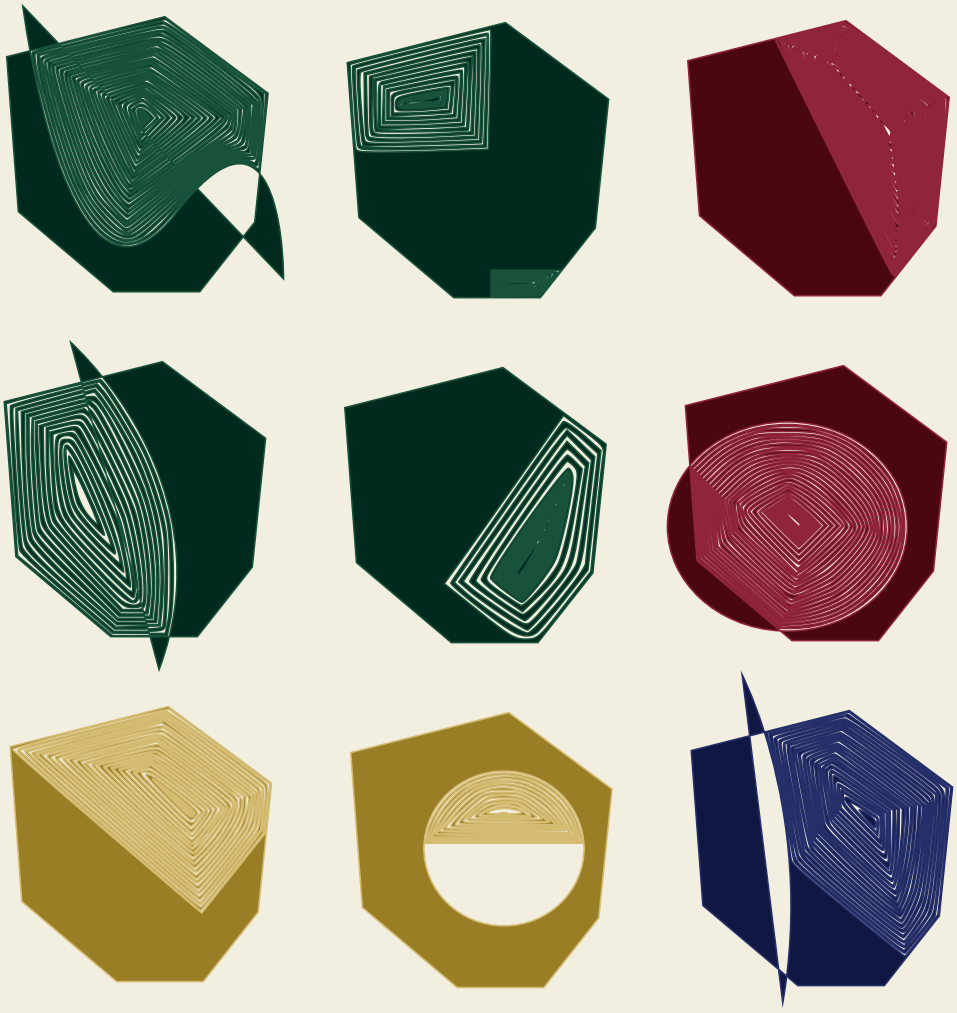
Optimization with constraint learning

Maragno, D.

Publication date
2025
Document Version
Final published version

OPTIMIZATION WITH CONSTRAINT LEARNING



Donato Maragno

Optimization with Constraint Learning

Donato Maragno

Printed by ProefschriftMaken
Cover design Michele Carlucci
ISBN 978-94-6510-512-3

The research described here and the publications of this dissertation were made possible by support from the Dutch Scientific Council (NWO) through the grant OCENW.GROOT.2019.015, Optimization for and with Machine Learning (OPTIMAL).

Optimization with Constraint Learning

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 4 april 2025, te 14.00 uur

door Donato Maragno
geboren te Matera

Promotiecommissie

Promotores:	prof. dr. ir. D. den Hertog	Universiteit van Amsterdam
	prof. dr. S.I. Birbil	Universiteit van Amsterdam
Overige leden:	prof. dr. J.A.S. Gromicho	Universiteit van Amsterdam
	prof. dr. D. Bertsimas	Massachusetts Institute of Technology
	prof. dr. ir. K.I. Aardal	TU Delft
	prof. dr. M.D. Romero Morales	Copenhagen Business School
	dr. J. Kurtz	Universiteit van Amsterdam
	dr. A. Kuiper	Universiteit van Amsterdam

Faculteit Economie en Bedrijfskunde

"I knew exactly what to do. But in a much more real sense, I had no idea what to do." - Michael Scott

Contents

1 Introduction **1**

1.1 Mathematical optimization 1

1.2 Mathematical optimization *with* machine learning 2

1.3 Mathematical optimization *for* explainable machine learning 3

1.4 Contributions 4

1.5 Outline 5

2 Optimization with Constraint Learning: A Framework and Survey **9**

2.1 Introduction 9

2.1.1 The synergy between optimization and machine learning 9

2.1.2 Motivation for this study 9

2.1.3 Outline 11

2.2 Running examples 11

2.2.1 Palatability considerations in WFP food baskets 11

2.2.2 Radiotherapy optimization 12

2.3 General description of framework 13

2.3.1 Step 1: setup of the conceptual optimization model 14

2.3.2 Step 2: data gathering and preprocessing 16

2.3.3 Step 3: selection and training of predictive model(s) 17

2.3.4 Step 4: resolution of the optimization model 18

2.3.5 Step 5: verification and improvement of the optimization model 19

2.4 Review 20

2.4.1 Setup of the conceptual model 20

2.4.2 Data gathering and preprocessing 22

2.4.3 Selection and training of ML models 23

2.4.4 Resolution of the optimization model 27

2.4.5 Verification and improvement of the optimization model 31
2.5 Challenges and opportunities in constraint learning 34
2.6 Conclusions 36

3 Mixed-integer Optimization with Constraint Learning **37**
3.1 Introduction 37
3.1.1 Literature review 38
3.1.2 Contributions 39
3.2 Embedding predictive models 40
3.2.1 Conceptual model 42
3.2.2 MIO-representable predictive models 44
3.2.3 Convex hull as trust region 47
3.3 Uncertainty and Robustness 49
3.3.1 Model-wrapper approach 50
3.3.2 Enlarged convex hull 51
3.4 Case study: a palatable food basket for the World Food Programme . . 51
3.4.1 Conceptual model 52
3.4.2 Dataset and predictive models 54
3.4.3 Optimization results 55
3.4.4 Robustness results 57
3.5 Case study: chemotherapy regimen design 59
3.5.1 Conceptual model 60
3.5.2 Dataset 62
3.5.3 Predictive models 62
3.5.4 Evaluation framework 63
3.5.5 Optimization results 63
3.6 Discussion 65

Appendices **67**
A Machine learning model embedding 67
A.1 Linear models 67

A.2 Decision trees . . . 68
A.3 Multi-layer perceptrons . . . 70
A.4 Model-wrapper approach . . . 72
B Trust region . . . 73
B.1 Defining the convex hull . . . 73
B.2 Column selection . . . 74
C WFP case study . . . 78
C.1 Food baskets generation and palatability function . . . 78
C.2 Predictive models . . . 79
C.3 Effect of robustness parameters . . . 79
D Chemotherapy regimen design . . . 83
D.1 Data processing . . . 83
D.2 Predictive models . . . 84
D.3 Prescription evaluation . . . 84
D.4 Optimization runtimes . . . 85
E Comparison with JANOS and EML . . . 85
E.1 OptiCL vs JANOS . . . 87
E.2 OptiCL vs EML . . . 88

4 Embedding Machine Learning Based Toxicity Models within Radiotherapy Treatment Plan Optimization 91

4.1 Introduction . . . 91
4.2 Methodology . . . 92
4.2.1 General OCL framework . . . 93
4.2.2 Experimental design . . . 100
4.3 Results . . . 104
4.3.1 Toy example . . . 104
4.3.2 Clinical study . . . 106
4.4 Discussion . . . 107
4.5 Conclusion . . . 112

Appendices 113

A Clinical Study: size of the optimization model 113

5 Counterfactual Explanations Using Optimization With Constraint Learning 115

5.1 Introduction 115

5.2 Generation of counterfactual explanations 117

5.3 Experiments and results 119

5.4 Discussion 121

Appendices 123

A Generating counterfactuals 123

B Comparison against other methods 125

C Case studies 129

C.1 German Credit Data tables 129

C.2 German Credit Data CE generation model 131

C.3 Heart tables 132

C.4 Heart CE generation model 133

6 Finding Regions of Counterfactual Explanations via Robust Optimization 135

6.1 Introduction 135

6.2 Robust counterfactual explanations 138

6.2.1 Comparison against model-robustness 139

6.2.2 Algorithm 141

6.3 Trained models 145

6.3.1 Decision trees 145

6.3.2 Tree ensembles 150

6.3.3 Neural networks 152

6.4 Experiments 154

6.5 Discussion and future work 159

Appendices 161

A How to choose the robustness budget 161

A.1 Probability guarantee 161

A.2 Pareto front 162
B Linear models 163
C Proof of Lipschitz continuity of neural networks 164
D Performances of predictive models 165

7 Conclusion **167**
7.1 The implications of OCL 167
7.2 Limitations and future research 168

Bibliography **171**

Acknowledgments **189**

Summary
Optimization with Constraint Learning **191**

Samenvatting
Optimization with Constraint Learning **193**

Chapter 1

Introduction

1.1 Mathematical optimization

Mathematical optimization is a powerful tool that finds application across various domains, from engineering and logistics to finance and healthcare. It addresses diverse and large-scale problems, allowing for the efficient allocation of resources, time, and effort, ultimately leading to improved decision-making and problem-solving. The impact of mathematical optimization on society and business is profound and far-reaching. In the business sector, optimization techniques are pivotal in enhancing operational efficiency, reducing costs, and improving decision-making processes. Companies leverage optimization to streamline supply chains (Zhou et al. 2000), manage inventories (Ahiska and King 2010), optimize pricing strategies (Ito and Fujimaki 2017), and design effective marketing campaigns (Altshuler et al. 2014). These improvements not only boost profitability but also contribute to sustainable business practices by minimizing waste and optimizing the use of resources. Beyond the corporate world, optimization plays a critical role in addressing societal challenges. In healthcare, for instance, mathematical optimization is used to improve the allocation of medical resources, design effective treatment plans, and manage public health interventions (Grigoroudis and Phillis 2013, Bertsimas et al. 2018).

A sub-field of mathematical optimization is mixed integer optimization (MIO). It uses linear functions and both continuous and integer variables to describe a problem as an optimization model. Its structure makes it possible to solve to optimality very large problems thanks to efficient algorithms and the remarkable rate at which the computational power has grown in the last two decades (Koch et al. 2022).

The effectiveness of MIO and mathematical optimization in general, hinges on the representational accuracy of the model, meaning how well it captures reality through mathematical equations. This can be challenging when optimization problems involve unknown parameters or constraints and objective functions that lack explicit formulas. Machine learning (ML) methods can be used to learn unknown parts of an optimization model when data is available. Techniques known as "predict and optimize" can be adopted to train an ML model and use it to predict the parameters of the optimization model (Bertsimas and Kallus 2020, Elmachtoub and Grigas 2022). However,

when the functions of an optimization model are unknown, the predict-and-optimize approach does not offer a viable solution.

This thesis addresses the process of learning unknown constraints and objective functions by presenting a unified framework. Here, MIO-representable ML models are trained on features that also serve as decision variables and are embedded into the optimization model to approximate the unknown functions. Overall, this work contributes to bridging the gap between mathematical optimization and ML, demonstrating how both disciplines can mutually benefit each other in innovative ways.

When referring to the studies of this thesis, I will use a plural pronoun to acknowledge the contributions of my collaborators.

1.2 Mathematical optimization *with* machine learning

In mathematical optimization, accurately defining the model can be challenging. Consider the following case: the World Food Programme wants to optimize the food supply chain for a specific population in need. This problem can be formulated as an MIO model, where the objective is to minimize total costs, and the constraints concern the material flow, fleet capacity, and supply availability. However, as formulated by Peters et al. (2021), the model does not include constraints to ensure that the food basket is "palatable" — meaning that the recipients enjoy the food and know how to prepare good dishes. There is no explicit function to represent people's taste, but we can survey preferences, collect data, and use this data to learn the constraints. This process of learning constraints and objective functions from data is referred to as "constraint learning." The term *constraint learning*, as used in this thesis, was first applied in the fields of constraint satisfaction and constraint programming (De Raedt et al. 2018), and was later adopted by the MIO field, beginning with the works presented in the next two chapters.

Optimization with constraint learning (OCL) initially found applications in developing chemotherapy regimens via linear regression (Bertsimas et al. 2016b). In recent years, more complex predictive models, such as decision trees (Bonfietti et al. 2015, Verwer et al. 2017), ensembles of trees (Mišić 2020), and neural networks (Grimstad and Andersson 2019, Amos et al. 2017), have been studied for embedding into optimization models. These studies pioneered the use of ML to learn the constraints of an optimization model, but none conceptualized a framework that could generalize to different scenarios and different ML models. Thus, in Chapter 2 and Chapter 3 of this thesis, we illustrate a framework for OCL, focusing on ML models that can be efficiently embedded into an optimization model using MIO formulations. Focusing on MIO-representable ML models helps preserve computational speed and global optimality, which is particularly relevant for large-scale problems.

A comprehensive framework is essential to facilitate adoption by practitioners and researchers, offering significant opportunities for further research to refine and expand the applicability of OCL across various fields. The framework addresses crucial topics such as model selection and the inherent uncertainty of learning from data. As demonstrated in Chapter 4, in the context of large-scale problems, the complexity introduced by the ML model becomes decisive in determining whether a problem can be solved within a reasonable time-frame. Concerning the challenges posed by learning from data, the framework introduces two approaches aimed at increasing the robustness of the learned constraints and preserving the predictive performance of trained ML models during the optimization process. Both approaches allow an end user to directly trade-off the conservativeness of the learned constraint satisfaction.

OCL has the potential to disrupt the application of mathematical optimization in practice. It enables practitioners to develop optimization models that more accurately reflect real-world problems and can be tailored to specific scenarios. For instance, medicine is one field that would greatly benefit from personalization, as it would lead to more effective treatments and better-targeted therapies, thereby increasing the likelihood of successful outcomes (Stefanicka-Wojtas and Kurpas 2023). In Chapter 4, we demonstrate how OCL can be used to personalize radiotherapy treatments by leveraging historical data to learn patient-specific constraints, rather than adopting population-based approaches.

1.3 Mathematical optimization *for* explainable machine learning

Machine learning is experiencing remarkable growth, attracting numerous researchers and practitioners to the field. One of the core components of ML is mathematical optimization, as the essence of most ML algorithms lies in building an optimization model and learning the parameters in the objective function from the given data (Sun et al. 2019). The increasing adoption of ML models in our daily lives has necessitated the development of techniques to enhance their transparency. Explainability in ML is essential for building trust and transparency, as it allows users to understand how decisions are made by the model. This is particularly relevant in the context of OCL, where ML models are used to quantify the quality or feasibility of a solution.

One approach to increase the transparency of ML models is through counterfactual explanations (CEs), which provide information on how to minimally change the original variables in order to achieve a different prediction outcome (Wachter et al. 2018). As an example, imagine a doctor using an ML model to predict whether a patient is at high risk for diabetes, and the model predicts that the patient is indeed at high risk. A counterfactual explanation would help the doctor understand what changes in the

patient's health profile can lead to a different outcome. For example, the explanation might indicate that if the patient's body mass index was five points lower and their fasting blood sugar level was 30 mg/dL lower, then the model would predict a low risk of diabetes. This explanation shows the specific changes that could alter the model's prediction, making the decision-making process more transparent and actionable for both the doctor and the patient.

