward problem, a feasibility problem's solutions do not have optimality characteristics and only are feasible. Therefore, in the inverse satisfaction model, we impose constraints on the feasibility and infeasibility of observations (see the first and second constraint of problem (32), respectively).

To capture the discrepancy loss function $l(\mathbf{w})$, there could be various formulation of problem (32) which could lead to different run time efficiency. We use the following model to solve this problem:

$$
\begin{aligned}
IS(\hat{\mathbf{x}}, \boldsymbol{\nu}) : \min_{\mathbf{w},\alpha} \quad & \sum_{q \in \{1,\ldots,|\mathcal{O}_{\mathbf{w}_{true}}|\}} z_q \\
& \text{subject to:} \\
& \sum_{i\in\mathcal{I}(t)} f(\hat{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \geq \alpha && \forall \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}} \\
& \sum_{i\in\mathcal{I}(t)} f(\hat{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) + \epsilon \leq \alpha && \forall \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}} \\
& \sum_{i\in\mathcal{I}(t)} f(\hat{x}^r_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \geq \sum_{i\in\mathcal{I}(t)} f(\hat{x}^k_i, c(\boldsymbol{\nu}_i, \mathbf{w})) - Mz_q && \forall (\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k) \in \mathcal{O}_{\mathbf{w}_{true}} \\
& \sum_{i\in\mathcal{I}(t)} f(\hat{x}^r_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \leq \sum_{i\in\mathcal{I}(t)} f(\hat{x}^k_i, c(\boldsymbol{\nu}_i, \mathbf{w})) + M - Mz_q && \forall (\hat{\mathbf{x}}^r, \hat{\mathbf{x}}^k) \in \mathcal{O}_{\mathbf{w}_{true}} \\
& \mathbf{w} \in \mathcal{W} \\
& \alpha \in \mathcal{A}
\end{aligned}
\tag{33}
$$

where $\mathcal{O}_{\mathbf{w}_{true}}$ is the ranking of the decisions from the satisficer decision maker, $\mathbf{z}$ is the vector of binary decision variables that specify the preferential order of observations and M is a big number. Note that $\mathcal{O}_{\mathbf{w}_{true}}$ does not contain any information about the parameter $\mathbf{w}_{true}$ and only has information about the ranking of the observations under $\mathbf{w}_{true}$.

5.5 Estimation Consistency

The proposed inverse satisfaction model provides an estimate for the forward feasibility problem parameter $\mathbf{w}$. We assume that $\mathbf{w}_{true}$ is the true value of $\mathbf{w}$ and the parameter value that a satisficer decision maker implicitly uses to make decisions. Based on Aswani et al. (2018) definition of estimation consistency which has been previously introduced by Bickel and Doksum (2015), the proposed inverse satisfaction model is estimation consistent if its estimate of $\mathbf{w}$ converges in probability to the true value $\mathbf{w}_{true}$. As inverse satisfaction model gives different estimates with different number of satisficing and unsatisficing observations, we use $\mathbf{w}^{kl}$ as the estimate of $\mathbf{w}$ by inverse satisfaction problem with k satisficing and l unsatisficing observations. Therefore:

Definition 1. The estimate $\boldsymbol{w}^{kl}$ is consistent if:

$$\boldsymbol{w}^{kl} \xrightarrow{p} \boldsymbol{w}_{true} \tag{34}$$

In the context of our problem, an estimate $\mathbf{w}^{kl}$ is consistent if with increasing the number of observations, k and l, the estimate $\mathbf{w}^{kl}$ obtained from the inverse satisfaction problem (32) approaches to $\mathbf{w}_{true}$ and its value converges to $\mathbf{w}_{true}$. We can restate Exp. (34) as:

$$\underset{k,l \longrightarrow \infty}{lim} \mathbb{P}(d(\mathbf{w}^{kl}, \mathbf{w}_{true}) > \sigma) = 0 \tag{35}$$

where d is any distance measure (e.g., Euclidean distance) and σ is any positive number.

In this proof we show Exp. (34) and Eq. (35) by showing that any sample path of the estimates $\mathbf{w}^{kl}$ converges to $\mathbf{w}_{true}$. To proceed with the proof of estimation consistency we make the following assumptions:

A1. Let $\mathcal{X}(\mathbf{w}, \boldsymbol{\nu})$ denote the forward problem feasible region. We assume that $\mathcal{X}(\mathbf{w}, \boldsymbol{\nu})$ is convex, bounded and has a non-empty interior, i.e, $\mathcal{X}(\mathbf{w}, \boldsymbol{\nu}) \neq \emptyset$. Therefore Slater's condition holds because the feasible region is non-empty.

A2. $\mathcal{W}$ and $\mathcal{A}$ are convex, bounded and non-empty and $\mathbf{w}_{true} \in \mathcal{W}$ and $\alpha_{true} \in \mathcal{A}$.

A3. $\hat{\mathcal{X}}_{\mathcal{S}} \neq \emptyset$ and $\hat{\mathcal{X}}_{\mathcal{U}} \neq \emptyset$.

A4. We assume that ϵ in problem (32) is a small enough number to ensure that any observation $\hat{\mathbf{x}} \in \hat{\mathcal{X}}_U^{true}$ violates the satisficing constraint as $\sum_{i\in\mathcal{I}(t)} f(\hat{x}_i, c(\mathbf{v}_i, \mathbf{w})) \geq \alpha + \epsilon$ and counts as an unsatisficing decision.

A1 states that the feasible region of the forward problem is non-empty which is necessary to ensure that the forward feasibility problem has a solution. **A2** is to ensure that the inverse satisfaction search space $\mathcal{W}$ and $\mathcal{A}$ are well-posed and contain the true values to ensure the possibility of estimation consistency of the inverse satisfaction model. **A3** is to ensure that there are observations available for the inverse satisfaction model. Finally, **A4** is to ensure that all observations are noise-free.

Following the notation of Aswani et al. (2018), we define notation for a sequential set as $[r] = \{1, 2, ..., r\}$. We define sequential sets $[|\hat{\mathcal{X}}_{\mathcal{S}}|] = \{1, 2, ..., |\hat{\mathcal{X}}_{\mathcal{S}}|\}$ and $[|\hat{\mathcal{X}}_{\mathcal{U}}|] = \{1, 2, ..., |\hat{\mathcal{X}}_{\mathcal{U}}|\}$ as the set of satisficing and unsatisficing observations, respectively. These sets are used as input the proposed inverse satisfaction model.