Verma et al. (2020) define a set of criteria for "good" CEs: they should be close to the original instance, should be coherent, should not suggest too many changes, and should be attainable while satisfying causal relationships in the data. In Chapter 5, we demonstrate that the OCL framework can be used to generate CEs that satisfy all the criteria identified in the literature. We focus on MIO-representable ML models and show how the field of mathematical optimization and the field of ML, apparently different, have many similarities, and how advances in one field can enhance the other.

Using a mathematical optimization approach to generating CEs allows for the flexible incorporation of constraints to improve the CE quality. However, in some cases, providing the user with only one or a few CEs might result in recourses that are not achievable for the user. In Chapter 6, we expand on the theory introduced in Chapter 5 to derive an iterative method to calculate robust CEs, *i.e.*, CEs that remain valid even when slightly perturbed. This consideration is relevant in situations where the decision-making process is influenced not only by the user but also by external factors, which can affect the recourse. Our method employs algorithmic ideas from robust optimization to provide a whole region of CEs, allowing the user to *choose* a suitable recourse to achieve the desired outcome.

1.4 Contributions

This thesis makes significant contributions to the field of mathematical optimization by presenting a survey of the OCL literature and identifying critical research gaps and opportunities for future investigation. Moreover, it introduces a comprehensive framework along with user-friendly software, enabling practitioners to apply the OCL framework to their specific problems. A case study on cancer treatment planning demonstrates how OCL can lead to personalized treatments that align more closely with the patient's medical profile. Additionally, the thesis advances the field of explainable artificial intelligence through two key applications of OCL. First, it demonstrates the use of the OCL framework to generate high-quality CEs for a wide range of ML models. Second, it illustrates how robust optimization techniques can be employed to produce CEs that are resilient to small perturbations.

This thesis is based on the following works:

Ch. 2. A. O. Fajemisin, D. Maragno, D. den Hertog

Optimization with constraint learning: A framework and survey
European Journal of Operational Research, 314(1):1-14, 2024.

Ch. 3. D. Maragno, H. Wiberg, D. Bertsimas, Ş. İ. Birbil, D. den Hertog, A. O. Fajemisin
Mixed-integer optimization with constraint learning
Operations Research, 0(0), 2023.

Ch. 4. D. Maragno, G. Buti, Ş. İ. Birbil, Z. Liao, T. Bortfeld, D. den Hertog, A. Ajdari
Embedding machine learning based toxicity models within radiotherapy treatment plan optimization
Physics in Medicine & Biology, 69(7):075003, 2024.

Ch. 5. D. Maragno, T. E. Röber, İ. Birbil
Counterfactual explanations using optimization with constraint learning
OPT 2022: Optimization for Machine Learning (*NeurIPS 2022 Workshop*), 2022.

Ch. 6. D. Maragno, J. Kurtz, T. E. Röber, R. Goedhart, Ş. İ. Birbil, D. den Hertog
Finding regions of counterfactual explanations via robust optimization
INFORMS Journal on Computing, 0(0), 2024.

Each of these chapters functions as a standalone paper and can be read independently. The author contributions per chapter are as shown in Table 1.1.

	Conceptualization	Methodology	Software	Writing	Visualization	Supervision
Maragno	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	
den Hertog	2,3,4,6	2,3,4,6		2,3,4,6		2,3,4,6
Birbil	3,4,5,6	3,4,5,6		3,4,5,6		3,4,5,6
Bertsimas	3	3				3
Fajemisin	2,3	2,3		2,3	2	
Röber	5,6	5,6	5,6	5,6	5,6	
Wiberg	3	3	3	3	3	
Goedhart	6	6		6		
Kurtz	6	6		6		
Ajdari	4	4		4		4
Bortfeld						4
Buti		4	4	4	4	
Liao						4

Table 1.1. Author contributions per chapter.

1.5 Outline

In the first part of the thesis, we propose a framework for OCL and review recent literature to identify trends and future research areas. We then focus on mixed-integer optimization, developing a framework to learn constraints using MIO-representable predictive models. We showcase its effectiveness through experimentation on a real-world application in radiotherapy treatment planning. Finally, we illustrate how OCL can be used to generate optimal and robust counterfactual explanations, enhancing the interpretability of trained machine learning models.

Chapter 2: Optimization with Constraint Learning: A Framework and Survey

In Chapter 2, we present a comprehensive survey on optimization with constraint learning. Many real-world optimization problems involve constraints or objectives without explicit formulations. However, if data on feasible and infeasible states are available, these constraints can be learned from the data. This chapter highlights the advantages of such an approach and emphasizes the necessity for a structured methodology. The framework includes the following steps: (i) setup of the conceptual optimization model, (ii) data gathering and preprocessing, (iii) selection and training of predictive models, (iv) resolution of the optimization model, and (v) verification and improvement of the optimization model. To illustrate the main concepts of the framework, we use two applications throughout the chapter: one in radiotherapy optimization and one in the palatability of food baskets intended for populations in need. We review recent OCL literature within this framework, identifying current trends and suggesting potential areas for future research.

Chapter 3: Mixed-Integer Optimization with Constraint Learning

In Chapter 3, we establish a broad methodological foundation for mixed-integer optimization with learned constraints. We propose an end-to-end pipeline for data-driven decision-making in which constraints and objectives are directly learned from data using machine learning, and the trained models are embedded in an optimization formulation. We exploit the mixed-integer optimization-representability of many machine learning methods, including linear models, decision trees, ensembles, and multi-layer perceptrons, which allows us to capture various underlying relationships between decisions, contextual variables, and outcomes. We also introduce two approaches for handling the inherent uncertainty of learning from data. First, we characterize a decision trust region using the convex hull of the observations, to ensure credible recommendations and avoid extrapolation. We efficiently incorporate this representation using column generation and propose a more flexible formulation to deal with low-density regions and high-dimensional datasets. Then, we propose an ensemble learning approach that enforces constraint satisfaction over multiple bootstrapped estimators or multiple algorithms. In combination with domain-driven components, the embedded models and trust region define a mixed-integer optimization problem for prescription generation. We demonstrate the method in both World Food Programme planning and chemotherapy optimization. The case studies illustrate the framework's ability to generate high-quality prescriptions as well as the value added by the trust region, the use of ensembles to control model robustness, the consideration of multiple machine learning methods, and the inclusion of multiple learned constraints.

Chapter 4: Embedding Machine Learning Based Toxicity Models within Radiotherapy Treatment Plan Optimization

In Chapter 4, we use the optimization with constraint learning framework to address radiation-induced toxicity (RIT) challenges in radiotherapy (RT) by developing personalized treatment plans. The framework leverages patient-specific data and dosimetric information to create an optimization model that limits adverse side effects using constraints learned from historical data. It consists of three steps: optimizing the baseline treatment plan using population-wide dosimetric constraints, training an ML model to estimate the patient's RIT for the baseline plan; and adapting the treatment plan to minimize RIT using ML-learned patient-specific constraints. Various predictive models, including classification trees, ensembles of trees, and neural networks, are applied to predict the probability of grade 2+ radiation pneumonitis (RP2+) for non-small cell lung (NSCLC) cancer patients three months post-RT. The methodology is assessed with four high RP2+ risk NSCLC patients, with the goal of optimizing the dose distribution to constrain the RP2+ outcome below a pre-specified threshold. Conventional and OCL-enhanced plans are compared based on dosimetric parameters and predicted RP2+ risk. Sensitivity analysis on risk thresholds and data uncertainty is performed using a toy NSCLC case. Experiments show the methodology's capacity to directly incorporate all predictive models into RT treatment planning. In the four patients studied, mean lung dose and V20 were reduced by an average of 1.78 Gy and 3.66%, resulting in an average RP2+ risk reduction from 95% to 42%. Notably, this reduction maintains tumor coverage, although in two cases, sparing the lung slightly increased spinal cord max-dose (0.23 Gy and 0.79 Gy). By integrating patient-specific information into learned constraints, the study significantly reduces adverse side effects like RP2+ without compromising target coverage. This unified framework bridges the gap between predicting toxicities and optimizing treatment plans in personalized RT decision-making.

Chapter 5: Counterfactual Explanations Using Optimization With Constraint Learning

In Chapter 5, we demonstrate how optimization with constraint learning can be applied in the field of explainable AI, specifically to generate CEs. To increase the practical adoption of CEs, the literature has proposed several criteria that they should meet. We introduce counterfactual explanations using optimization with constraint learning (CE-OCL), a generic and flexible approach that addresses all these criteria and allows room for further extensions. Specifically, we discuss how the optimization with constraint learning framework can be leveraged to generate counterfactual explanations and how its components align with the criteria. We also propose two novel modeling approaches to address data manifold closeness and diversity, which are essential

for practical counterfactual explanations. We test CE-OCL on several datasets and present our results in the German Credit case study. Compared to current state-of-the-art methods, CE-OCL offers greater flexibility and superior performance across various evaluation metrics.

Chapter 6: Finding Regions of Counterfactual Explanations via Robust Optimization

In Chapter 6, we build on the methodology from the previous chapter to develop an iterative method for generating *robust* counterfactual explanations, *i.e.*, CEs that remain valid even after the features are slightly perturbed. Our method provides a whole region of CEs, enabling users to select a suitable recourse to achieve the desired outcome. In this chapter, we also compare recourse robustness, the focus of this work, with model robustness, a related topic that has garnered significant interest in recent literature, highlighting the key differences between these two concepts. We use algorithmic ideas from robust optimization and prove convergence results for the most common machine learning methods including decision trees, tree ensembles, and neural networks. Our experiments show that our method can efficiently generate globally optimal robust CEs for a variety of data sets and classification models.

Chapter 7: Discussion

In Chapter 7, I discuss the theoretical and practical contributions of this thesis. Moreover, I identify and discuss the limitations of our studies and provide suggestions for future research.

Chapter 2

Optimization with Constraint Learning: A Framework and Survey

2.1 Introduction

2.1.1 The synergy between optimization and machine learning

The synergy between mathematical optimization and Machine Learning has come to the forefront in recent years and has been discussed in depth in the recent literature. Sun et al. (2019) provide a recent survey of optimization methods from an ML perspective. In another survey, Gambella et al. (2021) present fundamental ML tasks such as classification, regression, clustering, adversarial learning, and others as optimization problems. In the optimization field, more and more attention is being given to using ML to aid the formulation and solution of optimization problems. Practical optimization problems can be large and difficult to solve, and more and more authors are exploiting the flexibility and usefulness of ML to aid in the solution of difficult optimization problems (*e.g.*, Fischetti and Fraccaro (2019) and Abbasi et al. (2020)). In addition to speeding up the solution of problems, ML and deep learning have also been used to speed up tree search in Hottung et al. (2020), accelerate the Branch-and-Price Algorithm in Vaclavik et al. (2018), and guide branching in Mixed Integer Programs (MIP) in Khalil et al. (2016). ML has also been used to improve meta-heuristic solution approaches (Karimi-Mamaghan et al. 2022). A recently published survey by Bengio et al. (2021) provides an overview of recent attempts to leverage machine learning to solve combinatorial optimization problems.

2.1.2 Motivation for this study

The examples in the previous section use ML to learn or obtain solutions to optimization problems in a more efficient manner. In our view, however, there is even more value in using ML for optimization: ML can be used to learn constraints. Many real-life optimization problems contain one or more constraints or objectives for which there are no explicit formulae. However when there is data available, one can use the data to learn the constraints. Two such examples are given in Section 2.2. These exam-

ples show the difficulty of formulating some constraints using explicit formulae and highlight the usefulness of being able to learn such constraints from available data. In addition to using traditional ML approaches to learn constraints for optimization problems (*e.g.*, regression trees and linear regression in Verwer et al. (2017) and neural networks in Villarrubia et al. (2018)), other techniques such as genetic programming (*e.g.*, in Pawlak and Krawiec (2018)), and mixed-integer linear programming (MILP) (*e.g.*, in Pawlak and Krawiec (2017a)) have also been used. In our literature search, we have noticed several works of this kind from several different application areas and even several scientific fields. However, it seems beneficial to have these works collated and discussed in one place. Additionally, we propose to take a more holistic view of the entire constraint learning process, starting from defining the base problem, to the resolution and validation of the final optimization model.

This paper is therefore intended to provide both a survey of literature on constraint learning for optimization, as well as offer a framework for effectively facilitating Optimization with Constraint Learning (OCL). The five steps of the proposed framework are: (i) setup of the conceptual optimization model, (ii) data gathering and processing, (iii) selection and training of the predictive model(s), (iv) resolution of the optimization model, and (v) validation/improvement of the optimization model. These steps will be discussed in more detail throughout this paper. We use the term "Constraint Learning" to make clear that we are interested in cases in which most of the optimization model is known, but some of the constraints are difficult to model explicitly. This is to differentiate from other terms such as "model learning" (Donti et al. 2017) and "model seeking" (Beldiceanu and Simonis 2016) seen in the literature. We should also note that there are several approaches in the literature that are similar to *Constraint Learning* but do not fit in this survey for several reasons. In Constraint Programming (CP) for example, the process of learning CP models from data (positive and/or negative examples) has been widely studied (Bessiere et al. 2004, O'Sullivan 2010, Beldiceanu and Simonis 2016, Kolb 2016, Galassi et al. 2018). These are not included as we are interested in cases where predictive models are embedded as constraints in mathematical optimization.

We also exclude papers on the subject of Satisfiability Modulo Theories (SMT) (Nieuwenhuis and Oliveras 2006) from our survey. Additionally, papers such as Bertsimas and Kallus (2020), Elmachtoub and Grigas (2022), Demirovic et al. (2019) and Villarrubia et al. (2018) which focus on predicting problem *parameters*, or use predictive models to approximate *known* objective functions and constraints are excluded. We also exclude papers where the structure of the problem is known, but predictive models are used to replace the problem in order to speed up computation (Nascimento et al. 2000, Nagata and Chu 2003, Venzke et al. 2020a, Venzke and Chatzivasileiadis 2020). Similarly, papers like Shang et al. (2017) and Han et al. (2021) which use predictive models to learn the uncertainty set for robust optimization problems are not considered. Finally, papers which show an interplay between ML and optimization in some way but do

not focus on constraint learning (*e.g.*, Gilan and Dilkina (2015), Bagloee et al. (2018), Say and Sanner (2018) and Say et al. (2020)) are also out of the scope of this survey.

While Lombardi and Milano (2018) and Kotary et al. (2021) provide short surveys on OCL, we present a more extensive framework for constraint learning and consider a wider range of the literature. Papers identified in our literature search will be discussed in light of the proposed framework, and gaps in knowledge and scope for future lines of research will be identified. Running examples will be used to illustrate the steps of the framework. It is envisaged that in addition to providing a review as well as the framework, our work will bring together ideas from diverse fields.

2.1.3 Outline

This framework and survey is structured as follows. In Section 2.2, we first present the running examples that will be used to illustrate our framework. We then describe this framework in Section 2.3 and review the recent literature on light of it in Section 2.4. Areas for further research are presented in Section 2.5, with general conclusions given in Section 2.6.

2.2 Running examples

In order to illustrate the concepts presented in later sections, we first present two practical optimization problems in which constraint learning plays an important role.

2.2.1 Palatability considerations in WFP food baskets

This first example deals with the issue of providing palatable rations in the food baskets delivered by the World Food Programme (WFP) to population groups in need. These food baskets are designed to meet the nutritional requirements of a population and take into account existing levels of malnutrition and disease, climatic conditions, activity levels, and so on (UNWFP 2021). In addition to considering the nutritional content, we would also like to be able to consider the palatability of the rations. This problem is particularly relevant in that the palatability of a food basket can change drastically between different groups of beneficiaries. Since the palatability constraint is difficult to model manually, and a data set with different food baskets and their palatability scores is available, a predictive model can be used to derive the palatability constraint from the data, which can then be embedded into the original food basket optimization problem.