Following the notation of Aswani et al. (2018), we define $\mathcal{C}_{kl}(\hat{\mathbf{x}}, \epsilon)$ as the feasible region of the inverse optimization problem with k satisficing and l unsatisficing observations as:

$$\begin{aligned}\mathcal{C}_{kl}(\hat{\mathbf{x}}, \epsilon) = \{\mathbf{w} \in \mathcal{W}, \alpha \in \mathcal{A} \mid &\sum_{i\in\mathcal{I}(t)} f(\hat{x}_i^{k'}, c(\mathbf{v}_i^{k'}, \mathbf{w})) \geq \alpha, \\ &\sum_{i\in\mathcal{I}(t)} f(\hat{x}_i^{l'}, c(\mathbf{v}_i^{l'}, \mathbf{w})) + \epsilon \leq \alpha \quad \forall k' \in [k] \ \& \ \forall l' \in [l]\}\end{aligned} \tag{36}$$

We define the imputed $\mathbf{w}$ and α from solving the proposed inverse satisfaction problem (32) as:

$$\Theta_{kl} = \{\mathbf{w}^{kl}, \alpha^{kl} \mid (\mathbf{w}^{kl}, \alpha^{kl}) \in \underset{(\mathbf{w},\alpha)\in\mathcal{C}_{kl}}{\arg\min}\ l(\mathbf{w})/2\} \quad \forall k \in [|\hat{\mathcal{X}}_{\mathcal{S}}|]\ \&\ \forall l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|] \tag{37}$$

where $\mathbf{w}^{kl}$ and α^{kl} are the imputed values of parameters $\mathbf{w}$ and α from solving the inverse satisfaction problem with k satisficing and l unsatisficing observations, respectively. Note that the formulation (32) can have multiple solutions and Θ_{kl} can represent more than one pair of $\mathbf{w}^{kl}$ and α^{kl}. As Θ_{kl} is also a feasible solution of inverse satisfaction problem, we have:

$$\Theta_{kl} \in \mathcal{C}_{kl}(\hat{\mathbf{x}}, \epsilon) \quad \forall k \in [|\hat{\mathcal{X}}_{\mathcal{S}}|]\ \&\ \forall l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|] \tag{38}$$

Remark 1. As previously defined, we have assumed that we only deal with noise-free observations set of $\mathcal{D}^{\circ}$. We also assume that when we increase the size of the data set $\mathcal{D}^{\circ}$, we add new observations to the previous observations present in the data set.

Proposition 1. Suppose **A2** and **A3** hold. $\mathbf{w}_{true}$ and α_{true} are the feasible solutions for inverse satisfaction problem (32) with any subsets of any size of satisficing and unsatisficing observation sets:

$$(\mathbf{w}_{true}, \alpha_{true}) \in \mathcal{C}_{kl}(\hat{\mathbf{x}}, \epsilon) \quad \forall k \in [|\hat{\mathcal{X}}_{\mathcal{S}}|]\ \&\ \forall l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|].$$

Proof. The observations were originally generated by satisficers using $\mathbf{w}_{true}$ and α_{true} in their forward feasibility problem. The satisficing observations of the forward problem $\hat{\mathcal{X}}_{\mathcal{S}}$ are:

$$\hat{\mathcal{X}}_{\mathcal{S}} \subseteq \underset{\mathbf{x}}{\text{argmin}}\{\mathbf{0}^T\mathbf{x} \mid \sum_{i\in\mathcal{I}(t)} f(x_i, c(\boldsymbol{\nu}_i, \mathbf{w}_{true})) \geq \alpha_{true}\}. \tag{39}$$

And the unsatisficing observations are

$$\hat{\mathcal{X}}_{\mathcal{U}} \subseteq \underset{\mathbf{x}}{\text{argmin}}\{\mathbf{0}^T\mathbf{x} \mid \sum_{i\in\mathcal{I}(t)} f(x_i, c(\boldsymbol{\nu}_i, \mathbf{w}_{true})) + \epsilon \leq \alpha_{true}\}. \tag{40}$$

By the definition of the inverse satisfaction problem (32), the feasible region of this problem is:

$$\begin{aligned} &\sum_{i\in\mathcal{I}(t)} f(\hat{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \qquad \geq \alpha \quad \forall \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}} \\ &\sum_{i\in\mathcal{I}(t)} f(\hat{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) + \epsilon \leq \alpha \quad \forall \hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}} \\ &\mathbf{w} \in \mathcal{W} \\ &\alpha \in \mathcal{A} \end{aligned} \tag{41}$$

As mentioned in **Remark 1**, In this study, we assume that we do not have mislabeled observations in sets $\hat{\mathcal{X}}_{\mathcal{S}}$ and $\hat{\mathcal{X}}_{\mathcal{U}}$ (i.e., we do not have noisy observations). Therefore, $\mathbf{w}_{true}$ and α_{true} satisfy the first two constraints of the feasible region for all $\hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}}$ and $\hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{U}}$ as showed in Exp. (39) and (40). From **A2** we have $\mathbf{w}_{true} \in \mathcal{W}$ and $\alpha_{true} \in \mathcal{A}$. Therefore, $\mathbf{w}_{true}$ and α_{true} satisfy the last two constraints as well. Thus, we can conclude that:

$$(\mathbf{w}_{true}, \alpha_{true}) \in \mathcal{C}_{kl}(\hat{\mathbf{x}}, \epsilon) \quad \forall k \in [|\hat{\mathcal{X}}_{\mathcal{S}}|] \ \& \ \forall l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|].$$

□

In order to show the estimation consistency of formulation (32), we need to show that the added constraints to the inverse satisfaction model resulting from adding more satisficing and unsatisficing observations cannot remove $\mathbf{w}_{true}$ from the inverse satisfaction feasible region and that $\mathbf{w}^{kl}$ the estimate that the inverse satisfaction problem provides generally become closer to the $\mathbf{w}_{true}$ with adding more observations to the data set.

By the definition of inverse satisfaction problem (32), adding new observations to the sets $\hat{\mathcal{X}}_{\mathcal{S}}$ and $\hat{\mathcal{X}}_{\mathcal{U}}$ means adding more constraints to the inverse optimization problem. As having more constraints leads to a smaller feasible region, then we have nested convex subsets of feasible regions as:

$$\mathcal{C}_{11}(\hat{\mathbf{x}}, \epsilon) \supseteq \Big(\mathcal{C}_{21}(\hat{\mathbf{x}}, \epsilon),\ \mathcal{C}_{12}(\hat{\mathbf{x}}, \epsilon)\Big) \supseteq \mathcal{C}_{22}(\hat{\mathbf{x}}, \epsilon) \supseteq ... \tag{42}$$

Exp. (42) means that the sets $\mathcal{C}_{21}(\hat{\mathbf{x}}, \epsilon)$ and $\mathcal{C}_{12}(\hat{\mathbf{x}}, \epsilon)$ are both subsets of the set $\mathcal{C}_{11}(\hat{\mathbf{x}}, \epsilon)$ which are in turn subsets of $\mathcal{C}_{22}(\hat{\mathbf{x}}, \epsilon)$, and so on.

Theorem 1. *Suppose* **A2** *and* **A3** *hold. Given i.i.d sampling of observations, for any distance function* $d : \mathbb{R}^{|\mathcal{J}|} \longrightarrow \mathbb{R}^+$,

$$\exists\, (\mathbf{w}^{kl}, \alpha^{kl}) \in \mathcal{C}_{kl}(\hat{\mathbf{x}}, \epsilon) \quad \forall k \in [|\hat{\mathcal{X}}_{\mathcal{S}}|], l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|]$$

such that

$$d(\mathbf{w}^{1,1}, \mathbf{w}_{true}) \geq \Big(d(\mathbf{w}^{2,1}, \mathbf{w}_{true}), d(\mathbf{w}^{1,2}, \mathbf{w}_{true})\Big) \geq$$
$$... \geq d(\mathbf{w}^{|\hat{\mathcal{X}}_{\mathcal{S}}|,|\hat{\mathcal{X}}_{\mathcal{U}}|}, \mathbf{w}_{true}) \geq ... \geq 0. \tag{43}$$

Proof. We will prove Theorem 1 by induction.

Base case: Starting with $|\hat{\mathcal{X}}_{\mathcal{S}}| = 1$ and $|\hat{\mathcal{X}}_{\mathcal{U}}| = 1$, by adding another observation to either $\hat{\mathcal{X}}_{\mathcal{S}}$ or $\hat{\mathcal{X}}_{\mathcal{U}}$, from Exp.(42) we have:

$$\begin{aligned} &\mathcal{C}_{11}(\hat{\mathbf{x}}, \epsilon) \supseteq \mathcal{C}_{21}(\hat{\mathbf{x}}, \epsilon) \\ &\mathcal{C}_{11}(\hat{\mathbf{x}}, \epsilon) \supseteq \mathcal{C}_{12}(\hat{\mathbf{x}}, \epsilon) \end{aligned} \tag{44}$$

where subsets $\mathcal{C}_{11}(\hat{\mathbf{x}}, \epsilon)$, $\mathcal{C}_{12}(\hat{\mathbf{x}}, \epsilon)$ and $\mathcal{C}_{21}(\hat{\mathbf{x}}, \epsilon)$ are feasible regions of the decision variable $\mathbf{w}$. From Proposition (1) we know that adding more observations to the data set cannot remove $\mathbf{w}_{true}$ from the feasible regions of inverse satisfaction problem (32). Therefore as $\mathcal{W}$ is bounded based on assumption **A2** and it always contain $\mathbf{w}_{true}$, among the solutions of the inverse satisfaction problem (32) (in case of multiple solutions) there is at least one solution for which we have:

$$\begin{aligned} &d(\mathbf{w}^{1,1}, \mathbf{w}_{true}) \geq d(\mathbf{w}^{2,1}, \mathbf{w}_{true}) \\ &d(\mathbf{w}^{1,1}, \mathbf{w}_{true}) \geq d(\mathbf{w}^{1,2}, \mathbf{w}_{true}) \end{aligned} \tag{45}$$

Induction step: Assume the result holds for $|\hat{\mathcal{X}}_S| = n$ and $|\hat{\mathcal{X}}_U| = m$. Let $\hat{\mathbf{x}}^{n+1}$ and $\hat{\mathbf{x}}^{m+1}$ be new observations that are added to the set of satisficing and unsatisficing observations $\hat{\mathcal{X}}_{\mathcal{S}}$ and $\hat{\mathcal{X}}_{\mathcal{U}}$ respectively. By the definition of the inverse satisfaction problem (32), adding $\hat{\mathbf{x}}^{n+1}$ and $\hat{\mathbf{x}}^{m+1}$ to $\hat{\mathcal{X}}_{\mathcal{S}}$ and $\hat{\mathcal{X}}_{\mathcal{U}}$ is equivalent to adding the constraints $f(\hat{\mathbf{x}}^{n+1}, c(\boldsymbol{\nu}^{n+1}, \mathbf{w})) \geq \alpha$ and $f(\hat{\mathbf{x}}^{m+1}, c(\boldsymbol{\nu}^{m+1}, \mathbf{w})) + \epsilon \leq \alpha$ to the inverse satisfaction problem (32).

Adding such constraints removes some of the feasible solutions of the inverse satisfaction problem (32). However, the added constraints cannot remove $\mathbf{w}_{true}$ from the feasible region as it is always a feasible solution for all inverse satisfaction problem instances with any k and l from *Proposition* 1. Also, from Exp.(42) we know that the feasible region of the inverse satisfaction problem is a non-increasing nested convex set and from **A2** we know that $\mathcal{W}$ is bounded. Therefore the distance function defined on this non-increasing nested convex set is itself non-increasing. Therefore we have:

$$d(\mathbf{w}^{n,m}, \mathbf{w}_{true}) \geq \Big(d(\mathbf{w}^{n+1,m}, \mathbf{w}_{true}), d(\mathbf{w}^{n,m+1}, \mathbf{w}_{true})\Big) \geq 0. \tag{46}$$

Thus, by the principle of mathematical induction, Exp.(43) holds. □

Theorem 2. *Suppose* **A1***,* **A2***,* **A3** *and* **A4** *hold, given i.i.d sampling of observations, then we have: if* $|\hat{\mathcal{X}}_{\mathcal{S}}| \longrightarrow \infty$*, then* $\;\exists\, k \in [|\hat{\mathcal{X}}_{\mathcal{S}}|]\,,$ *for which*

$$d(\mathbf{w}^{kl}, \mathbf{w}_{true}) = 0 \qquad \forall l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|]. \tag{47}$$

Proof. Let's call the observations $\hat{\mathbf{x}} \in \hat{\mathcal{X}}_{\mathcal{S}}$ that satisfy the forward feasibility problem constraint in equality as $\sum_{i \in \mathcal{I}(t)} f(\hat{\mathbf{x}}_i, c(\boldsymbol{\nu}_i, \mathbf{w}_{true})) = \alpha_{true}$, *binding decisions.* Given the i.i.d sampling of the observations, when $|\hat{\mathcal{X}}_S| \longrightarrow \infty$ we are guaranteed to have binding observations in $\hat{\mathcal{X}}_S$. By solving the inverse satisfaction problem, we are finding estimates of $\mathbf{w}_{true}$ and α_{true} which are parameters of the unknown constraint $f(\mathbf{x}, c(\boldsymbol{\nu}, \mathbf{w}_{true}))$. Let $\tilde{f}_n$ be the estimate of function f with n binding observations. We know that $f(\mathbf{x}, c(\boldsymbol{\nu}, \mathbf{w}_{true}))$ is continuous and differentiable with respect to $\mathbf{x}$ for any $\mathbf{w}$ and $\boldsymbol{\nu}$. Because of **A1** and continuity of the function $f(\mathbf{x}, c(\boldsymbol{\nu}, \mathbf{w}_{true}))$, this function is bounded on its domain $\mathcal{X}$ and on its range. As the function $f(\mathbf{x}, c(\boldsymbol{\nu}, \mathbf{w}_{true}))$ is bounded both on its domain and its range and because of its differentiability we can say:

$$\exists n \in \mathbb{N} \text{ such that } \tilde{f}_n = f \tag{48}$$

Expression (48) is equivalent to:

$$\underset{|\hat{\mathcal{X}}_S|\longrightarrow\infty}{lim} \; d(\mathbf{w}^{kl}, \mathbf{w}_{true}) = 0 \qquad \forall l \in [|\hat{\mathcal{X}}_{\mathcal{U}}|]. \tag{49}$$

□

From **Theorem 1** and **Theorem 2**, we can say that the distance between $\mathbf{w}^{kl}$ and $\mathbf{w}_{true}$ has a non-increasing sequence with a tight lower bound of 0 for any sample path on the sets $\hat{\mathcal{X}}_S$ and $\hat{\mathcal{X}}_U$. Therefore we can conclude that:

$$\underset{k,l\longrightarrow\infty}{lim} \mathbb{P}(d(\mathbf{w}^{kl}, \mathbf{w}_{true}) > \sigma) = 0$$

And consequently:

$$\mathbf{w}^{kl} \xrightarrow{p} \mathbf{w}_{true}$$

□

We could take the same proof steps for α and show that inverse satisfaction problem (32) provides a consistent estimation for α too.