Let $\mathcal{K}$ be the set of commodities (such as rice, flour, chickpeas, sugar, etc.) that may be included in a food basket, and $\mathcal{L}$ be the set of nutrients (such as vitamins, minerals,

proteins, etc.) which are important for maintaining good health. If the nutritional requirement for nutrient $l \in \mathcal{L}$ (in grams/person/day) is denoted by $Nutreq_l$, and the nutritional value for nutrient l (per grams of commodity $k \in \mathcal{K}$) is denoted by $Nutval_{kl}$, then a simple model for food basket optimization problem can be given as

$$\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \tag{2.1a}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} Nutval_{kl}\, x_k \geq Nutreq_l, \qquad \forall l \in \mathcal{L}, \tag{2.1b}$$

$$Palatability\_score(\boldsymbol{x}) \geq P_{min}, \tag{2.1c}$$

$$x_k \geq 0, \qquad \forall k \in \mathcal{K}, \tag{2.1d}$$

where x_k is the amount of each commodity $k \in \mathcal{K}$ (measured in grams) included in the food ration, $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{K}|}$ is the cost vector associated with the commodities (in \$/gram), and the objective function (2.1a) minimizes the ration cost. Constraints (2.1b) are used to ensure a minimum level of nutritional requirements in the rations. Constraint (2.1c) has been learned using a predictive model and is used to ensure that the palatability of the ration is at least equal to P_{min}. The diet problem in (2.1a) to (2.1d) is only a part of a much bigger model (see Peters et al. (2021)) that also includes the sourcing, facility location, and transportation problems. The constraint to be learned here is a function of relatively few variables ($|\mathcal{K}|$ in this case), compared with the thousands of variables in the full model.

The predictive model is trained on a sparse data set $\boldsymbol{D}$ in $\mathbb{R}^{n \times (|\mathcal{K}|+1)}$ composed of n samples, with each sample composed of $|\mathcal{K}|$ features and one label. Each sample is a different food basket where the $|\mathcal{K}|$ features are the amounts of each commodity k that make up the food basket, and the label corresponds to the palatability score associated with each specific food basket. This label can either be numeric or categorical, resulting in $Palatability\_score(\boldsymbol{x})$ being either a regression or classification model respectively. The data and code for this example are available at `https://github.com/hwiberg/OptiCL`.

2.2.2 Radiotherapy optimization

The second example problem we will use to illustrate the framework is radiotherapy (RT) optimization. In RT, ionizing radiation beams are used to control tumor growth by damaging cancerous tissue. Healthy, non-cancerous tissue (also known as organs-at-risk (OAR)) are however present in close proximity to the tumors, and consequently incur unavoidable damage. The tumor damage can be quantified using the Tumor Control Probability (TCP), while the unavoidable damage to the OARs can be quantified using the Normal Tissue Complications Probability (NTCP). The final optimization problem is to maximize the TCP subject to a set of constraints ensuring that

the NTCP does not exceed a specified level. If $\mathcal{Q}$ is the set of OARs (*e.g.*, stomach, liver, spinal cord) that need to be protected, then the optimization problem can be written as

$$\max_{\boldsymbol{x},\boldsymbol{d}} \text{TCP}(\boldsymbol{d}, \boldsymbol{w}) \tag{2.2a}$$

$$\text{s.t. NTCP}_q(\boldsymbol{d}, \boldsymbol{w}) \leq \delta_q, \qquad \forall q \in \mathcal{Q}, \tag{2.2b}$$

$$f(\boldsymbol{d}) \leq 0, \tag{2.2c}$$

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{d}, \tag{2.2d}$$

$$\boldsymbol{x} \geq 0, \tag{2.2e}$$

where $\boldsymbol{w}$ is a vector containing the biological characteristics of the tumor and the OARs, as well as other patient-specific information. The maximum allowable damage to OAR q is given by δ_q, while constraint (2.2c) is a placeholder for other constraints on the dose distribution $\boldsymbol{d}$ (*e.g.*, minimum dose, maximum dose, etc.) and is usually convex or can be converted to a convex form. The decision variables $\boldsymbol{x}$ and $\boldsymbol{d}$ are the set of the radiation intensities and the doses to be delivered over the course of the treatment respectively, while $\mathbf{A}$ is the dose deposition matrix. A more detailed treatment of intensity-modulated radiation therapy optimization can be found in Hoffmann et al. (2008).

In the current literature, explicit expressions are used for the TCP and NTCP functions (Gay and Niemierko 2007). These however are surrogate functions based on population-wide estimates, and as such, the response of different patients to the same treatment can be quite varied (Scott et al. 2021b). Thus, while some patients experience complete remission with minimal damage to OARs, others suffer from radiation poisoning or tumor recurrence. It has been shown in Tucker et al. (2013) that the inclusion of patient-specific characteristics (such as genetic predispositions) can improve the prediction of patients' response to RT treatment. We therefore intend to use data to learn (2.2a) and (2.2b), leading to the creation of more effective treatment plans for patients. This problem can be reformulated following the approach given in Section 2.3.1 to learn these objectives and constraints.

The above two examples have illustrated the potential benefits of OCL. In the next section, we will describe the general framework for OCL, as well as show how to apply the steps of the proposed framework to optimization problems.

2.3 General description of framework

The proposed framework is the result of a careful analysis of many approaches adopted in the literature covered by this survey. It is the formalization of a procedure that (par-

tially) recurs in most approaches to learning constraints. We believe that a complete view of all the steps that characterize the framework will help future research see the big picture of constraint learning, and the impact that each step has on the final performance.

2.3.1 Step 1: setup of the conceptual optimization model

This initial step is usually done in collaboration with domain experts. This step begins with (i) defining the decision variables and parameters involved, and (ii) defining those constraints that are easy to model manually. After this, the constraint(s) to be learned can be included. Finally, any additional auxiliary constraints required to facilitate the constraint learning process may be added. A broad formulation of all the fundamental aspects that characterize optimization with constraint learning is given as:

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x}) \tag{2.3a}$$
$$\text{s.t. } \boldsymbol{g}(\boldsymbol{x}) \leq 0, \tag{2.3b}$$
$$\boldsymbol{y} = \hat{\boldsymbol{h}}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z}), \tag{2.3c}$$
$$z_i = s_i(\boldsymbol{x}), \qquad i = 1, \dots, m \tag{2.3d}$$
$$\boldsymbol{\theta}(\boldsymbol{y}) \leq 0, \tag{2.3e}$$
$$\boldsymbol{x} \in X, \tag{2.3f}$$

where the vector $\boldsymbol{x}$ is the vector of decision variables, (2.3a) is the known objective function and (2.3b) are the constraints that are easy to model. In (2.3c), $\boldsymbol{y}$ is the difficult part of the model to design manually and is the output of a predictive model $\hat{\boldsymbol{h}}(\cdot)$ which takes as input the decision variables $\boldsymbol{x}$, and $\boldsymbol{z}$, which is the vector of variables designed as a combination of the $\boldsymbol{x}$ variables. The predictive model's performance can be often improved with the addition of other features $\boldsymbol{w}$ that are not part of the decision-making process, but convey additional or contextual information about the problem (Chen et al. 2020a, Bertsimas et al. 2023). For example in the radiotherapy optimization example in Section 2.2.2, although the decision variables are radiation intensities $\boldsymbol{x}$, additional information is conveyed by the vector $\boldsymbol{w}$ which contains patient-specific information (such as gene expressions). Although $\boldsymbol{w}$ is known in this example, it may be unknown in other problems and as such, we may have to optimize for the average or worst case. It may also be possible to treat any uncertainties in $\boldsymbol{w}$ using a robust optimization (Gorissen et al. 2015) or stochastic programming (Birge and Louveaux 2011) approach.

The vector $\boldsymbol{x}$ can also be exploited to build more complex features using a process

known as *feature engineering* (Kuhn and Johnson 2019). The idea behind this process is to use effective combinations of decision variables as new features to improve the performance of the predictive model, reduce the number of features used in the learning process, or embed prior knowledge into the predictive model. These new features $\boldsymbol{z} = (z_1, \ldots, z_m)$ can be expressed using constraint (2.3d), where s_i is the specific function used to extract the feature z_i. The palatability constraint in Section 2.2.1 is an example of such a case. It is not simply a function of the decision variables, but more accurately a function of aggregates of $\boldsymbol{x}$. These z_i variables represent the amount of food per macro category (Cereals & Grains, Pulses & Vegetables, Oils & Fats, Mixed & Blended foods, Meat & Fish & Dairy), so that (2.1c) becomes $Palatability\_score(\boldsymbol{z}) \geq P_{min}$. It should be noted that some part of the known constraint (2.3b) might also be a function of $\boldsymbol{y}$, *i.e.*, $\boldsymbol{g}(\boldsymbol{x})$ could be $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$, however we will continue with the definition where the known and unknown constraints are written separately. Similarly, some part of the objective function might also depend on the quantity to be learned (*i.e.*, $f(\boldsymbol{x}, \boldsymbol{y})$), but we will also continue with the definition where $f(\boldsymbol{x})$ is known. Finally, (2.3e) contains any required constraints on $\boldsymbol{y}$.

In order to avoid getting a solution far from what is seen in the data, we can restrict the optimization to a region that is densely populated by samples that have been used to train and test the predictive model. This region is known as a Trust Region (TR), and is briefly mentioned by Biggs et al. (2023) and expounded upon in Maragno et al. (2023). Here, the authors constrain the optimal solution to be within the convex hull of all samples (training and test samples regardless of their label). Although the computation of the convex hull is burdensome in a high-dimensional space, the authors circumvent the problem by constraining the optimal solution to be a convex combination of the samples. Thus, the computation of the convex hull facets is unnecessary and the problem distills to constraining the optimal solution with $O(n)$ constraints, where n is the dimensionality of the decision variable. We refer to Maragno et al. (2023) for full details.

Ideally, a trust region should account for outliers and prevent low-density regions to be considered trustworthy. Maragno et al. (2023) propose a two-step approach that first identifies high-density clusters and then defines the trust region as a combination of multiple convex hulls, one for each cluster. Smaller and denser trust regions lead to worse solutions in terms of objective function value, but also to greater confidence in the predictive model evaluation. A similar approach in terms of clustering high-density regions before constraint learning is seen in Sroka and Pawlak (2018), and Karmelita and Pawlak (2020).

2.3.2 Step 2: data gathering and preprocessing

The next step is to focus on collecting and preprocessing the data which will be used to learn the constraints (2.3c). The characteristics of the data inform the tasks to be performed and have a direct impact not only on the predictive model, but also on the final optimization model. For example, the amount of data available might influence the choice of the ML method used to learn the constraint, which may then have an influence on the approach used to solve the final optimization model. If the right data are available, then they can be used directly (with or without preprocessing) to learn the constraints (2.3c). If the data are not yet available, decisions have to be made on how to collect the data. Data collection or generation strategies should follow established principles such as Design of Experiments (DoE) techniques (which are applicable for physical experiments) or Design of Computer Experiments (DoCE) techniques (applicable for computational experiments). In the WFP application of Section 2.2.1 for example, collecting data on the palatability of rations during an epidemic or in a war zone might not be possible. In this case, the generation or simulation of data following DoCE principles can be done. A recent review of DoCE is given in Garud et al. (2017).

Data collection or generation can take place either in a single run or in multiple runs. Performing data collection multiple times can make the learning and optimization process more accurate. For example, if the simulation of a process from which data is obtained takes several hours, then the choice of input parameters for the simulator might need to be carefully considered. Adaptive DoCE techniques such as those in Crombecq et al. (2011) and Xu et al. (2014) may be used in such cases. In Derivative-Free Optimization (DFO) only a few or even one new data point is collected in each iteration to minimize the number of data points needed to obtain an optimal solution. We refer to Conn et al. (2009) for a detailed treatment on DFO. In addition to the availability or sourcing of the data, the size of the dataset also informs the choice of the learning approach. It is known that deep neural networks are able to exploit large datasets better than traditional algorithms such as logistic regression or support vector machines (Ng 2020). In contrast, methods such as kriging (Matheron 1963) can be used with smaller datasets. Regardless of the size of the datasets, most data (especially those not generated with the use of simulators) need to be pre-processed before they can be used in the next step (learning). Examples of various pre-processing methods are given in McKinney (2017). Whatever data collection, generation, and preprocessing approaches are used, the end result of this second step is a dataset that is ready to be used for training, testing, and validating the predictive algorithms. A review of the various data generation and preprocessing approaches used in the context of constraint learning is given in Section 2.4.2.

2.3.3 Step 3: selection and training of predictive model(s)

Once the data gathering and preprocessing activities have been completed, it is necessary to select, train, validate, and test the predictive model that will subsequently be used to represent one or more constraints in the final optimization model. Selection and training are two activities that go hand in hand. It is not possible to select the best predictive model without first training and testing it on the available dataset. Correspondingly, the poor performance of a model after training might require a different predictive model to be selected. The selection of the predictive model is based on six main factors:

i Classification or regression: The choice of the predictive model will depend on whether the constraint to be learned results in a discrete set of values (classification), or a continuous spectrum of values (regression). The use of a classification model results in a feasible set being learned, while the use of a regression model results in a function being learned.

ii Computational complexity: The choice of the predictive model directly affects the computational complexity of the final optimization model. In contrast to a linear regression model, deep learning models require several mathematical or logical statements to represent them (see Section 2.4.4) and undoubtedly increase the complexity of the final optimization problem. A decision might also be made to select a predictive model so that the complexity of the final optimization problem does not increase. In the WFP example, the underlying problem is linear, so a linear predictive model may be chosen to learn (2.1c).

iii Model performance: In addition to computational complexity, the predictive model has to perform adequately with respect to some performance measure (*e.g.*, accuracy or mean squared error). A simple but inaccurate model can have worse consequences than a complex but accurate model. The performance should be evaluated on both a test set and in the final optimization model. This means that the performance evaluation has to be performed both in Step 3 and Step 5.

iv Confidence intervals: If the predictive models used provide confidence intervals, it might be possible to use this information in approaches such as stochastic optimization or robust optimization.

v Data quality: Despite the attention focused on the dataset in Step 2, it may be necessary to choose a predictive algorithm that is robust to noise or missing data. This is especially the case if the data to be used has been provided by a third party.

vi Interpretability: The selection of a predictive model might also depend on whether or not the practitioner requires the model to be interpretable. In the radiotherapy

optimization problem of Section 2.2.2, clinicians may prefer highly explainable predictive models to be used to learn the TCP$(\boldsymbol{d}, \boldsymbol{w})$ and NTCP$_q(\boldsymbol{d}, \boldsymbol{w})$ functions due to the seriousness of the application. There are however pros and cons to consider. Decision trees, for example, are known to be highly explainable but may require the linearization of several disjunctions to encode them in the optimization problem. We refer to Section 2.4.4 for an example.

Table 2.1 shows the pros and cons of the predictive models generally used in constraint learning. It can be seen that while linear models are very good in terms of explainability and computational complexity, their performance is lacking. Conversely, while neural networks perform quite well, they require large amounts of labeled data to train (Ng 2020) and are not considered explainable (Molnar 2020). A review of the various predictive models used for constraint learning in optimization is given in Section 2.4.3.

Table 2.1. Pros and cons of predictive models commonly used in constraint learning

	Explainability	Complexity	Data Required	Performance
Linear models*	++	++	++	−−
Decision Trees	+	+	+	+
Tree ensembles	−	−	−	+
Neural Networks	−−	−−	−−	++

++: very good; +: good; −: bad; −−: very bad

* Linear regression, linear support vector machine

The implementation process is a cycle of training the predictive model, evaluating the model's performance, and improving the model in terms of the data and the model parameters. If a feed-forward neural network is used, for example, parameters such as the number of layers, the number of nodes per layer, the activation functions, whether or not to use dropout regularization, etc., need to be determined. Out-of-sample testing should be done to ensure the adequate performance of the model, with changes made to improve performance if necessary. In the context of constraint learning, under the same computational complexity, the best predictive model is the one with higher performance in the final optimization model. This is related to the concept of predictive versus prescriptive models and is explained further in Bertsimas and Kallus (2020). The goal should be to generate prediction models that aim to minimize decision error, and not just prediction error (Elmachtoub and Grigas 2022).