5.5.1 Insights into the Estimation Consistency of Inverse Optimization

We conjecture that the estimation consistency of inverse optimization models in general can be proven if these three conditions are met by the model and the data: i) the prior knowledge, if there is any, contains the true value of the unknown parameters, ii) the observations are noise-free, and iii) the inverse optimization model is designed in a way that adding new observations generates valid inequalities that reduce the search space of the unknown parameter of the forward model. The first condition is necessary since the prior knowledge about the value of the parameters defines the search space, if the search space does not contain the true values of the forward model unknown parameters then it is not possible to prove estimation consistency. In practice, the first condition cannot be verified. However, in cases where the prior knowledge contains a broad range for parameter values, or when the unknown parameter has some known properties (e.g., physical properties), the assumption that prior knowledge contains the true parameter values is valid. The second condition is not a necessary condition for proving estimation consistency in general, however in the presence of noise the inverse model needs some form of penalizing to compensate the effect of the noise in observations in estimating of the unknown parameters. Previous study on estimation consistency in inverse optimization by Aswani et al. (2018) investigated the estimation consistency of methods proposed by Keshavarz et al. (2011). They proved that the method proposed by Keshavarz et al. (2011) is not estimation consistent when observations of the decisions are corrupted by noise primarily because their choice of loss function minimizes residuals of optimality condition in objective function space rather than decision space. However, our experiments in Chapter 3 and this chapter suggest that the method proposed by Keshavarz et al. (2011) is estimation consistent when observations are noise-free. The third condition is also necessary condition to ensure that the inverse model converges to the true values and avoids converging to local optimums. To reach the true values of the parameters the inverse optimization model should be able to use observations to reduce some of the search space and if adding observations does not provide such cuts in the search space, there is no guarantee that the inverse optimization estimate of the unknown parameters converges to the true values.

5.6 Experimental Results

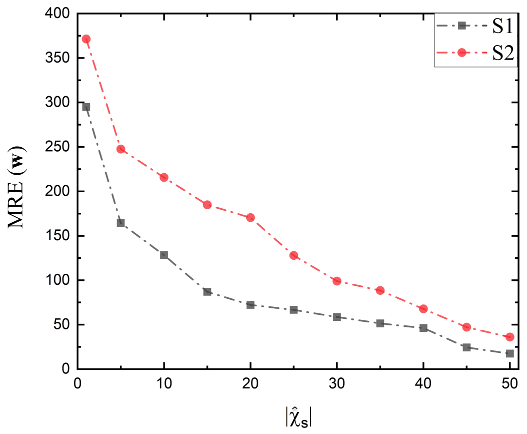
This section experimentally shows the performance of the proposed inverse satisfaction method in estimating the value of the unknown model parameters and predicting model decisions under multiple settings. In this section, we provide three sets of experiments which include the estimation accuracy of the inverse satisfaction method, the prediction accuracy of the inverse satisfaction method and the comparison of the proposed inverse satisfaction problem with the inverse optimization method of Keshavarz et al. (2011) and random forest (Breiman, 2001) from the machine learning literature for optimizer's and satisficer's decisions.

All the inverse satisfaction formulations are coded in C++ and all associated optimization problems are solved using *Concert Technology* of IBM ILOG CPLEX v.12.7.0. All experiments were run on an HP server with 20 Intel(R)Xeon(R) CPU E5-2687W v3 @ 3.10GHz processors and 512 GB RAM under a Linux environment. All the experiments were run on 30 randomly generated instances using POP toolbox (Oberdieck et al., 2016) to generate constraints. As we solve with feasibility problems in this study, we discarded the objective functions on POP generated instances. All the errors including mean relative error (MRE) and discrepancy rates are the average of all the instances. For all the experiments we chose linear f and c functions. $\mathbf{w}$ and $\boldsymbol{\nu}$ are drawn from uniform distribution as $\mathbf{w} \sim U[0,1]$ and $\boldsymbol{\nu} \sim U[5,20]$. To generate values of parameter α we solved the formulation (25) and optimized the function f and obtained f_{min} and f^{max} and generated α as $\alpha \sim U[f_{min}, f^{max}]$. The first and the third experiments show the prediction performance of the chosen models and the size of the training data is equal to the number of observations used in these experiments. The second experiment shows the prediction accuracy of inverse satisfaction model. Therefore, a test with 100 observations is used for each of the 30 samples.

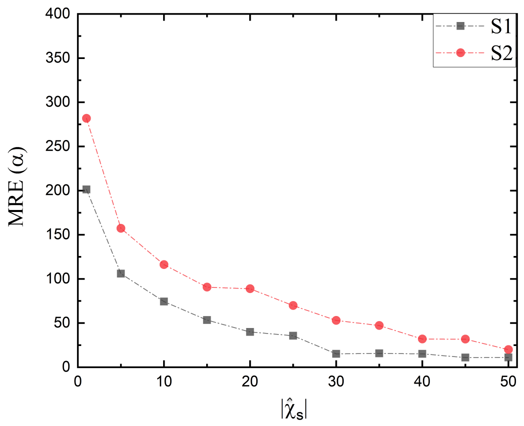
Experiment1: Estimation accuracy

In these experiments we show the accuracy in predicting the unknown parameters $\mathbf{w}$ and α by inverse satisfaction method for various problem sizes. Figure 16 shows how the performance of the proposed inverse satisfaction method changes when we increase the size of the sets of satisficing and unsatisficing observations. We show the performance of the inverse satisfaction method based on the size of the sets of the satisficing and unsatisficing observations for different decision size $\mathcal{I}$ and decision attributes size $\mathcal{J}$. As we mathematically proved in Section 5.5 that the proposed inverse satisfaction method is estimation consistent, its behavior demonstrated in Figure 16 is consistent with the mathematical proof in Section 5.5. Figure 16(a), 16(b), 16(c) and 16(d) show that with increasing the size of satisficing and unsatisficing observations in inverse modeling, the MRE of estimates of $\mathbf{w}$ and α decrease.

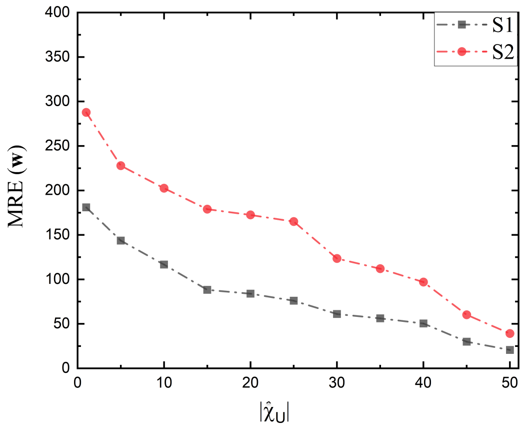
Figure 17 shows how increasing the size of the set of attributes $\mathcal{J}$ changes the estimation accuracy of $\mathbf{w}$ and α. This figure demonstrates the estimation accuracy of inverse satisfaction decreases when the number of observations is the same but problem size (i.e., the number of attributes) increases. Therefore, more observations are needed when there are more attributes involved in decision making.



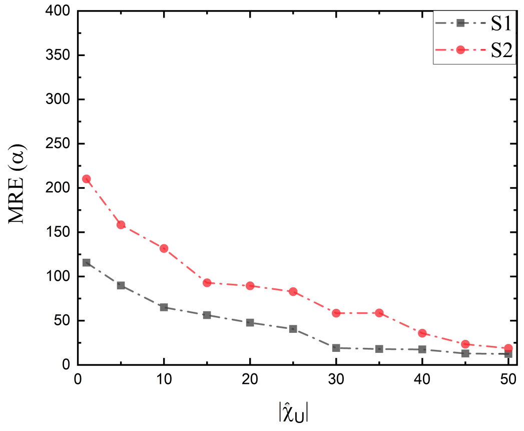
(a)



(b)



(c)



(d)

Figure 16: MRE($\mathbf{w}$) and MRE(α) v.s. the size of satisficing and unsatisficing observations where c is a linear function, $S1 = |\mathcal{I}| = 25, |\mathcal{J}| = 15$ and $S2 = |\mathcal{I}| = 20, |\mathcal{J}| = 30$. (a) and (b): $|\hat{\mathcal{X}}_{\mathcal{U}}| = 15$, (c) and (d): $|\hat{\mathcal{X}}_{\mathcal{S}}| = 15$.