2.3.4 Step 4: resolution of the optimization model

Once the predictive model has been used to learn the difficult-to-model constraint, it will need to be embedded into the rest of the optimization problem. Recall that the predictive model $\hat{\boldsymbol{h}}(\cdot)$ in (2.3c) may not be easy to add as a constraint and may need

to be encoded in such a way that it can be handled by commercial or conventional solvers. This however depends on the type of model used, as a linear or quadratic function can be embedded more easily than a neural network for example. The integration of the learned constraints is often not trivial and may result in a significant reformulation of the model. If possible, the predictive model may be incorporated either by providing gradients directly to the solver, or by reformulating it such that it becomes linear, convex, conic, etc., and can more readily be incorporated into the optimization problem. If desired, the optimization models may be modified to take into account any uncertainty measures provided by the predictive models. A successful embedding can also include providing additional information (such as bounds) to the solver of choice in order to facilitate a more efficient solution process. A successful embedding should make it possible to take full advantage of the optimization solver's capabilities. A deeper look at the various methods used in the literature to encode predictive models is given in Section 2.4.4.

2.3.5 Step 5: verification and improvement of the optimization model

In this final step, the performance of the constraint learning process is evaluated with respect to three things: the final optimization model, the predictive model, and the data. For the optimization model, the focus is on both the goodness of the solution (*i.e.*, the objective function value), as well as the time taken to find this solution. For the performance of the predictive model, the focus is on ensuring that the learned constraints accurately represent the difficult-to-model constraints. The data used to learn constraints must also be examined. If, for example, the data points used are far from the boundary, this might have a negative effect on the constraint learned and the optimal solution (Thams et al. 2017). If it is possible to generate data, then using different data to improve the predictive model can in turn help to improve the optimal solution. One way of doing this is via a process called active learning (Settles 2009). Here, the dataset is continually modified by adding new solutions (output of the optimization model) to it, which is then used to re-train the predictive model. A review of the approaches used to verify and improve models with learned constraints is provided in Section 2.4.5.

Table 2.2 shows the emphasis placed on the different steps in the literature. While all steps are generally present in the literature, some steps are often emphasized more than others. It can be seen that the most emphasis is placed on Step 4, *i.e.*, learning and reformulating constraints so that the final optimization problem can be easily solved. While the above framework steps have been given in sequential order, it should be noted that some of these steps may be repeated. For example, if it is seen in Step 5 that the learned constraint is a poor representation of reality, we may need to go back to Step 1. Similarly, the incorporation of new data will mean that the predictive model

needs to be retrained and re-embedded, and so on. Internal loops are also possible — it may be necessary to iterate between steps 2 and 3 until an acceptable predictive model is developed, or between steps 4 and 5 until satisfactory results are achieved.

2.4 Review

2.4.1 Setup of the conceptual model

The papers by Lombardi et al. (2017) and Maragno et al. (2023) provide very detailed approaches to formalizing the process of OCL. There is also a clear structure provided in De Angelis et al. (2003) for learning unknown constraints. The methodology in Lombardi et al. (2017), called Empirical Model Learning (EML), is described as a way of obtaining components of a prescriptive model from machine learning. Similar to the steps given in Section 2.3.1, they define a conceptual model including known constraints, constraints to be learned, as well as the definition of constraints used for feature extraction. Maragno et al. (2023) also provide a general framework for OCL that can be applied every time data is available. A key focus of their approach is to learn constraints and embed them in such a way that the computational complexity of the final model is not increased. Padmanabhan and Tuzhilin (2003) and Deng et al. (2018) also conceptually discuss the use of data mining to define objectives and constraints.

While other papers which use a predictive element to learn constraints do not have as detailed a setup, they incorporate some of the elements of Step 1 in their approach. Some authors (*e.g.*, Verwer et al. (2017)) explicitly define constraints to perform feature extraction, while most of the papers reviewed simply identify which constraints are difficult to model manually and replace them with predictive models. This will be discussed later in Sections 2.4.3 and 2.4.4.

The approach taken to formalize the constraint learning process in Pawlak and Krawiec (2017a) is different in that their framework is a MILP. Similarly, other authors also leverage MILP (Schede et al. 2019) or Integer Programming (IP) (Cozad et al. 2014) frameworks for constraint learning. Sroka and Pawlak (2018) use local search, and Kumar et al. (2021) use an approach based on stochastic local search. A number of authors also use a genetic programming framework to obtain constraints from data (Pawlak and Krawiec 2017b, 2018). Pawlak (2019) use evolutionary strategies, and Pawlak and O'Neill (2021) use grammatical evolution (O'Neil and Ryan 2003). In these different setups, the authors give the relative simplicity of their approaches as their key justification — no predictive models have to be learned, no constraints for feature extraction have to be included, and no complex embedding or reformulating procedures need to be followed. The disadvantage of these approaches however is

Table 2.2. Emphasis placed on the different steps of the framework

	Step 1	Step 2	Step 3	Step 4	Step 5
Bergman et al. (2022)	●	●	●	●	●
Biggs et al. (2023)	●	●	●	●	●
Chatzivasileiadis (2020)	●	●	●	●	●
Chen et al. (2020b)	◐	◐	●	●	◐
Chi et al. (2007)	○	●	●	●	●
Cozad et al. (2014)	◐	●	●	●	●
Cremer et al. (2018)	◐	◐	●	●	●
De Angelis et al. (2003)	●	●	●	◐	●
Fahmi and Cremaschi (2012)	●	●	●	●	◐
Garg et al. (2018)	●	◐	●	●	●
Grimstad and Andersson (2019)	●	●	●	●	●
Gutierrez-Martinez et al. (2010)	●	◐	●	●	●
Halilbašić et al. (2018)	◐	●	●	●	●
Jalali et al. (2019)	●	●	●	●	●
Karmelita and Pawlak (2020)	●	●	n/a	●	●
Kudła and Pawlak (2018)	●	●	●	●	●
Kumar et al. (2019b)	●	●	n/a	●	●
Kumar et al. (2019a)	●	●	n/a	●	●
Kumar et al. (2021)	●	●	n/a	●	●
Lombardi et al. (2017)	●	●	●	●	●
Maragno et al. (2023)	●	●	●	●	●
Mišić (2020)	●	◐	●	●	●
Venzke et al. (2020b)	●	●	●	●	●
Paulus et al. (2021)	●	●	●	●	●
Pawlak and Krawiec (2017a)	●	◐	n/a	●	●
Pawlak and Krawiec (2017b)	●	●	n/a	●	●
Pawlak and Krawiec (2018)	●	●	n/a	●	●
Pawlak (2019)	●	●	n/a	●	●
Pawlak and Litwiniuk (2021)	●	●	●	●	●
Pawlak and O'Neill (2021)	●	●	n/a	●	●
Prat and Chatzivasileiadis (2020)	◐	●	●	●	◐
Say et al. (2017)	◐	●	●	●	◐
Schede et al. (2019)	●	○	●	●	●
Schweidtmann and Mitsos (2019)	◐	◐	●	●	◐
Sroka and Pawlak (2018)	●	●	n/a	●	●
Thams et al. (2017)	◐	●	●	●	●
Verwer et al. (2017)	●	◐	◐	●	●
Xavier et al. (2021)	◐	●	●	●	●
Yang and Bequette (2021)	●	●	●	●	●

●: most emphasis; ◐: some emphasis; ○: less emphasis; n/a: not applicable

that they do not make use of the learning power of predictive models which have been shown to be able to learn very complex decision boundaries well.

2.4.2 Data gathering and preprocessing

Of the papers considered in this survey, the work of Thams et al. (2017) has a detailed approach with regards to the data generation and preprocessing phase detailed in Section 2.3.2. Their procedure, also used by Halilbašić et al. (2018) and Venzke et al. (2020b), is designed to either make use of available data from sensors or generate them using a simulator. Possible operating points of the system under consideration are fed into a simulator, and the output of the simulator is analyzed and classified as either *stable* or *unstable*. These operating points along with their classifications are then stored in a database. The data is only generated once, and they attempt to keep the generated database as small as possible for computational reasons. Although the generated data is small, they ensure that enough data is collected so that their predictive models can be trained. They do this by focusing on generating points close to the boundary of their two-class problem while neglecting operating points far away from the boundary. In this way, they balance exploration and exploitation when generating data. In addition to this, class imbalance in the dataset is addressed using several strategies, and feature selection is done so that only easily accessible variables are considered.

With regards to following DoCE principles, the authors of Lombardi et al. (2017) use factorial designs to generate multiple training sets from a simulator. Data generation here, although time-consuming, only needs to be done once. The data is also normalized so that the input features fall within a certain range. The approach developed in Cozad et al. (2014) also takes DoCE principles into account by using either Latin Hypercube Sampling (LHS) or two-level factorial design (Mukerjee and Wu 2006). They also use adaptive sampling to iteratively refine the models generated and provide discussions on whether to use exploration-based or exploitation-based sampling. De Angelis et al. (2003) use small data collected using an on-field survey (an expensive process) to find a probability distribution that is later used in a simulator that generates enough data to train the predictive model. In Fahmi and Cremaschi (2012), the data is obtained from multiple simulators. Some transformation is done to the data to ensure that the distribution of the dependent variable is as close as possible to a uniform distribution. The authors here note that a large amount of data was needed to train their neural networks for higher accuracy. Data may also be collated from multiple available sources as in the case of Bertsimas et al. (2016b).

In terms of the sizes of the datasets surveyed, it can be seen that not many of the problems considered have large datasets. Data containing 500 or fewer examples are common. In Pawlak and Krawiec (2017a), the total amount of data generated using uniform sampling varied from 10 to 500 examples. Similarly, Schweidtmann and Mit-

sos (2019) use data with 500 or fewer examples. Data of a similar size is also seen in Kudła and Pawlak (2018), Yang and Bequette (2021) and several others. For one of the test problems in Schweidtmann and Mitsos (2019), only 46 examples are used. On the larger side, Chen et al. (2020b) use data containing $10,000$ instances, Venzke et al. (2020b) use $36,144$ data points generated using LHS, while the data used in Say et al. (2017) contains $100,000$ data points. For certain constraint learning approaches, the availability of a large amount of data is a must. The local search approach used in Sroka and Pawlak (2018) is such an example, where the test set contains $500,000$ examples generated via uniform sampling.

Some approaches for constraint learning necessitate the significant preprocessing of the data. In order to learn ellipsoidal constraints in Pawlak and Litwiniuk (2021), the uniformly generated data first needs to be clustered and standardized before Principal Component Analysis (PCA) is applied to the data. This clustering and PCA procedure shapes the data so that it can be represented using ellipsoidal constraints. In other cases, although preprocessing is done, it is not crucial to the constraint learning approach (*e.g.*, in Biggs et al. (2023) and Pawlak (2019)).

In some cases, not much emphasis is placed on the data generation and preprocessing phase. This is sometimes because the data is fully available, is neither big nor dirty, and has no need for preprocessing. For example, the situation of having high-dimensional data, where the number of examples is much less than the number of features, is only explicitly mentioned in Mišić (2020). In a few cases, however, the data step is an afterthought. Even when data is generated, this is cursorily mentioned without giving details or following exhaustive DoCE principles. We believe that our framework will help to provoke more thought on this phase of the constraint learning process.

2.4.3 Selection and training of ML models

We first note that papers that use regression functions as the main constraint learning methods have been excluded from Table 2.3, as they are too numerous to be included in this paper. Regression with first- and second-order polynomials is used frequently in engineering, especially in combination with DoE and DoCE. We refer to Kleijnen (2015) for a detailed treatment. Kriging models are also used often in engineering, especially when the objective function values can be obtained by (expensive) computer simulations. The big advantage of kriging is that it has very high predictive power even for a relatively small number of data points. The disadvantages of kriging are (i) it costs much computing time to obtain the kriging model (ii) the data points have to be space-filling (iii) embedding kriging functions in the optimization model makes it non-convex and hard to solve. See Forrester et al. (2008) for a detailed treatment on kriging.

It can be seen from Table 2.3 that Artificial Neural Networks (ANN) and Decision

Table 2.3. Methods used for constraint learning

	Neural Networks	Decision Trees	Random Forest	Other Ensemble	Support Vector Machines	Clustering	(M)ILP	Other
Bergman et al. (2022)	x							x
Biggs et al. (2023)		x	x					
Chatzivasileiadis (2020)	x	x						
Chen et al. (2020b)	x							
Chi et al. (2007)					x			
Cozad et al. (2014)							x	x
Cremer et al. (2018)		x		x				
De Angelis et al. (2003)	x							
Fahmi and Cremaschi (2012)	x							x
Garg et al. (2018)					x			
Grimstad and Andersson (2019)	x							
Gutierrez-Martinez et al. (2010)	x							
Halilbašić et al. (2018)		x						
Jalali et al. (2019)					x			
Karmelita and Pawlak (2020)						x		x
Kudła and Pawlak (2018)		x						
Kumar et al. (2019b)								x
Kumar et al. (2019a)							x	x
Kumar et al. (2021)								x
Lombardi et al. (2017)	x	x						
Maragno et al. (2023)	x	x	x	x	x	x		x
Mišić (2020)		x	x					
Venzke et al. (2020b)	x							
Paulus et al. (2021)	x							
Pawlak and Krawiec (2017a)							x	
Pawlak and Krawiec (2017b)								x
Pawlak and Krawiec (2018)								x
Pawlak (2019)								x
Pawlak and Litwiniuk (2021)						x		x
Pawlak and O'Neill (2021)							x	x
Prat and Chatzivasileiadis (2020)		x						
Say et al. (2017)	x							
Schede et al. (2019)		x					x	
Schweidtmann and Mitsos (2019)	x							
Sroka and Pawlak (2018)						x		x
Thams et al. (2017)		x						
Verwer et al. (2017)		x						x
Xavier et al. (2021)					x			
Yang and Bequette (2021)	x							

Trees (DT) are the most popularly used predictive models in constraint learning (with the exception of regression models). This could be because these methods have been shown to have high representative power, can be used for both regression and classification, and generally perform well. Reviews on ANNs and DTs are given in Cao et al. (2018) and Carrizosa et al. (2021) respectively. Training, testing, and evaluation are generally done as prescribed in the ML literature, and the methods are generally evaluated using metrics such as the Mean Square Error (MSE), accuracy, and so on. For example, to learn objective functions in Biggs et al. (2023), the data is split in a 70:30 ratio between training and test sets, and prediction accuracy used as the evaluation metric. See Chapter 2 of James et al. (2013) for more information on training and testing ML models.

In terms of the size of the ANNs, with the exception of Yang and Bequette (2021) who use a network with six hidden layers, most networks used in constraint learning are generally small. One reason for this could be to reduce the burden of embedding large networks as constraints — it will be seen from (2.5a) to (2.5e) in Section 2.4.4 that the number of binary variables required is proportional to the size of the network. Also, as the input of a current layer is the output of a previous one, using multiple layers results in nested activation functions in the final model. This could lead to issues regarding convexity and/or differentiability, therefore smaller networks are used. Another reason for small neural network sizes is that using larger networks provides no benefit in some cases. Lombardi et al. (2017) for example use a network with one hidden layer (with two neurons) as increasing the network size did not improve the solution quality of the optimization problem. To view the effect of network size on computation times, Schweidtmann and Mitsos (2019) vary both the number of neurons and the number of layers. Results showed that the solution time increases approximately exponentially with the number of layers used to learn the constraints. Deep neural networks required more processing time than shallow neural networks for the same total number of neurons.

An alternative to using small networks could be to use sparsification. Venzke et al. (2020b) use a network with three hidden layers and fifty neurons in each layer to learn constraints, however they reduce the size of the network by enforcing 70% of the weights to be zero. This sparsification procedure ensures that the network was not too large and also avoided over-fitting during training. Say et al. (2017) use a similar technique to remove connections to and from neurons with very small weights in order to strengthen the MILP formulation. Another alternative could be to ensure that the constraint learning procedure results in a convex optimization problem. Chen et al. (2020b) and Yang and Bequette (2021) use Input Convex Neural Networks (ICNN) (Amos et al. 2017) as their predictive models. An ICNN requires that all weights in the network are non-negative and that all activation functions are convex and non-decreasing. Although these networks require more training iterations and have lower representation power than conventional networks, their use guarantees that the final

problem is a convex optimization problem. This allows the authors to balance the trade-off between model accuracy and computational tractability.

The activation functions used in the hidden layers in the constraint learning literature are usually either the Rectified Linear Unit (ReLU) (*e.g.*, in Say et al. (2017)), the sigmoid function (*e.g.*, in Chen et al. (2020b)), (some form of) the hyperbolic tangent function (*e.g.*, in Fahmi and Cremaschi (2012)), or the softplus activation function (*e.g.*, in Chen et al. (2020b)). The output layer activation function is usually an identity linear function. An overview of the most common types of activation functions is given in Sharma (2017).

In addition to ANNs, DTs are also popularly used in constraint learning. DTs are able to capture non-convex and discontinuous feasibility spaces (Chatzivasileiadis 2020), and provide these rules in an easily understandable format (Prat and Chatzivasileiadis 2020). For this reason, Thams et al. (2017) and Halilbašić et al. (2018) specifically choose decision trees as their method of choice to learn constraints in an optimal power flow problem. Their rationale is that as electricity is a crucial utility, using easily interpretable rules in their analysis can lower the barrier to power plant operators adopting their methods. This interpretability requirement could also favor the use of decision trees in the radiotherapy example of Section 2.2.2.

As random forests are a natural extension of using decision trees, they have also been used in constraint learning. Biggs et al. (2023) learn objective functions using random forest while Mišić (2020) optimize tree ensembles. AdaBoost (Freund and Schapire 1997) (with DTs as base learners) is also used in Cremer et al. (2018) to learn a probabilistic description of a constraint.

Support Vector Machines (SVM) can be expressed very clearly as optimization problems, and as a result, there is a significant amount of optimization literature on SVMs. Most of the literature however deals with subjects such as optimal feature selection (*e.g.*, Jiménez-Cordero et al. (2021)), or mathematical formulation and solution of the SVM problem (*e.g.*, Baldomero-Naranjo et al. (2020)), or improving predictions (*e.g.*, Yao et al. (2017)). There are however a few cases where SVM is used in the constraint learning process. Jalali et al. (2019) use SVM to learn power inverter control rules. Chi et al. (2007) use SVM for regression with a second order polynomial kernel function. A feature selection process is further used to remove insignificant terms and improve model accuracy. Xavier et al. (2021) also use SVM with linear kernels to learn hyperplanes within which the solution of the optimization problem is very likely to exist.

Clustering is also used in constraint learning, although it is often used in addition to other methods. Pawlak and Litwiniuk (2021) combine clustering with PCA and ellipsoidal constraints to get an MIQCP, while Sroka and Pawlak (2018) combine clustering with local search to search the clusters for LP constraints that represent each cluster.

Combinations of predictive models are often used. Verwer et al. (2017) combine re-

gression trees and regression models for learning revenue constraints for an auction optimization problem. In addition to using ANNs to learn most of the constraints required, Fahmi and Cremaschi (2012) also use non-linear regression models for specific constraints. Paulus et al. (2021) use both integer programming and neural networks to learn both the cost terms and the constraints for integer programs. Although Schede et al. (2019) use an LP formulation to learn polytopes that cover feasible examples, they also include a decision tree heuristic to model the importance of examples to be considered. Additionally, Maragno et al. (2023) are able to use different predictive models to learn different constraints of the same optimization problem in order to achieve better performance.

There are also approaches that do not use predictive models to learn constraints. Pawlak and Krawiec (2017b, 2018) and Pawlak (2019) use genetic programming as their approach of choice, Pawlak and Krawiec (2017a) use a mixed-integer formulation and Cozad et al. (2014) combine simple basis functions using a MILP formulation to learn constraints.

Although RF is an ensemble method, the column "Other Ensemble" in Table 2.3 is separated from the "Random Forest" column to highlight the fact that there is an opportunity to use other ensemble methods where different individual base learners can be used and their predictors combined. There is also the opportunity to try a wider variety of predictive models such as symbolic regression or Bayesian methods, especially if they are more easily represented using mathematical functions. Studies could be done to see if certain families of models perform better for certain types of problems. Finally, methods that provide some measure of uncertainty (such as confidence intervals or conditional probabilities) might also be chosen in order to be able to incorporate this uncertainty into the final mathematical model.

2.4.4 Resolution of the optimization model

With regression functions, the process of embedding them as constraints is relatively straightforward as the functions are simply a weighted sum of the decision variables (Bertsimas et al. 2016b, Verwer et al. 2017). With other predictive models, however, the embedding process is more involved. With ANNs for example, their embedding is also dependent on the activation functions used. The output $\hat{y}$ of a fully connected neural network with h hidden layers can be written as:

$$\mathbf{H}_1 = \sigma_1 \left(\mathbf{W}_1 \mathbf{X} \right), \tag{2.4a}$$

$$\mathbf{H}_i = \sigma_i \left(\mathbf{W}_i \mathbf{H}_{i-1} \right), \tag{2.4b}$$

$$\hat{y} = \sigma_{h+1} \left(\mathbf{W}_{\boldsymbol{h+1}} \mathbf{H}_h \right), \tag{2.4c}$$

where $\mathbf{X}$ is the training data, $\mathbf{W}_i$ are the weights associates with layer i, and σ_i is the activation function used in in layer i. In order to incorporate a trained neural network with ReLU activation functions (*i.e.*, $\sigma(x) = max(0, x)$) as a constraint, the $max()$ function can be replaced by introducing binary variables b_j for each neuron in the network:

$$y_j = max\left(0, \hat{y}_j\right) \implies \begin{cases} y_j \leq \hat{y}_j - \hat{y}_j^{min}\left(1 - b_j\right), & \text{(2.5a)} \\ y_j \geq \hat{y}_j, & \text{(2.5b)} \\ y_j \leq \hat{y}_j^{max} b_j, & \text{(2.5c)} \\ y_j \geq 0, & \text{(2.5d)} \\ b_j \in \{0,1\}^{N_j}. & \text{(2.5e)} \end{cases}$$

If the input to a neuron $\hat{y}_j > 0$, then $b_j = 1$ and (2.5a) and (2.5b) force the neuron output y_j to the input $\hat{y}_j$. On the other hand, if the input to the neuron $\hat{y}_j \leq 0$ then $b_j = 0$, and (2.5c) and (2.5d) force the output y_j to 0. The resulting MILP can then be solved with off-the-shelf solvers such as Gurobi (Gurobi Optimization, LLC 2022) or CPLEX (IBM 2022). This is done by authors such as Venzke et al. (2020b) and Bergman et al. (2022).

The disadvantages of this approach are twofold. Firstly, large networks require a correspondingly large number of binary variables. Secondly, the bounds $\hat{y}_j^{min}$ and $\hat{y}_j^{max}$ have to be chosen to be as tight as possible for the MILP solver. Anderson et al. (2020) present a MIP formulation that produces strong relaxations for ReLUs. They start with the big-M formulation and then add strengthening inequalities as needed. Say et al. (2017) also encode ReLU into MILP using big-M constraints, but include reformulations to strengthen the bounds. They also use sparsification to remove neurons with very small weights, as constraints with very small coefficients can be difficult to solve for commercial solvers (D'andreagiovanni and Gleixner 2016).

The embedding of trained ANNs as constraints is not always as complicated as mentioned above. Equations (2.4a) to (2.4c) can be used directly as constraints, as long as the solver can handle the activation function σ. This approach is used in Fahmi and Cremaschi (2012) with the hyperbolic tangent activation function, as well as in Gutierrez-Martinez et al. (2010) where the activation function is differentiable so that the constraint can be easily handled by conventional solvers. It is nevertheless noted in Lombardi et al. (2017) that several commercial solvers rely on convexity for providing globally optimal results and that this approach may result in locally optimal solutions depending on the σ used. In Chen et al. (2020b), the use of ICNN guarantees that the optimization problem is convex and will converge to a globally optimal solution. They do however speed up this process by using a smoothened version of the ReLU, called the Softplus activation function. A large value of the parameter t in the Softplus function allows it to be smooth enough, while still being convex. The final problem

is solved using dual decomposition (Yu and Lui 2006, Chiang et al. 2007). Yang and Bequette (2021) also use ICNNs to get convex optimization problems which are then solved using sequential quadratic programming.

Noticing the use of ANNs in various optimization applications, Schweidtmann and Mitsos (2019) provide an efficient method for deterministic global optimization problems which have ANNs embedded in them. As the only non-linearities of ANNs are their activation functions (the $\tanh$ function in their case), the tightness of their relaxations is influenced by the relaxations of the $\tanh$ functions. The key idea of their paper is to therefore hide the equations that describe the ANN and its variables from the Branch-and-Bound (B&B) solver using McCormick relaxations of the activation functions. These relaxations are once continuously differentiable, and the lower bounds are calculated by automatic propagation of McCormick relaxations through the network equations. This "Reduced-Space" formulation results in significantly fewer variables and equalities than the original formulation (by two orders of magnitude in some cases), with significantly shorter computation times in most cases. They also see that shallow ANNs show much tighter relaxations than deep ANNs. Grimstad and Andersson (2019) also provide bound tightening procedures for ReLU networks to reduce solution times.

The embedding of kriging functions in the final model is done in a direct way: substituting the kriging function leads to an explicitly formulated, but highly non-convex constraint. First and second-order derivatives can be calculated, and therefore standard nonlinear solvers can be used. However, due to the non-convexity, solvers might end up in local optima. See also den Hertog and Stehouwer (2002), and Stinstra and den Hertog (2008) for examples.

With DTs, the paths from the root to leaf nodes can be represented using disjunctions of conjunctions. These rules can either be linearized or embedded as constraints using a framework like Generalized Disjunctive Programming (Grossmann 2009, GAMS Development Corporation 2021). Although Lombardi et al. (2017) do not embed DTs into MINLPs due to the potential for poor bounds being obtained as a result of the extensive linearization of disjunctions required, other authors have. In Kudła and Pawlak (2018), branches of the tree from root to leaf nodes are encoded as linear constraints using a big-M formulation. Paths that end up in feasible decisions are denoted with the $(+)$ sign (see Figure 2.1). These paths are then converted into linear constraints using auxiliary binary variables to show which path in the tree is followed. For example, at the top of the tree, if a binary variable $b_1 = 1$, then c_1 is active and (2.6) holds. Otherwise if $b_1 = 0$, then c_2 is active and (2.7) holds. Mathematically,

$$c_1 : x_1 \leq a_1 \rightarrow x_1 \leq a_1 + M\,(1 - b_1)\,, \tag{2.6}$$

$$c_2 : x_1 > a_1 \rightarrow x_1 > a_1 - M b_1. \tag{2.7}$$

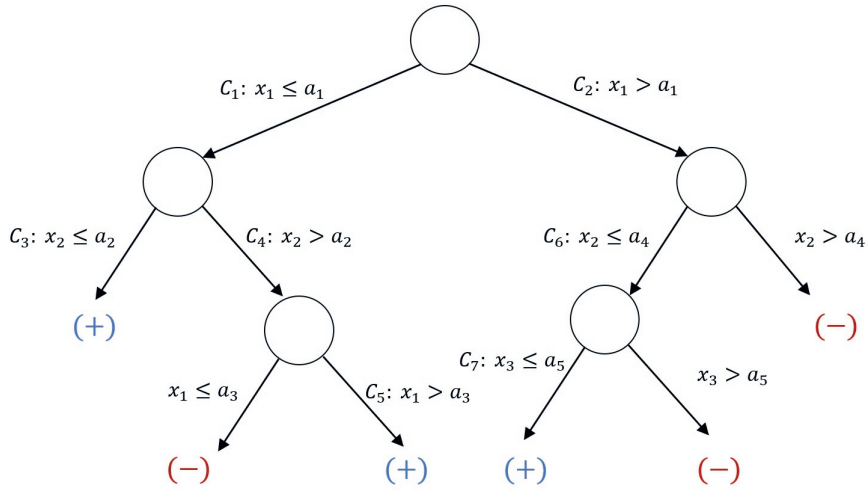


Figure 2.1. Extracting constraints from a decision tree (adapted from Kudła and Pawlak (2018)).

This process is applied to the whole tree until a set of linear constraints capturing the tree is obtained. A similar approach is used in Verwer et al. (2017) to embed regression trees as constraints. In Thams et al. (2017), each path from the root to a leaf node is represented by two constraints, with one binary variable per path to determine if a path is chosen. The resulting MILP is solved with Gurobi. Halilbašić et al. (2018) also follow a similar procedure except that the resulting problem is further reformulated as a Mixed-Integer Second-Order Cone Program (MISOCP) because of the presence of additional non-linear constraints. The resulting MISOCP and MINLP are then solved using commercial solvers.

To embed random forests as constraints, multiple DTs can be used as above, with an additional constraint that selects a consensus based on the outcome chosen by most of the individual decision trees (Bonfietti et al. 2015). Biggs et al. (2023) maximize a random forest as the objective. For each tree, they use logical constraints to determine which leaf a solution lies in. For the forest, they use a constraint to ensure that only one leaf for each tree is active. To ensure that the feasible set is bounded and that the solution is not too dissimilar from the observed data, the solution is constrained to be within the convex hull. This is similar to the approach in Mišić (2020) which also has an objective function expressed as the prediction of a tree ensemble, and also uses some form of Benders reformulation with lazy constraint generation to solve the problem. Biggs et al. (2023) also compare solving the MIP for the whole random forest, versus splitting the forest into a number of subsets, and solving each subset using the MIP, and a cross-validation heuristic procedure to achieve performance improvements. In Cremer et al. (2018), where the AdaBoost ensemble learning method is used

(with DTs as the weak learners), there is one set of constraints for each DT, and exactly one disjunction must be selected for each learner. Probability estimates for all learners are combined in a weighted sum, and the resulting MILP is solved using a commercial solver. For SVMs, linear or quadratic polynomials can be used directly in the optimization problem, providing the solvers used can handle quadratic constraints.

The advantage of using LP or MILP approaches to learn constraints is seen in this step of the framework, as there are no complex embedding procedures to be followed. In Pawlak and Krawiec (2017a), the resolution of the final optimization problem here is relatively easy as the entire constraint learning process was formulated as a MILP from the very beginning. This is also seen in papers like Sroka and Pawlak (2018) and Schede et al. (2019). In Pawlak and Litwiniuk (2021), the data is first clustered, and then PCA is used to give each cluster an ellipsoidal shape. These ellipsoids can then be represented using quadratic constraints. Their approach produces Mixed Integer Quadratically Constrained Programs (MIQCP), which can then be solved by any solver that supports quadratic programming.

In summary, it can be seen that this step of the constraint learning process is the most complex one. Incorporating predictive models sometimes results in a significant reformulation of the original model. The embedding of the predictive models should however be done in a way that takes full advantage of the optimization solver's capabilities. This can be done by providing gradients or other useful information directly to the solver or reformulating the problem to become linear, convex, conic quadratic, etc. The approach by Garg et al. (2018) results in a quadratic program with linear constraints that is solved using off-the-shelf solvers. Constraint learning approaches that use MILP or LP frameworks such as in Pawlak and Krawiec (2017a) avoid these complex embedding procedures, however, the disadvantages of this approach could include a lack of flexibility and limited learning ability. There is therefore a trade-off between simplicity and performance to be considered. Wherever possible, it is desirable to keep the final optimization problem in the same complexity class as the original problem.