Experiment 2: Prediction accuracy

In this experiment we show the accuracy of the proposed inverse satisfaction method in predicting the decisions. We used a test set of 100 decisions for each of the 30 samples we used in each problem size. Figure 18 shows the performance of the inverse satisfaction method in predicting users future decisions. Inverse satisfaction problem finds unknown parameters $\mathbf{w}$ and α of the users satisficing constraints $\sum_{i\in\mathcal{I}(t)} f(\mathbf{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w})) \geq \alpha$ of the problem (25), given observations of the users past

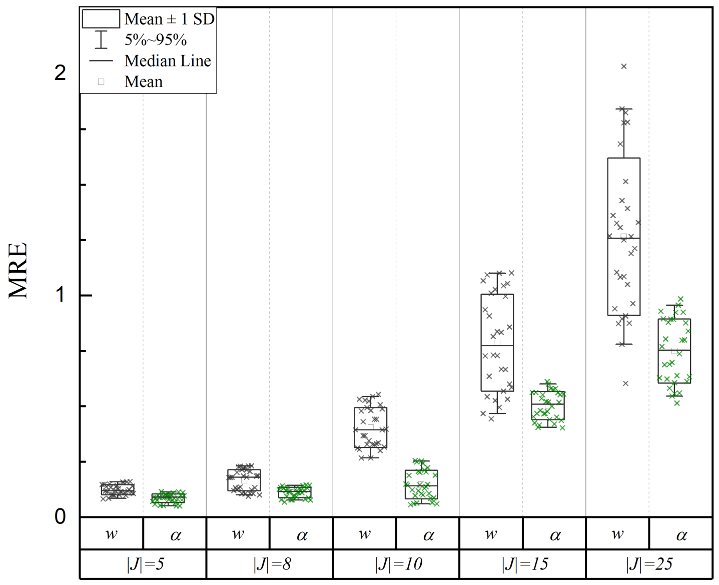


Figure 17: MRE($\mathbf{w}$) and MRE(α) v.s. the size of attributes $\mathcal{J}$ with $|\mathcal{I}| = 5, |\hat{\mathcal{X}}_{\mathcal{S}}| = 15$, $|\hat{\mathcal{X}}_{\mathcal{U}}| = 10$ and $|\mathcal{O}| = 10$ with linear c.

decisions. We use the imputed parameters $\mathbf{w}^{kl}$ and α^{kl} in the satisficing constraint as $\sum_{i\in\mathcal{I}(t)} f(\mathbf{x}_i, c(\mathbf{v}_i, \mathbf{w}^{kl})) \geq \alpha^{kl}$ with a new set of items to predict whether those items

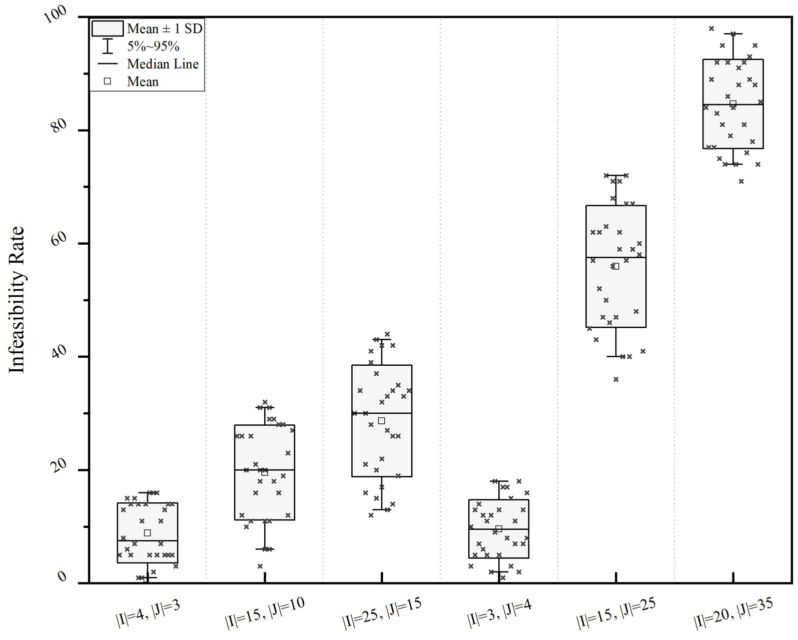


Figure 18: Infeasibility rate v.s. the size of attributes $\mathcal{J}$ with $|\hat{\mathcal{X}}_{\mathcal{S}}| = 15$, $|\hat{\mathcal{X}}_{\mathcal{U}}| = 10$ and $|\mathcal{O}| = 10$ with linear c.

are satisficing for the user (i.e., $\sum_{i \in \mathcal{I}(t)} f(\mathbf{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w}^{kl})) \geq \alpha^{kl}$) or unsatisficing (i.e., $\sum_{i \in \mathcal{I}(t)} f(\mathbf{x}_i, c(\boldsymbol{\nu}_i, \mathbf{w}^{kl})) \ngeq \alpha^{kl}$) and compare the estimated set of satisficing $\hat{\mathcal{X}}_{\mathcal{S}}$ and unsatisficing $\hat{\mathcal{X}}_{\mathcal{U}}$ decisions with $\hat{\mathcal{X}}_{\mathcal{S}}{}^{true}$ and $\hat{\mathcal{X}}_{\mathcal{U}}{}^{true}$. We capture the discrepancy as follows:

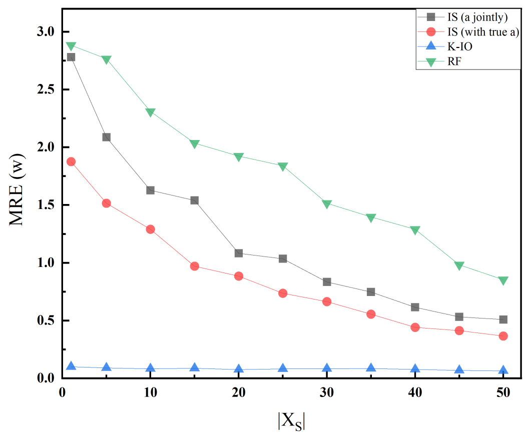
$$\text{Discrepancy Rate} = \frac{|\hat{\mathcal{X}}_{\mathcal{S}}{}^{true} \Delta \hat{\mathcal{X}}_{\mathcal{S}}| + |\hat{\mathcal{X}}_{\mathcal{U}}{}^{true} \Delta \hat{\mathcal{X}}_{\mathcal{U}}|}{|\hat{\mathcal{X}}_{\mathcal{S}}{}^{true}| + |\hat{\mathcal{X}}_{\mathcal{U}}{}^{true}|} \times 100 \tag{50}$$

where Δ is an operator that captures the symmetric difference of two sets.