2.4.5 Verification and improvement of the optimization model

The validation of the final optimization model is straightforward when dealing with benchmark problems, as the ground truth is already known. This approach to verifying models is seen in a lot of the literature. In optimal power flow, for example, the availability of benchmark problems allows Prat and Chatzivasileiadis (2020) to find solutions to problems that were previously intractable. Garg et al. (2018) are able to show that their methods obtain near-optimal solutions. The use of benchmark problems also allows for the comparison of the number of synthesized constraints with

the actual number of constraints from the benchmark problems (Pawlak and Krawiec 2017a, 2018, Pawlak 2019, Karmelita and Pawlak 2020). Moreover, they are able to assess the similarity of the syntaxes by computing the mean angle between weight vectors of the corresponding constraints in the synthesized and actual models.

The feasible regions of the synthesized models can also be compared to those of the actual feasible regions using the Jaccard Index (Jaccard 1912), as it is used to measure the similarity between sets. This is used in Kudła and Pawlak (2018), Pawlak and Krawiec (2017a), and others. Visual comparisons (limited to three dimensions) have also been used to illustrate how well a constraint learning approach captures the real feasible region (*e.g.*, in Pawlak and Litwiniuk (2021)). In Schede et al. (2019), the authors compute the probabilities that feasible (and infeasible) points of the benchmark problem lie in the feasible (and infeasible) region of the learned problem. They also compare the differences between the optimum values of the original and learned problems. Gutierrez-Martinez et al. (2010), Cremer et al. (2018) and Schweidtmann and Mitsos (2019) similarly also test their approaches on benchmark problems, and compare their results to the known solutions of the problems. A similar approach can be taken to evaluate the effect of constraint learning on the generated treatment plans for the radiotherapy example of Section 2.2.2. The new treatment plans can be compared to conventional ones using a dose-volume histogram (DVH) (Drzymala et al. 1991). In addition to using benchmark problems, Xavier et al. (2021) also evaluate the performance of the method by using out-of-distribution data to measure the robustness against moderate dataset shift.

Using synthetic or well-known benchmark problems allows one to accurately determine how far learned constraints are from the real ones (Kumar et al. 2019a), however, this is not always possible. Consequently, other model verification procedures are used in the literature. Simulators can also be used to evaluate constraint learning approaches by comparing learned solutions to solutions from the simulator (*e.g.*, in Verwer et al. (2017)).

Authors also compare the constraint learning approaches with the typical approaches used to solve those types of problems. This is done in Say et al. (2017), for example, where they compare their constraint learning approach with the rule-based approach commonly used to solve their type of problem. Cozad et al. (2014) implement model improvement by iteratively generating new data and updating the model with these data. The process continues until the error is less than a specified tolerance value. An overview of the approaches used for verification and improvement of the learned models is given in Table 2.4.

Table 2.4. Approaches used for model verification and improvement

	Benchmark Problems	Compare Constraints and/or Feasible Regions	Simulator Results	Compare to Other Approaches	Adaptive Sampling	Other
Bergman et al. (2022)	x					
Chatzivasileiadis (2020)	x			x		
Chen et al. (2020b)	x		x	x		
Chi et al. (2007)			x			
Cozad et al. (2014)			x		x	
Cremer et al. (2018)	x		x	x		
De Angelis et al. (2003)			x		x	
Fahmi and Cremaschi (2012)			x			
Garg et al. (2018)	x			x		
Grimstad and Andersson (2019)	x					
Gutierrez-Martinez et al. (2010)	x			x		
Halilbašić et al. (2018)	x			x		
Jalali et al. (2019)	x			x		
Karmelita and Pawlak (2020)	x	x		x		
Kudła and Pawlak (2018)	x	x		x		
Kumar et al. (2019b)	x					
Kumar et al. (2019a)	x			x		
Kumar et al. (2021)	x			x		
Lombardi et al. (2017)			x	x		
Maragno et al. (2023)	x			x		
Mišić (2020)				x		
Venzke et al. (2020b)	x			x		
Paulus et al. (2021)	x					
Pawlak and Krawiec (2017a)	x	x				
Pawlak and Krawiec (2017b)	x	x				
Pawlak and Krawiec (2018)	x	x				
Pawlak (2019)	x	x		x		
Pawlak and Litwiniuk (2021)	x	x		x		
Pawlak and O'Neill (2021)	x	x		x		
Prat and Chatzivasileiadis (2020)	x			x		
Say et al. (2017)	x			x		
Schede et al. (2019)	x	x		x		
Schweidtmann and Mitsos (2019)	x			x		
Sroka and Pawlak (2018)	x	x		x		
Thams et al. (2017)	x			x		
Verwer et al. (2017)			x			
Xavier et al. (2021)	x					x
Yang and Bequette (2021)	x			x		

2.5 Challenges and opportunities in constraint learning

As the process of constraint learning is relatively new, it stands to reason that there are several challenges or limitations that need to be taken into consideration. The primary challenge is that predictive models are designed for predictive purposes, and not to be embedded into an optimization model. Other challenges that new practitioners should keep in mind include:

1. Constraint violation: The optimal solution can be infeasible in reality if the learned constraints do not approximate correctly the true constraints.

2. Explainability: Although optimization models are usually considered explainable, and therefore understandable to humans, the use of learned constraints makes the optimization model difficult to understand and the optimal solution potentially difficult to explain.

3. Data availability: A dataset of historical solutions is required to fit a predictive model. Some predictive models require more data than others to have acceptable performances. Obtaining the data for training purposes might be difficult in certain cases. Additionally in certain applications, one-class data (*i.e.*, data on only feasible (or infeasible states)) may be more available than two-class data (*i.e.*, data on both feasible and infeasible states). Potential practitioners of OCL should be aware of what approaches work best in either case.

4. Computational complexity: Some predictive models require auxiliary variables and many additional constraints in order to be embedded into the optimization model. The number and class of additional constraints and auxiliary variables have a direct effect on the computational complexity of the optimization model.

5. Confounders: The use of predictive models to define part of the optimization model should account for potential confounders. It has been noted by de Mast et al. (2022) that machine learning models are correlational models and not causal models. The fact that certain values of the features co-occur with certain values of the outcome does not imply that changing the value of the features would change the outcome. These features could be causal, however, there is no guarantee of this. Therefore, whenever a causal model on the features space is available, it should be considered in the embedding of the predictive model.

In light of these challenges, some of the opportunities for further research are presented below. The opportunities are discussed in light of the steps of our proposed framework.

Data Gathering and Preprocessing: There is an opportunity to understand the effects of the data generation and preprocessing phase on the solution of the optimization problem. Is there a particular strategy (*e.g.*, DoCE, Kriging, etc.) that performs best for constraint learning? It is also necessary to understand the size and type of data that is needed in order to get a good predictive model for constraint learning, and not just for prediction.

Selection and Training: Firstly, the choice of which model to use to learn constraints should also be further investigated. Most of the approaches used to learn constraints are the most commonly known methods (regression, ANNs, DTs, etc.), however, there is an opportunity to see how less well-known approaches perform in terms of solution quality, as well as ease of embedding as constraints. Secondly, predictive models often provide a measure of the uncertainty of their predictions. Neural network classifiers, for example, can have Softmax scores (Grave et al. 2017) while decision trees report the number of misclassified examples at their leaf nodes. Although Cremer et al. (2018) have used probabilistic information from their classifier to good results, further research on how to incorporate such measures of uncertainty into optimization problems should be carried out. Approaches such as robust optimization or stochastic programming might be useful here. Thirdly, as noted above, predictive models are designed for predictive purposes, and not to be embedded into an optimization model. There could be an opportunity to develop models designed to be embedded, as well as models that perform well with small amounts of data. Finally, only a few approaches have used ensemble methods to learn constraints. While not all problems will require the additional predictive power that an ensemble approach brings, it is to be expected that as the best predictive models in ML applications are often ensembles, it makes sense to incorporate their superior abilities into constraint learning.

Resolution of the Optimization Model: Several approaches for embedding ML models as constraints have been seen, however, can these models be embedded in more efficient ways? The difficulty of using larger neural networks or larger decision trees to learn constraints might be overcome if more efficient embedding procedures are developed. In Schweidtmann and Mitsos (2019), difficult parts of the neural network (`tanh` activation functions) were hidden from the B&B solver. Further research on hiding unnecessary information from solvers is needed. Further research is also needed on providing additional useful information (such as pre-computed derivatives) to commercial solvers so as to improve solution time and quality. This will be useful when the incorporation of predictive models makes these computations impractical or expensive.

Verification and Improvement: In the reviewed literature, methods for model verifica-

tion and improvement, usually consist of either comparing to benchmark problems, or to other competing approaches. It could be promising to look into developing formal approaches for verifying learned constraint models. A framework for verifying neural network behavior was developed in Venzke and Chatzivasileiadis (2020), and research could be done to see if similar frameworks can be applied during the constraint learning process. In terms of model improvement, more focus should be put on improving models via processes such as active learning or adaptive sampling. In such cases, it would be interesting to see the effects (if any) of exploration versus exploitation when generating data on the overall solution of the optimization problem. Questions also arise with respect to the robustness of the solution – What is the sensitivity of the optimal solution with respect to the uncertainty in the learned constraint? How is the robustness of the solution related to the noise in the dataset? A structured and formal approach for verifying optimization problems with learned constraints can help to answer questions such as these.

2.6 Conclusions

OCL helps to capture constraints that would otherwise have been difficult to capture (Kumar et al. 2019b), providing there is data available. The benefits are clearly stated by Halilbašić et al. (2018) where their data-driven approach increases the feasible space, identifies the global optimum, and takes less time than conventional methods. There is however a need to go about the process of OCL in an organized manner, in order to avoid potential pitfalls. We have therefore provided a framework for OCL, which we believe will help to formalize and direct the process of OCL. Besides providing the framework, we reviewed the literature in light of the different steps of the framework, highlighting common trends in constraint learning, as well as possible areas for future research. It is our belief that this paper will help to guide future efforts in OCL, and be of benefit to the wider Operations Research community.

Chapter 3

Mixed-integer Optimization with Constraint Learning

3.1 Introduction

Mixed-integer optimization (MIO) is a powerful tool that allows us to optimize a given objective subject to various constraints. This general problem statement of optimizing under constraints is nearly universal in decision-making settings. Some problems have readily quantifiable and explicit objectives and constraints, in which case MIO can be directly applied. The situation becomes more complicated, however, when the constraints and/or objectives are not explicitly known.

For example, suppose we deal with cancerous tumors and want to prescribe a treatment regimen with a limit on toxicity; we may have observational data on treatments and their toxicity outcomes, but we have no natural function that relates the treatment decision to its resultant toxicity. We may also encounter constraints that are not directly quantifiable. Consider a setting where we want to recommend a diet, defined by a combination of foods and quantities, that is sufficiently "palatable." Palatability cannot be written as a function of the food choices, but we may have qualitative data on how well people "like" various potential dietary prescriptions. In both of these examples, we cannot directly represent the outcomes of interest as functions of our decisions, but we have *data* that relates the outcomes and decisions. This raises a question: how can we consider data to learn these functions?

In this work, we tackle the challenge of data-driven decision making through a combined ML and MIO approach. ML allows us to learn functions that relate decisions to outcomes of interest directly through data. Importantly, many popular ML methods result in functions that are MIO-representable, meaning that they can be embedded into MIO formulations. This MIO-representable class includes both linear and nonlinear models, allowing us to capture a broad set of underlying relationships in the data. While the idea of learning functions directly from data is core to the field of ML, data is often underutilized in MIO settings due to the need for functional relationships between decision variables and outcomes. We seek to bridge this gap through *constraint learning*; we propose a general framework that allows us to learn constraints and objectives directly from data, using ML, and to optimize decisions accordingly,

using MIO. Once the learned constraints have been incorporated into the larger MIO, we can solve the problem directly using off-the-shelf solvers.

The term *constraint learning*, used several times throughout this work, captures both constraints and objective functions. We are fundamentally learning functions to relate our decision variables to the outcome(s) of interest. The predicted values can then either be incorporated as constraints or objective terms; the model learning and embedding procedures remain largely the same. For this reason, we refer to them both under the same umbrella of *constraint learning*. We describe this further in Section 3.2.2.

3.1.1 Literature review

Previous work has demonstrated the use of various ML methods in MIO problems and their utility in different application domains. The simplest of these methods is the regression function, as the approach is easy to understand and easy to implement. Given a regression function learned from data, the process of incorporating it into an MIO model is straightforward, and the final model does not require complex reformulations. As an example, Bertsimas et al. (2016b) use regression models and MIO to develop new chemotherapy regimens based on existing data from previous clinical trials. Kleijnen (2015) provides further information on this subject.

More complex ML models have also been shown to be MIO-representable, although more effort is required to represent them than simple regression models. Neural networks which use the ReLU activation function can be represented using binary variables and big-M formulations (Amos et al. 2017, Grimstad and Andersson 2019, Anderson et al. 2020, Chen et al. 2020b, Spyros 2020, Venzke et al. 2020b). Where other activation functions are used (Gutierrez-Martinez et al. 2010, Lombardi et al. 2017, Schweidtmann and Mitsos 2019), the MIO representation of neural networks is still possible, provided the solvers are capable of handling these functions.

With decision trees, each path in the tree from root to leaf node can be represented using one or more constraints (Bonfietti et al. 2015, Verwer et al. 2017, Halilbašić et al. 2018). The number of constraints required to represent decision trees is a function of the tree size, with larger trees requiring more linearizations and binary variables. The advantage here, however, is that decision trees are known to be highly interpretable, which is often a requirement of ML in critical application settings (Thams et al. 2017). Random forests (Biggs et al. 2021, Mišić 2020) and other tree ensembles (Cremer et al. 2018) have also been used in MIO in the same way as decision trees, with one set of constraints for each tree in the forest/ensemble along with one or more additional aggregate constraints.

Data for constraint learning can either contain information on continuous data, feasible and infeasible states (two-class data), or only one state (one-class data). The problem of learning functions from one-class data and embedding them into optimiza-

tion models has been recently investigated with the use of decision trees (Kudła and Pawlak 2018), genetic programming (Pawlak and Krawiec 2018), local search (Sroka and Pawlak 2018), evolutionary strategies (Pawlak 2019), and a combination of clustering, principal component analysis and wrapping ellipsoids (Pawlak and Litwiniuk 2021).

The above selected applications generally involve a single function to be learned and a fixed ML method for the model choice. Verwer et al. (2017) use two model classes (decision trees and linear models) in a specific auction design application, but in this case the models were determined a priori. Some authors have presented a more general framework of embedding learned ML models in optimization problems such as JANOS (Bergman et al. 2022) and EML (Lombardi et al. 2017), but in practice these works are restricted to limited problem structures and learned model classes. We take a broader perspective, proposing a comprehensive end-to-end pipeline, called `OptiCL`, that encompasses the full ML and optimization components of a data-driven decision making problem. In contrast to EML and JANOS, `OptiCL` supports a wider variety of predictive models — neural networks (with ReLU), linear regression, logistic regression, decision trees, random forests, gradient boosted trees and linear support vector machines. `OptiCL` is also more flexible than JANOS, as it can handle predictive models as constraints, and it also incorporates new concepts to deal with uncertainty in the ML models. A comparison of `OptiCL` against JANOS and EML on two test problems is shown in Appendix E.

Our work falls under the umbrella of prescriptive analytics. Bertsimas and Kallus (2020) and Elmachtoub and Grigas (2022) leverage ML model predictions as inputs into an optimization problem. Our approach is distinct from existing work in that we directly embed ML models rather than extracting predictions, allowing us to optimize our decisions over the model. In the broadest sense, our framework relates to work that jointly harnesses ML and MIO, an area that has garnered significant interest in recent years in both the optimization and machine learning communities (Bengio et al. 2021).