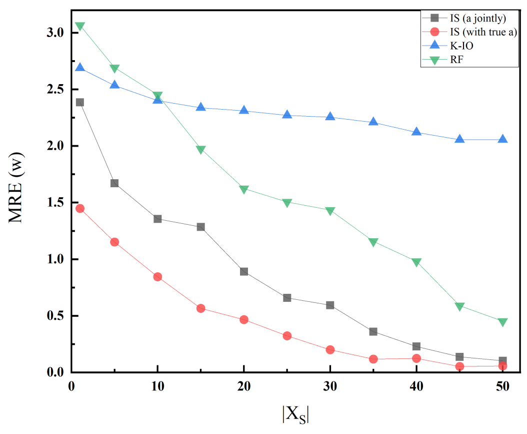
The discrepancy rate shows the predictive performance of the proposed inverse satisfaction method on test data. We see that with the same number of observations, the problems that have bigger decision attributes have higher discrepancy. Figure 18 suggests that when the size of the set of attributes is bigger, more observations are needed to obtain the same level of predictive performance.

Experiment 3: Learning from satisficer's and optimizer's decisions

In this experiment we investigate the learnability of inverse satisfaction given optimizer's and satisficer's decisions. We compare the performance of inverse satisfaction with the inverse optimization method proposed by Keshavarz et al. (2011). Moreover, since the problem we are considering falls within the realm of learning, we also compare these inverse methods with random forest regression from the machine learning literature. We used the scikit-learn citepscikit implementation of random forest. Figure 19 compares the performance of the inverse satisficing method with and without prior knowledge about the value of α_{true}, i.e., one that know the value of α and only estimates the value of $\mathbf{w}$ and one that estimates both $\mathbf{w}$ and α, Keshavarz et al. (2011) inverse optimization method (K-IO) and random forest (RF) from machine learning literature on observations from optimizers and satisficer users. We consider



(a) Optimizers



(b) Satisficers

Figure 19: Inverse satisficing (IS) method with and without prior knowledge about α_{true} compared to random forest (RF) and Keshavarz et al. (2011) inverse optimization (K-IO) method for optimizing and satisficing decisions for $|\mathcal{I}| = 15$, $|\mathcal{J}| = 10$ with $\hat{\mathcal{X}}_{\mathcal{U}} = \emptyset$, $\mathcal{O} = \emptyset$.

optimizers as an extreme case of satisficers. While satisficers want to make decisions to satisfy their utility function to a certain acceptable threshold, optimizers aim to optimize its value. Therefore, any optimized decision is a satisficed decision too. From Figure 19(a) we see that K-IO outperforms other methods. That is because K-IO has prior knowledge about the optimality of the decisions. We see that inverse satisfaction methods have lower MRE than random forest because even though inverse satisfaction methods do not consider the optimality of the decisions, they still fit the optimized observations to a feasibility problem. In contrast, random forest uses a decision tree model. Figure 19(b) shows that for satisficing decisions inverse satisfaction methods perform better than random forest and K-IO as both these methods use the wrong prior knowledge (i.e., optimality for K-IO and a tree-like decision model for random forest) for satisficing decisions. Comparing inverse satisfaction and inverse optimization methods performance in Figure 19 suggests that inverse satisfaction can learn from both satisficers and optimizers decisions (though with different rates). However, the inverse optimization method can only learn from the optimizer's decisions due to its assumption regarding the observations optimality.

In this section, we showed the performance of the proposed inverse satisfaction method under various configurations. In conclusion, inverse satisfaction in both estimating the value of unknown parameters and predicting the future decisions depend on the number of observations and the size of the set of attributes. We also showed the unlike the inverse optimization methods in the literature that can only learn from optimizer's decisions, the proposed inverse satisfaction method can learn from both optimizer's and satisficer's decisions.

5.7 Conclusion

In this chapter, we consider the problem of learning the behaviour of satisficers, i.e., decision makers that choose decisions to satisfy their utility up to an acceptable threshold, given observations of their satisficing and unsatisficing decisions. We proposed an inverse satisfaction model to estimate the attribute weights and satisficers acceptable threshold jointly. We assumed that satisficers solve a feasibility problem and proposed IS to estimate unknown parameters of a feasibility problem. We mathematically proved that our proposed method has estimation consistency, i.e., the estimation provided by IS converges to its real value in probability. We also experimentally showed the proposed method performance in estimating the value of feasibility decision model unknown parameters and predicting the model decisions under various settings. We showed that with increasing the size of satisficing and unsatisficing observations, the mean relative error (MRE) of the parameters estimation and discrepancy rate of the predicted decisions decreases. We also compared the performance of the proposed inverse satisfaction method with one inverse optimization method and one machine learning method (random forest). We showed that inverse satisfaction can learn from both satisficer and optimizer decisions as optimizers are a special case of satisficers. Inverse optimization method performance is better than inverse satisfaction on optimizer data. However, inverse optimization cannot learn from satisficers because of its assumption about the optimality of the observations. In contrast, as inverse satisfaction relaxes the optimality assumption of the data, it is capable of learning from satiisficer's decisions.

Chapter 6

Conclusion

6.1 Conclusion

In this dissertation, we study inverse optimization in two folds. First, we position inverse optimization as a learning method and identify the strengths and weaknesses of inverse optimization in the context of learning from parametric optimization problem solutions compared to out-of-the-box machine learning methods. Second, we address the challenges that inverse optimization methods suffer in practice when used for the purpose of prediction.

Chapter 3 connects inverse optimization and machine learning methods by positioning inverse optimization as a learning method and studies the performance of the inverse optimization method proposed by Keshavarz et al. (2011) compared to machine learning methods. We experimentally show that data generated from a more complex optimization process e.g., data generated with parametric optimization problems with piecewise objective function and more constraints, could be learned in a more data-efficient manner as inverse optimization considers the optimality of the data in the process of learning model parameters. We also identify the difficulties of the learning problem as (i) the training set size, (ii) the dependence of the optimization problem on external parameters, (iii) the level of confidence with regards to the correctness of the optimization prior, and (iv) the number of critical regions in the solution space.

Inverse optimization methods in the literature usually assume an alternative-based perspective in learning model parameters, i.e., learning the importance of alternatives/decisions. This assumption implicitly assumes that the set of decisions that inverse optimization learns from is fixed, making inverse optimization methods not applicable for settings where the set of decisions changes. In Chapter 4, we propose an attribute-perspective for inverse optimization modeling which imputes the importance of the attributes that make a decision optimal instead of imputing the weight of decisions over each other. The new attribute-based perspective provides a framework for looking at the decisions attributes, a more realistic and applicable approach in practice.

One main assumption in the inverse optimization literature is the optimality of the observations (with the exception of Boutilier et al. (2015) and Chan et al. (2014a) which consider infeasible observations as well). In Chapter 5, we relax the optimality assumption for the observations and propose an inverse satisfaction method for learning from feasible and infeasible solutions. The proposed inverse satisfaction method extends the inverse optimization literature to settings more general than before. Also, we mathematically prove that the proposed inverse satisfaction method provides statistically consistent estimation for the forward feasibility model parameters.

6.2 Future Work

Positioned as a learning problem, studying the statistical properties of inverse optimization problems can be a future research direction. Statistical measures similar to R^2 and properties such as consistency of the inverse optimization models have been previously studied by Chan et al. (2019) and Aswani et al. (2018); in Chapter 5 of this dissertation, we also established statistical consistency of the proposed inverse satisfaction problem. However, specific tests and measures for model-data fitness and model selection have not been studied which can be useful especially when dealing with big noisy data sets. Model selection is essential especially when the optimality (or sub-optimality) of observations is not guaranteed and inverse optimization is one of the proposed methods along with other learning methods such as supervised machine learning.