3.1.2 Contributions

Our work unifies several research areas in a comprehensive manner. Our key contributions are as follows:

1. We develop an end-to-end framework that takes data and directly implements model training, model selection, integration into a larger MIO, and ultimately optimization. We make this available as an open-source software, `OptiCL` (Optimization with Constraint Learning) to provide a practitioner-friendly tool for making better data-driven decisions. The code is available at `https://github.com/hwiberg/OptiCL`. The software encompasses the full ML

and optimization pipeline with the goal of being accessible to end users as well as extensible by technical researchers. Our framework natively supports models for both regression and classification functions and handles constraint learning in cases with both one-class and two-class data. We implement a cross-validation procedure for function learning that selects from a broad set of model classes. We also implement the optimization procedure in the generic mathematical modeling library Pyomo, which supports various state-of-the-art solvers. We introduce two approaches for handling the inherent uncertainty when learning from data. First, we propose an ensemble learning approach that enforces constraint satisfaction over an ensemble of multiple bootstrapped estimators or multiple algorithms, yielding more robust solutions. This addresses a shortcoming of existing approaches to embedding trained ML models, which rely on a single point prediction: in the case of learned constraints, model misspecification can lead to infeasibility. Additionally, we restrict solutions to lie within a trust region, defined as the domain of the training data, which leads to better performance of the learned constraints. We offer several improvements to a basic convex hull formulation, including a clustering heuristic and a column selection algorithm that significantly reduce computation time. We also propose an enlargement of the convex hull which allows for exploration of solutions outside of the observed bounds. Both the ensemble model wrapper and trust region enlargement are controlled by parameters that allow an end user to directly trade-off the conservativeness of the constraint satisfaction.

2. We demonstrate the power of our method in two real-world case studies, using data from the World Food Programme and chemotherapy clinical trials. We pose relevant questions in the respective areas and formalize them as constraint learning problems. We implement our framework and subsequently evaluate the quantitative performance and scalability of our methods in these settings.

3.2 Embedding predictive models

Suppose we have data $\mathcal{D} = \{(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{w}}_i, \bar{\boldsymbol{y}}_i)\}_{i=1}^N$, with observed treatment decisions $\bar{\boldsymbol{x}}_i$, contextual information $\bar{\boldsymbol{w}}_i$, and outcomes of interest $\bar{\boldsymbol{y}}_i$ for sample i. Following the guidelines proposed in Fajemisin et al. (2024), we present a framework that, given data $\mathcal{D}$, learns functions for the outcomes of interest $(\boldsymbol{y})$ that are to be constrained or optimized. These learned representations can then be used to generate predictions for a new observation with context $\boldsymbol{w}$. Figure 3.1 outlines the complete pipeline, which is detailed in the sections below.

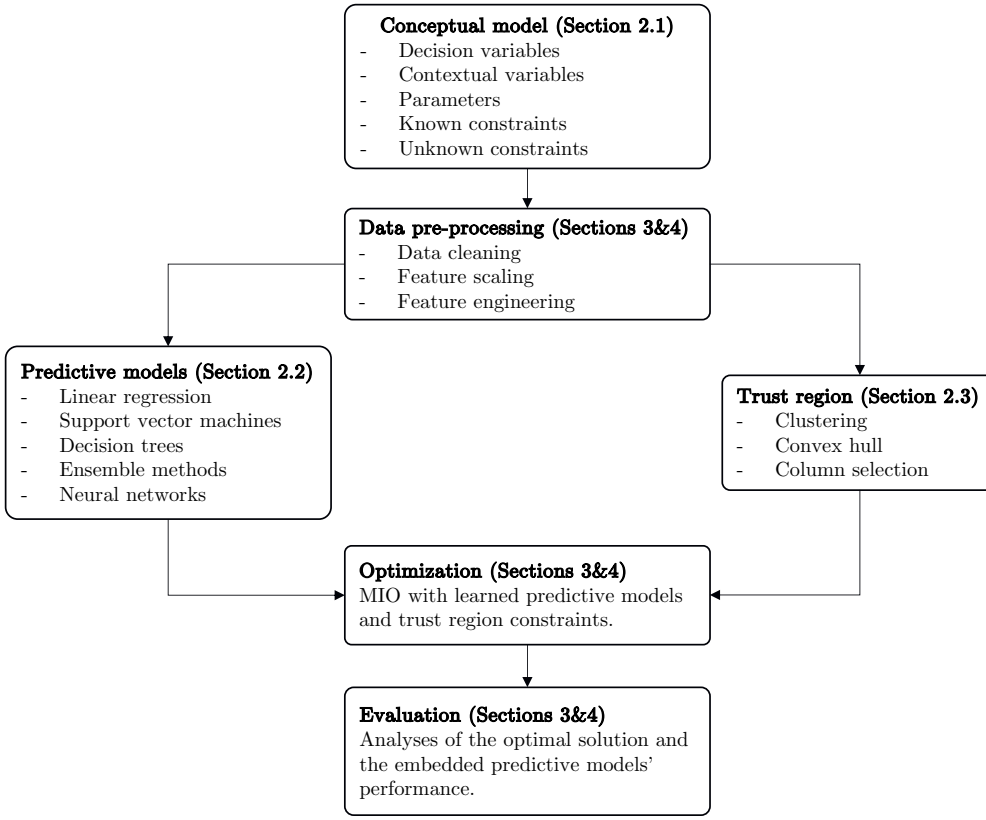


Figure 3.1. Constraint learning and optimization pipeline.

3.2.1 Conceptual model

Given the decision variable $\boldsymbol{x} \in \mathbb{R}^n$ and the fixed feature vector $\boldsymbol{w} \in \mathbb{R}^p$, we propose model M($\boldsymbol{w}$)

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{y}\in\mathbb{R}^k} \; & f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}) \\ \text{s.t. } & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}) \leq \boldsymbol{0}, \\ & \boldsymbol{y} = \hat{\boldsymbol{h}}_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{w}), \\ & \boldsymbol{x} \in \mathcal{X}(\boldsymbol{w}), \end{aligned} \tag{3.1}$$

where $f(., \boldsymbol{w}, .) : \mathbb{R}^{n+k} \mapsto \mathbb{R}$, $\boldsymbol{g}(., \boldsymbol{w}, .) : \mathbb{R}^{n+k} \mapsto \mathbb{R}^m$, and $\hat{\boldsymbol{h}}_{\mathcal{D}}(., \boldsymbol{w}) : \mathbb{R}^n \mapsto \mathbb{R}^k$. Explicit forms of f and $\boldsymbol{g}$ are known but they may still depend on the predicted outcome $\boldsymbol{y}$. Here, $\hat{\boldsymbol{h}}_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{w})$ represents the predictive models, one per outcome of interest, which are ML models trained on $\mathcal{D}$. Although our subsequent discussion mainly revolves around linear functions, we acknowledge the significant progress in nonlinear (convex) integer solvers. Our discussion can be easily extended to nonlinear models that can be tackled by those ever-improving solvers.

We note that the embedding of a single learned outcome may require multiple constraints and auxiliary variables; the embedding formulations are described in Section 3.2.2. For simplicity, we omit $\mathcal{D}$ in further notation of $\hat{\boldsymbol{h}}$ but note that all references to $\hat{\boldsymbol{h}}$ implicitly depend on the data used to train the model. Finally, the set $\mathcal{X}(\boldsymbol{w})$ defines the trust region, *i.e.*, the set of solutions for which we trust the embedded predictive models. In Section 3.2.3, we provide a detailed description of how the trust region $\mathcal{X}(\boldsymbol{w})$ is obtained from the observed data. We refer to the final MIO formulation with the embedded constraints and variables as EM($\boldsymbol{w}$).

Model M($\boldsymbol{w}$) is quite general and encompasses several important *constraint learning* classes:

1. **Regression.** When the trained model results from a regression problem, it can be constrained by a specified upper bound τ, *i.e.*, $g(y) = y - \tau \leq 0$, or lower bound τ, *i.e.*, $g(y) = -y + \tau \leq 0$. If $\boldsymbol{y}$ is a vector (*i.e.*, multi-output regression), we can likewise provide a threshold vector $\boldsymbol{\tau}$ for the constraints.

2. **Classification.** If the trained model is obtained with a binary classification algorithm, in which the data is labeled as "feasible" (1) or "infeasible" (0), then the prediction is generally a probability $y \in [0, 1]$. We can enforce a lower bound on the feasibility probability, *i.e.*, $y \geq \tau$. A natural choice of τ is 0.5, which can be interpreted as enforcing that the result is more likely feasible than not. This can also extend to the multi-class setting, say k classes, in which the output $\boldsymbol{y}$ is a k-dimensional unit vector, and we apply the constraint $y_i \geq \tau$ for whichever class i is desired. When multiple classes are considered to be feasible, we can

add binary variables to ensure that a solution is feasible, only if it falls in one of these classes with sufficiently high probability.

3. **Objective function.** If the objective function has a term that is also learned by training an ML model, then we can introduce an auxiliary variable $t \in \mathbb{R}$, and add it to the objective function along with an epigraph constraint. Suppose for simplicity that the model involves a single learned objective function, $\hat{h}$, and no learned constraints. Then the general model becomes

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n, y\in\mathbb{R}, t\in\mathbb{R}} \quad & t \\ \text{s.t. } & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w}) \leq 0, \\ & y = \hat{h}(\boldsymbol{x}, \boldsymbol{w}), \\ & y - t \leq 0, \\ & \boldsymbol{x} \in \mathcal{X}(\boldsymbol{w}). \end{aligned}$$

Although we have rewritten the problem to show the generality of our model, it is quite common in practice to use y in the objective and omit the auxiliary variable t.

We observe that constraints on learned outcomes can be applied in two ways depending on the model training approach. Suppose that we have a continuous scalar outcome y to learn and we want to impose an upper bound of $\tau \in \mathbb{R}$ (it may also be a lower bound without loss of generality). The first approach is called *function learning* and concerns all cases where we learn a regression function $\hat{h}(\boldsymbol{x}, \boldsymbol{w})$ without considering the feasibility threshold (τ). The resultant model returns a predicted value $y \in \mathbb{R}$. The threshold is then applied as a constraint in the optimization model as $y \leq \tau$. Alternatively, we could use the feasibility threshold τ to binarize the outcome of each sample in $\mathcal{D}$ into feasible and infeasible, that is $\bar{y}_i \coloneqq \mathbb{I}(\bar{y}_i \leq \tau),\ i = 1, \ldots, N$, where $\mathbb{I}$ stands for the indicator function. After this relabeling, we train a binary classification model $\hat{h}(\boldsymbol{x}, \boldsymbol{w})$ that returns a probability $y \in [0, 1]$. This approach, called *indicator function learning*, does not require any further use of the feasibility threshold τ in the optimization model, since the predictive models directly encode feasibility.

The function learning approach is particularly useful when we are interested in varying the threshold τ as a model parameter. Additionally, if the fitting process is expensive and therefore difficult to perform multiple times, learning an indicator function for each potential τ might be infeasible. In contrast, the indicator function learning approach is necessary when the raw data contains binary labels rather than continuous outcomes, and thus we have no ability to select or vary τ.

3.2.2 MIO-representable predictive models

Our framework is enabled by the ability to embed learned predictive models into an MIO formulation with linear constraints. This is possible for many classes of ML models, ranging from linear models to ensembles, and from support vector machines to neural networks. In this section, we outline the embedding procedure for decision trees, tree ensembles, and neural networks to illustrate the approach. We include additional technical details and formulations for these methods, along with linear regression and support vector machines, in Appendix A.

In all cases, the model has been *pre-trained*; we embed the trained model $\hat{h}(\boldsymbol{x}, \boldsymbol{w})$ into our larger MIO formulation to allow us to constrain or optimize the resultant predicted value. Consequently, the optimization model is not dependent on the complexity of the model training procedure, but solely the size of the final trained model. Without loss of generality, we assume that y is one-dimensional; *i.e.*, we are learning a single model, and this model returns a scalar, not a multi-output vector.

All of the methods below can be used to learn constraints that apply upper or lower bounds to y, or to learn y that we incorporate as part of the objective. We present the model embedding procedure for both cases when $\hat{h}(\boldsymbol{x}, \boldsymbol{w})$ is a continuous or a binary predictive model, where relevant. We assume that either regression or classification models can be used to learn feasibility constraints, as described in Section 3.2.1.

Decision Trees. Decision trees partition observations into distinct *leaves* through a series of *feature splits*. These algorithms are popular in predictive tasks due to their natural interpretability and ability to capture nonlinear interactions among variables. Breiman et al. (1984) first introduced Classification and Regression Trees (CART), which constructs trees through parallel splits in the feature space. Decision tree algorithms have subsequently been adapted and extended. Bertsimas and Dunn (2017) propose an alternative decision tree algorithm, Optimal Classification Trees (and Optimal Regression Trees), that improves on the basic decision tree formulation through an optimization framework that approximates globally optimal trees. Optimal trees also support multi-feature splits, referred to as *hyper-plane splits*, that allow for splits on a linear combination of features (Bertsimas, D. and Dunn, J. 2018).

A generic decision tree of depth 2 is shown in Figure 3.2. A split at node i is described by an inequality $A_i^\top \boldsymbol{x} \leq b_i$. We assume that A can have multiple non-zero elements, in which we have the hyper-plane split setting; if there is only one non-zero element, this creates a parallel (single feature) split. Each terminal node j (*i.e.*, leaf) yields a prediction (p_j) for its observations. In the case of regression, the prediction is the average value of the training observations in the leaf, and in binary classification, the prediction is the proportion of leaf members with the feasible class. Each leaf can be described as a polyhedron, namely a set of linear constraints that must be satisfied by

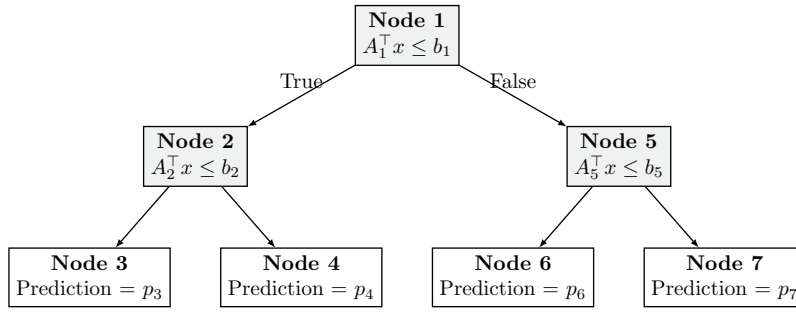


Figure 3.2. A decision tree of depth 2 with four terminal nodes (leaves).

all leaf members. For example, for node 3, we define $\mathcal{P}_3 = \left\{x : A_1^\top x \leq b_1, A_2^\top x \leq b_2\right\}$.

Suppose that we wish to constrain the predicted value of this tree to be at most τ, a fixed constant. After obtaining the tree in Figure 3.2, we can identify which paths satisfy the desired bound ($p_i \leq \tau$). Suppose that p_3 and p_6 do satisfy the bound, but p_4 and p_7 do not. In this case, we can enforce that our solution belongs to $\mathcal{P}_3$ or $\mathcal{P}_6$. This same approach applies if we only have access to two-class data (feasible vs. infeasible); we can directly train a binary classification algorithm and enforce that the solution lies within one of the "feasible" prediction leaves (determined by a set probability threshold).

If the decision tree provides our only learned constraint, we can decompose the problem into multiple separate MIOs, one per feasible leaf. The conceptual model for the subproblem of leaf i then becomes

$$\begin{aligned} \min_{\boldsymbol{x}} \; & f(\boldsymbol{x}, \boldsymbol{w}) \\ \text{s.t. } & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w}) \leq 0, \\ & (\boldsymbol{x}, \boldsymbol{w}) \in \mathcal{P}_i, \end{aligned}$$

where the learned constraints for leaf i's subproblem are implicitly represented by the polyhedron $\mathcal{P}_i$. These subproblems can be solved in parallel, and the minimum across all subproblems is obtained as the optimal solution. Furthermore, if all decision variables $\boldsymbol{x}$ are continuous, these subproblems are linear optimization problems (LOs), which can provide substantial computational gains. This is explored further in Appendix A.2.