Inverse optimization as a parameter estimation method has been studied only for point estimation (i.e., estimating a single value for the parameters). However, another direction for future studies could be to expand inverse optimization as an interval estimation method when an acceptable interval for the parameters is estimated to keep observations minimally suboptimal. The only related study to interval estimation is done by Chassein and Goerigk (2018) where a variable-sized uncertainty is considered for the parameters. They focused on finding the largest possible size of uncertainty for which a nominal solution still remains optimal for the forward linear robust optimization problem.

So far, inverse optimization models have been more focused on offline and batch settings (except for Bärmann et al. (2017, 2020); Dong et al. (2018)). Given the data efficiency of inverse optimization methods which is showed in Chapter 3, inverse optimization could be an ideal learning method for online learning. The online setting that has been previously studied by Bärmann et al. (2017, 2020); Dong et al. (2018) is also well compatible with the proposed attribute-based perspective of Chapter 4. Therefore, another future direction for inverse optimization could be to study inverse attribute based optimization methods that learn parameters in an online fashion.

Another issue with real data in practice is that data can be incomplete and only partial observations of decisions be available. Inverse optimization with partial observations has been briefly addressed in the literature by Kim and Kececioglu (2008) and Yang and Zhang (2007). However, this setting can still be extended to more general frameworks and models such as convex optimization problems. Also, inverse optimization with partial observations has applications in learning user's parameters where obtaining complete observations is usually costly or not possible in practice. The computational complexity of learning from partial observations in comparison to complete observations could be another direction for the future research in inverse optimization.

Bibliography

M. Abdellaoui. Parameter-free elicitation of utility and probability weighting functions. *Management Science*, 46(11):1497–1512, 2000.

P. Ahmadi-Moshkenani, T. A. Johansen, and S. Olaru. Combinatorial approach toward multiparametric quadratic programming based on characterizing adjacent critical regions. *IEEE Transactions on Automatic Control*, 63(10):3221–3231, 2018.

R. K. Ahuja and J. B. Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001.

T. Ajayi, T. Lee, and A. J. Schaefer. Objective selection for cancer treatment: An inverse optimization approach. *Optimization Online*, 2020.

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

A. Aswani, P. Kaminsky, Y. Mintz, E. Flowers, and Y. Fukuoka. Behavioral modeling in weight loss interventions. *European Journal of Operational Research*, 272(3):1058–1072, 2019a.

A. Aswani, Z.-J. M. Shen, and A. Siddiq. Data-driven incentive design in the medicare shared savings program. *Operations Research*, 67(4):1002–1026, 2019b.

A. Babier, J. J. Boutilier, M. B. Sharpe, A. L. McNiven, and T. C. Chan. Inverse optimization of objective function weights for treatment planning using clinical dose-volume histograms. *Physics in Medicine & Biology*, 63(10):105004, 2018.

A. Babier, T. C. Chan, T. Lee, R. Mahmood, and D. Terekhov. An ensemble learning framework for model fitting and evaluation in inverse linear optimization. *Informs Journal on Optimization*, 3:119–226, 2021.

A. Bärmann, S. Pokutta, and O. Schneider. Emulating the expert: Inverse optimization through online learning. In *International Conference on Machine Learning*, pages 400–410, 2017.

A. Bärmann, A. Martin, S. Pokutta, and O. Schneider. An online-learning approach to inverse optimization. *arXiv preprint arXiv:1810.12997v2*, 2020.

F. H. Barron. Limits and extensions of equal weights in additive multiattribute models. *Acta Psychologica*, 68(1-3):141–152, 1988.

D. Beil and L. Wein. An inverse-optimization-based auction mechanism to support a multiattribute RFQ process. *Management Science*, 49(11):1529–1545, 2003.

N. Beldiceanu and H. Simonis. Modelseeker: Extracting global constraint models from positive examples. In *Data Mining and Constraint Programming*, pages 77–95. Springer, 2016.

D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

D. Bertsimas, V. Gupta, and I. Paschalidis. Inverse optimization: A new perspective on the Black-Litterman model. *Operations Research*, 60(6):1389–1403, 2012.

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

C. Bessiere, A. Daoudi, E. Hebrard, G. Katsirelos, N. Lazaar, Y. Mechqrane, N. Narodytska, C.-G. Quimper, and T. Walsh. New approaches to constraint acquisition. In *Data mining and constraint programming*, pages 51–76. Springer, 2016.

P. J. Bickel and K. A. Doksum. *Mathematical statistics: basic ideas and selected topics, volume I*, volume 117. CRC Press, 2015.

C. Boutilier. A pomdp formulation of preference elicitation problems. In *AAAI/IAAI*, pages 239–246. Edmonton, AB, 2002.

C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8-9): 686–713, 2006.

J. J. Boutilier, T. Lee, T. Craig, M. B. Sharpe, and T. C. Y. Chan. Models for predicting objective function weights in prostate cancer IMRT. *Medical Physics*, 42(4):1586–1595, 2015.

D. Braziunas and C. Boutilier. Preference elicitation and generalized additive utility. In *AAAI*, 2006.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

A. Bulut and T. Ralphs. On the complexity of inverse mixed integer linear optimization. Technical report, Tech. rep. COR@ L Laboratory Report 15T-001-R3, Lehigh University, (cit. on p. 147), 2015.

E. Bunduchi and I. Mandric. Parametric linear programming, 2011. URL http://demonstrations.wolfram.com/ParametricLinearProgramming/.

D. Burton and P. L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53(1-3):45–61, 1992.

D. Burton and P. L. Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63(1-3):1–22, 1994.

W. N. Caballero, D. Ríos Insua, and D. Banks. Decision support issues in automated driving systems. *International Transactions in Operational Research*, 2021.

U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, pages 363–369, 2000.

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

T. C. Chan and T. Lee. Trade-off preservation in inverse multi-objective convex optimization. *European Journal of Operational Research*, 270(1):25–39, 2018.

T. C. Chan, T. Craig, T. Lee, and M. B. Sharpe. Generalized inverse multiobjective optimization with application to cancer therapy. *Operations Research*, 62(3):680–695, 2014a.

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

T. C. Y. Chan, T. Craig, T. Lee, and M. B. Sharpe. Generalized inverse multi-objective optimization with application to cancer therapy. *Operations Research*, 62(3):680–695, 2014b.

A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations research*, 11(1):18–39, 1963.

A. Chassein and M. Goerigk. Variable-sized uncertainty and inverse problems in robust optimization. *European Journal of Operational Research*, 264(1):17–28, 2018.

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

R. M. Cyert, J. G. March, et al. A behavioral theory of the firm. *Englewood Cliffs, NJ*, 2(4): 169–187, 1963.

R. M. Dawes and B. Corrigan. Linear models in decision making. *Psychological bulletin*, 81 (2):95, 1974.

C. Dong and B. Zeng. Expert learning through generalized inverse multiobjective optimization: Models, insights, and algorithms. In *International Conference on Machine Learning*, 2020.

C. Dong, Y. Chen, and B. Zeng. Generalized inverse optimization through online learning. In *Advances in Neural Information Processing Systems*, pages 86–95, 2018.

H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, pages 155–161, 1997.

C. W. Duin and A. Volgenant. Some inverse optimization problems under the hamming distance. *European Journal of Operational Research*, 170(3):887–899, 2006.

J. S. Dyer. Maut—multiattribute utility theory. In *Multiple criteria decision analysis: state of the art surveys*, pages 265–292. Springer, 2005.

P. Egri, T. Kis, A. Kovács, and J. Váncza. An inverse economic lot-sizing approach to eliciting supplier cost parameters. *International Journal of Production Economics*, 149: 80–88, 2014.

M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

F. Eisenführ, M. Weber, and T. Langer. *Rational decision making.* Springer, 2009.

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

V. F. Farias, S. Jagabathula, and D. Shah. Building optimized and hyperlocal product assortments: A nonparametric choice approach. *Available at SSRN 2905381*, 2017.

R. Fernández-Blanco, J. M. Morales, S. Pineda, and Á. Porras. Ev-fleet power forecasting via kernel-based inverse optimization. *arXiv preprint arXiv:1908.00399*, 2019.

B. Gebken and S. Peitz. Inverse multiobjective optimization: Inferring decision criteria from data. *Journal of Global Optimization*, ( ):1–27, 2020.

K. Ghobadi and H. Mahmoudzadeh. Inferring linear feasible regions using inverse optimization. *European Journal of Operational Research*, 290(3):829–843, 2020a.

K. Ghobadi and H. Mahmoudzadeh. Multi-point inverse optimization of constraint parameters. *arXiv preprint arXiv:2001.00143*, 2020b.

K. Ghobadi, T. Lee, H. Mahmoudzadeh, and D. Terekhov. Robust inverse optimization. *Operations Research Letters*, 46(3):339–344, 2018.

G. Gigerenzer and P. M. Todd. Fast and frugal heuristics: The adaptive toolbox. In *Simple heuristics that make us smart*, pages 3–34. Oxford University Press, 1999.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

S. Greco, B. Matarazzo, and R. Slowinski. Rough sets methodology for sorting problems in presence of multiple attributes and criteria. *European journal of operational research*, 138 (2):247–259, 2002.

S. Greco, J. Figueira, and M. Ehrgott. *Multiple criteria decision analysis*, volume 37. Springer, 2016.

A. B. Hempel, P. J. Goulart, and J. Lygeros. Inverse parametric optimization with an application to hybrid system control. *IEEE Transactions on Automatic Control*, 60(4): 1064–1069, 2014.

E. J. Horvitz, J. S. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *arXiv preprint arXiv:1301.7385*, 2013.

J. P. Ignizio. *Goal programming and extensions.* Lexington Books, 1976.

G. Iyengar and W. Kang. Inverse conic programming with applications. *Operations Research Letters*, 33(3):319–330, 2005.

P. Jaillet, S. D. Jena, T. S. Ng, and M. Sim. Satisficing awakens: Models to mitigate uncertainty. *Optimization Online*, 2016.

R. L. Keeney, H. Raiffa, and R. F. Meyer. *Decisions with multiple objectives: preferences and value trade-offs.* Cambridge university press, 1993.

A. Keshavarz, Y. Wang, and S. Boyd. Imputing a convex objective function. In *Intelligent Control (ISIC), 2011 IEEE International Symposium on*, pages 613–619. IEEE, 2011.

E. Kim and J. Kececioglu. Learning scoring schemes for sequence alignment from partial examples. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(4): 546–556, 2008.

A. G. Kök and M. L. Fisher. Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research*, 55(6):1001–1021, 2007.

A. G. Kök, M. L. Fisher, and R. Vaidyanathan. Assortment planning: Review of literature and industry practice. In *Retail supply chain management*, pages 99–153. Springer, 2008.

J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

A. Kovács. Parameter elicitation for consumer models in demand response management. In *2019 1st Global Power, Energy and Communication Conference (GPECOM)*, pages 456–460. IEEE, 2019.

S. M. Lahaie and D. C. Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 180–188, 2004.

A. Lallouet, M. Lopez, L. Martin, and C. Vrain. On learning constraint problems. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 45–52, 2010. doi: 10.1109/ICTAI.2010.16.

G. Lee, S. Bauer, P. Faratin, and J. Wroclawski. Learning user preferences for wireless services provisioning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 480–487, 2004.

A. M. Mármol, J. Puerto, and F. R. Fernández. The use of partial information on weights in multicriteria decision problems. *Journal of Multi-Criteria Decision Analysis*, 7(6):322–329, 1998.

M. Moghaddass and D. Terekhov. Inverse integer optimization with an imperfect observation. *Operations Research Letters*, 48(6):763–769, 2020.

M. Naghavi, A. A. Foroughi, and M. Zarepisheh. Inverse optimization for multi-objective linear programming. *Optimization Letters*, 13(2):281–294, 2019.

R. Oberdieck, N. Diangelakis, M. Papathanasiou, I. Nascu, and E. Pistikopoulos. POP – Parametric Optimization Toolbox. *Industrial & Engineering Chemistry Research*, 55(33): 8979–8991, 2016. doi: 10.1021/acs.iecr.6b01913. URL https://doi.org/10.1021/acs.iecr.6b01913.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

E. N. Pistikopoulos, M. C. Georgiadis, and V. Dua. *Multi-Parametric Programming*, volume 1. 2011.

C. Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

J. Roland, Y. De Smet, and J. R. Figueira. Inverse multi-objective combinatorial optimization. *Discrete applied mathematics*, 161(16-17):2764–2771, 2013.

S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

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

A. Schaefer. Inverse integer programming. *Optimization Letters*, 3(4):483–489, 2009.

Z. Shahmoradi and T. Lee. Quantile inverse optimization: Improving stability in inverse linear programming, 2020.

H. A. Simon. Rational decision making in business organizations. *The American economic review*, 69(4):493–513, 1979.

Y. Siskos, E. Grigoroudis, and N. F. Matsatsinis. Uta methods. In *Multiple criteria decision analysis*, pages 315–362. Springer, 2016.

W. G. Stillwell, D. A. Seaver, and W. Edwards. A comparison of weight approximation techniques in multiattribute utility decision making. *Organizational behavior and human performance*, 28(1):62–77, 1981.

Y. Tan, A. Delong, and D. Terekhov. Deep inverse optimization. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 540–556. Springer, 2019.

Y. Tan, D. Terekhov, and A. Delong. Learning linear programs from optimal decisions. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

A. Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903. ACM, 2005.

O. Tavaslıoğlu, T. Lee, S. Valeva, and A. J. Schaefer. On the structure of the inverse-feasible region of a linear program. *Operations Research Letters*, 46(1):147–152, 2018.

M. D. Troutt. A maximum decisional efficiency estimation principle. *Management Science*, 41(1):76–82, 1995.

M. D. Troutt, S. K. Tadisina, C. Sohn, and A. A. Brandyberry. Linear programming system identification. *European journal of operational research*, 161(3):663–672, 2005.

M. D. Troutt, A. A. Brandyberry, C. Sohn, and S. K. Tadisina. Linear programming system identification: The general nonnegative parameters case. *European Journal of Operational Research*, 185(1):63–75, 2008.

J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management science*, 54(7):1336–1349, 2008.

L. Wang. Cutting plane algorithms for the inverse mixed integer linear programming problem. *Operations Research Letters*, 37(2):114–116, 2009.

X. Yang and J. Zhang. Partial inverse assignment problems under l1 norm. *Operations research letters*, 35(1):23–28, 2007.

C.-H. Yeh. A problem-based selection of multi-attribute decision-making methods. *International Transactions in Operational Research*, 9(2):169–181, 2002.

K. P. Yoon and C.-L. Hwang. *Multiple attribute decision making: an introduction.* Sage publications, 1995.