In the more general setting where the decision tree forms one of many constraints, or we are interested in varying the τ limit within the model, we can directly embed the model into a larger MIO. We add binary variables representing each leaf, and set y

to the predicted value of the assigned leaf. An observation can only be assigned to a leaf, if it obeys all of its constraints; the structure of the tree guarantees that exactly one path will be fully satisfied, and thus, the leaf assignment is uniquely determined. A solution belonging to $\mathcal{P}_3$ will inherit $y = p_3$. Then, y can be used in a constraint or objective. The full formulation for the embedded decision tree is included in Appendix A.2. This formulation is similar to the proposal in Verwer et al. (2017). Both approaches have their own merits: while the Verwer formulation includes fewer constraints in the general case, our formulation is more efficient in the case where the problem can be decomposed into individual subproblems (as described above).

Ensemble Methods. Ensemble methods, such as random forests (RF) and gradient-boosting machines (GBM) consist of many decision trees that are aggregated to obtain a single prediction for a given observation. These models can thus be implemented by embedding many "sub-models" (Breiman 2001). Suppose we have a forest with P trees. Each tree can be embedded as a single decision tree (see previous paragraph) with the constraints from Appendix A.2, which yields a predicted value y_i.

RF models typically generate predictions by taking the average of the predictions from the individual trees:

$$y = \frac{1}{P} \sum_{i=1}^{P} y_i.$$

This can then be used as a term in the objective, or constrained by an upper bound as $y \leq \tau$; this can be done equivalently for a lower bound. In the classification setting, the prediction averages the probabilities returned by each model ($y_i \in [0, 1]$), which can likewise be constrained or optimized.

Alternatively, we can further leverage the fact that unlike the other model classes, which return a single prediction, the RF model generates P predictions, one per tree. We can impose a violation limit across the individual P estimators as proposed in Section 3.3.1.

In the case of GBM, we have an ensemble of base-learners which are not necessarily decision trees. The model output is then computed as

$$y = \sum_{i=1}^{P} \beta_i y_i,$$

where y_i is the predicted value of the i-th regression model $\hat{h}_i(\boldsymbol{x}, \boldsymbol{w})$, β_i is the weight associated with the prediction. Although trees are typically used as base-learners, in theory we might use any of the MIO-representable predictive models discussed in this section.

Neural Networks. We implement multi-layer perceptrons (MLP) with a rectified linear unit (ReLU) activation function, which form an MIO-representable class of neural networks (Grimstad and Andersson 2019, Anderson et al. 2020). These networks consist of an input layer, $L-2$ hidden layer(s), and an output layer. This nonlinear transformation of the input space over multiple nodes (and layers) using the ReLU operator ($v = \max\{0, x\}$) allows MLPs to capture complex functions that other algorithms cannot adequately encode, making them a powerful class of models.

Critically, the ReLU operator, $v = \max\{0, x\}$, can be encoded using linear constraints, as detailed in Appendix A.3. The constraints for an MLP network can be generated recursively starting from the input layer, which allows us to embed a trained MLP with an arbitrary number of hidden layers and nodes into an MIO. We refer to Appendix A.3 for details on the embedding of regression, binary classification, and multi-class classification MLP variants.

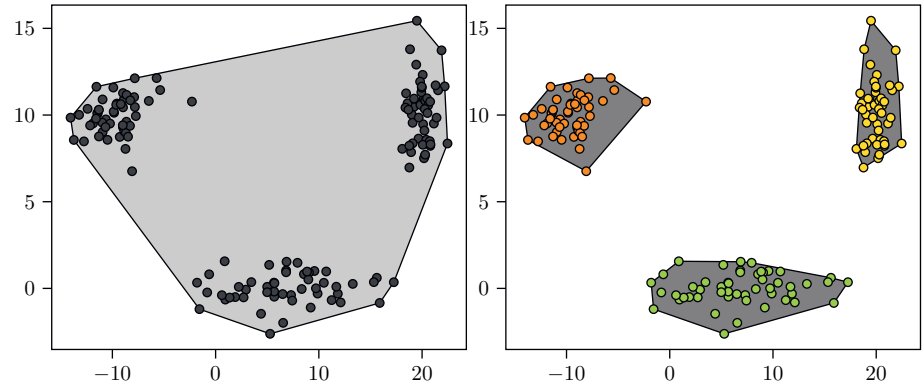
3.2.3 Convex hull as trust region

As the optimal solutions of optimization problems are often at the extremes of the feasible region, this can be problematic for the validity of the trained ML model. Generally speaking the accuracy of a predictive model deteriorates for points that are further away from the data points in $\mathcal{D}$ (Goodfellow et al. 2015). To mitigate this problem, we elaborate on the idea proposed by Biggs et al. (2021) to use the convex hull (CH) of the dataset as a trust region to prevent the predictive model from extrapolating. According to Ebert et al. (2014), when data is enclosed by a boundary of convex shape, the region inside this boundary is known as an interpolation region. This interpolation region is also referred to as the CH, and by excluding solutions outside the CH, we prevent extrapolation. If $\boldsymbol{X} = \{\hat{\boldsymbol{x}}_i\}_{i=1}^N$ is the set of observed input data with $\hat{\boldsymbol{x}}_i = (\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{w}}_i)$, we define the trust region as the CH of this set and denote it by $\text{CH}(\boldsymbol{X})$. Recall that $\text{CH}(\boldsymbol{X})$ is the smallest convex polytope that contains the set of points $\boldsymbol{X}$. It is well-known that computing the CH is exponential in time and space with respect to the number of samples and their dimensionality Skiena (2008). However, since the CH is a polytope, explicit expressions for its facets are not necessary. More precisely, $\text{CH}(\boldsymbol{X})$ is represented as

$$\text{CH}(\boldsymbol{X}) = \left\{ \boldsymbol{x} \,\middle|\, \sum_{i \in \mathcal{I}} \lambda_i \hat{\boldsymbol{x}}_i = \boldsymbol{x},\ \sum_{i \in \mathcal{I}} \lambda_i = 1,\ \boldsymbol{\lambda} \geq 0 \right\}, \tag{3.2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^N$, and $\mathcal{I} = \{1, \ldots, N\}$ is the index set of samples in $\boldsymbol{X}$.

In situations such as the one shown in Figure 3.3a, $\text{CH}(\boldsymbol{X})$ includes regions with few or no data points (low-density regions). Blindly using $\text{CH}(\boldsymbol{X})$ in this case can be problematic if the solutions are found in the low-density regions. We therefore advocate the use of a two-step approach. First, clustering is used to identify distinct



(a) CH($\boldsymbol{X}$) with single region. (b) CH($\boldsymbol{X}$) with clustered regions.

Figure 3.3. Use of the two-step approach to remove low-density regions.

high-density regions, and then the trust region is represented as the union of the CHs of the individual clusters (Figure 3.3b).

We can either solve EM($\boldsymbol{w}$) for each cluster, or embed the union of the $|\mathcal{K}|$ CHs into the MIO given by

$$\bigcup_{k\in\mathcal{K}} \text{CH}(\boldsymbol{X}_k) = \left\{ \boldsymbol{x} \,\middle|\, \sum_{i\in\mathcal{I}_k} \lambda_i \hat{\boldsymbol{x}}_i = \boldsymbol{x},\ \sum_{i\in\mathcal{I}_k} \lambda_i = u_k\ \forall k \in \mathcal{K}, \sum_{k\in\mathcal{K}} u_k = 1,\ \boldsymbol{\lambda} \geq 0,\ \boldsymbol{u} \in \{0,1\}^{|\mathcal{K}|} \right\}, \tag{3.3}$$

where $\boldsymbol{X}_k \subseteq \boldsymbol{X}$ refers to subset of samples in cluster $k \in \mathcal{K}$ with the index set $\mathcal{I}_k \subseteq \mathcal{I}$. The union of CHs requires the binary variables u_k to constrain a feasible solution to be exactly in one of the CHs. More precisely, $u_k = 1$ corresponds to the CH of the k-th cluster. As we show in Section 3.4, solving EM($\boldsymbol{w}$) for each cluster may be done in parallel, which has a positive impact on computation time. We note that both formulations (3.2) and (3.3) assume that $\hat{\boldsymbol{x}}$ is continuous. These formulations can be extended to datasets with binary, categorical and ordinal features. In the case of categorical features, extra constraints on the domain and one-hot encoding are required.

Although the CH can be represented by linear constraints, the number of variables in EM($\boldsymbol{w}$) increases with the increase in the dataset size, which may make the optimization process prohibitive when the number of samples becomes too large. We therefore provide a column selection algorithm that selects a small subset of the samples. This algorithm can be directly used in the case of convex optimization problems or embedded as part of a branch and bound algorithm when the optimization problem involves integer variables. Figure 3.4 visually demonstrates the procedure; we begin with an arbitrary sample of the full data, and use column selection to iteratively

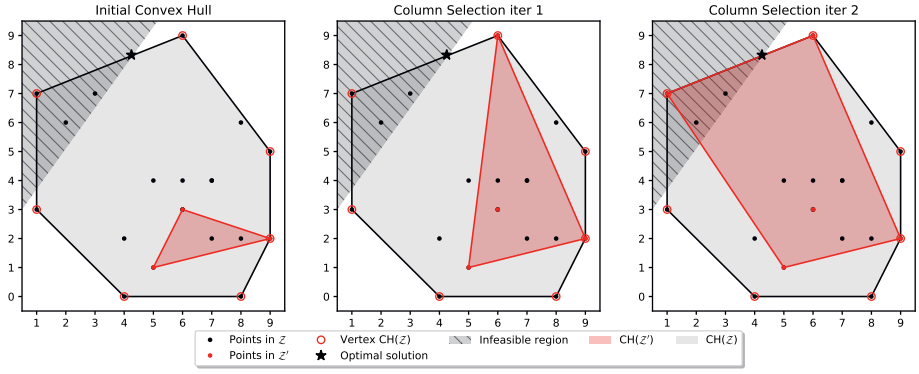


Figure 3.4. Visualization of the column selection algorithm. Known and learned constraints define the infeasible region. The column selection algorithm starts using only a subset of data points (red filled circles), $\boldsymbol{X}' \subseteq \boldsymbol{X}$ to define the trust region. In each iteration a vertex of CH($\boldsymbol{X}$) is selected (red hollow circle) and included in $\boldsymbol{X}'$ until the optimal solution (star) is within the feasible region, namely the convex hull of $\boldsymbol{X}'$. Note that with column selection we do not need the complete dataset to obtain the optimal solution, but rather only a subset.

add samples $\hat{\boldsymbol{x}}_i$ until no improvement can be found. In Appendix B.2, we provide a full description of the approach, as well as a formal lemma which states that in each iteration of column selection, the selected sample from $\boldsymbol{X}$ is also a vertex of CH($\boldsymbol{X}$). In synthetic experiments, we observe that the algorithm scales well with the dataset size. The computation time required by solving the optimization problem with the algorithm is near-constant and minimally affected by the number of samples in the dataset. The experiments in Appendix B.2 show optimization with column selection to be significantly faster than a traditional approach, which makes it an ideal choice when dealing with massive datasets.

3.3 Uncertainty and Robustness

There are multiple sources of uncertainty, and consequently notions of robustness, that can be considered when embedding a trained machine learning model as a constraint. We define two types of uncertainty in model (3.1).

Function Uncertainty. The first source of uncertainty is in the underlying functional form of $\hat{h}$. We do not know the ground truth relationship between $(\boldsymbol{x}, \boldsymbol{w})$ and y, and there is potential for model mis-specification. We mitigate this risk through our non-parametric model selection procedure, namely training $\hat{h}$ for a diverse set of methods

(*e.g.*, decision tree, regression, neural network) and selecting the final model using a cross-validation procedure.

Parameter Uncertainty. Even within a single model class, there is uncertainty in the parameter estimates that define $\hat{h}$. Consider the case of linear regression. A regression estimator consists of point estimates of coefficients and an intercept term, but there is uncertainty in the estimates as they are derived from noisy data. We seek to make our model robust by characterizing this uncertainty and optimizing against it. We propose model-wrapper ensemble approaches, which are agnostic to the underlying model. The rest of this section addresses the model-wrapper approaches and a looser formulation of the trust region that prevents the optimal solution from being too conservative when the predictive models have good extrapolation performance.

3.3.1 Model-wrapper approach

We begin by describing the model "wrapper" approach for characterizing uncertainty, in which we work directly with any trained models and their point predictions. Rather than obtaining our estimated outcome from a single trained predictive model, we suppose that we have P estimators. The set of estimators can be obtained by bootstrapping or by training models using entirely different methods. The uncertainty is thus characterized by different realizations of the predicted value from multiple estimators, which effectively form an ensemble.

We introduce a constraint that at most $\alpha \in [0, 1]$ proportion of the P estimators violate the constraint. Let $\hat{h}_1, \ldots, \hat{h}_P$ be the individual estimators. Then $\hat{h}_i(x) \leq \tau$ in at least $1 - \alpha P$ of these estimators. This allows for a degree of robustness to individual model predictions by discarding a small number of potential outlier predictions. Formally,

$$\frac{1}{P} \sum_{i=1}^{P} \mathbb{I}(y_i \leq \tau) \geq 1 - \alpha. \tag{3.4}$$

Note that $\alpha = 0$ enforces the bound for all estimators, yielding the most conservative estimate, whereas $\alpha = 1$ removes the constraint entirely. Constraint (3.4) is MIO-representable:

$$\begin{aligned} & y_i \leq \tau + M(1 - z_i), \quad i = 1, \ldots, P \\ & \frac{1}{P} \sum_{i=1}^{P} z_i \geq 1 - \alpha, \end{aligned}$$

where $z_i \in \{0, 1\}\ \forall i = 1, \ldots, P$, and M is a sufficiently large constant. Appendix A.4 includes further details on this formulation and special cases.

The violation limit concept can also be applied to estimators coming from multiple model classes, which allows us to enforce that the constraint is *generally* obeyed when modeled through distinct methods. This provides a measure of robustness to *function uncertainty*.

3.3.2 Enlarged convex hull

The use of the model-wrapper approach and the trust region constraints, as defined in (3.2), has a direct effect on the feasible region. The better performance of the learned constraints might be balanced out by the (potentially) unnecessary conservatism of the optimal solution. Although we introduced the trust region as a set of constraints to preserve the predictive performance of the fitted constraints, Balestriero et al. (2021) show how in a high-dimensional space the generalization performance of a fitted model is typically obtained extrapolating. In light of this evidence, we propose an ϵ-CH formulation which builds on (3.2), and more generally on (3.3). The relaxed formulation of the trust region enables the optimal solution of problem $M(\boldsymbol{w})$ to be outside $CH(X)$. Formally, we enlarge the trust region such that solutions outside CH($\boldsymbol{X}$) are considered feasible if they fall within the hyperball, with radius ϵ, surrounding at least one of the data points in $\boldsymbol{X}$, see Figure 3.5 (left). The ϵ-CH is formulated as follows:

$$\epsilon\text{-CH}(\boldsymbol{X}) = \left\{ (\boldsymbol{x}, \boldsymbol{s}) \middle| \sum_{i \in \mathcal{I}} \lambda_i \hat{\boldsymbol{x}}_i = \boldsymbol{x} + \boldsymbol{s},\ \sum_{i \in \mathcal{I}} \lambda_i = 1,\ \boldsymbol{\lambda} \geq 0,\ ||\boldsymbol{s}||_p \leq \epsilon \right\}, \tag{3.5}$$

with $\boldsymbol{s} \in \mathbb{R}^n$, and p set equal to 1,2 or ∞ to preserve the complexity of the optimization problem. Figure 3.5 (right) shows the extended region obtained with the ϵ-CH. The choice of ϵ is pivotal in the trade-off between the performance of the learned constraints and the conservatism of the optimal solution. In the next section, we demonstrate how an increase in ϵ affects both the performance of the embedded predictive models and the objective function value.

3.4 Case study: a palatable food basket for the World Food Programme

In this case study, we use a simplified version of the model proposed by Peters et al. (2021), which seeks to optimize humanitarian food aid. Its extended version aims to provide the World Food Programme (WFP) with a decision-making tool for long-term recovery operations, which simultaneously optimizes the food basket to be delivered, the sourcing plan, the delivery plan, and the transfer modality of a month-long food supply. The model proposed by Peters et al. (2021) enforces that the food baskets ad-

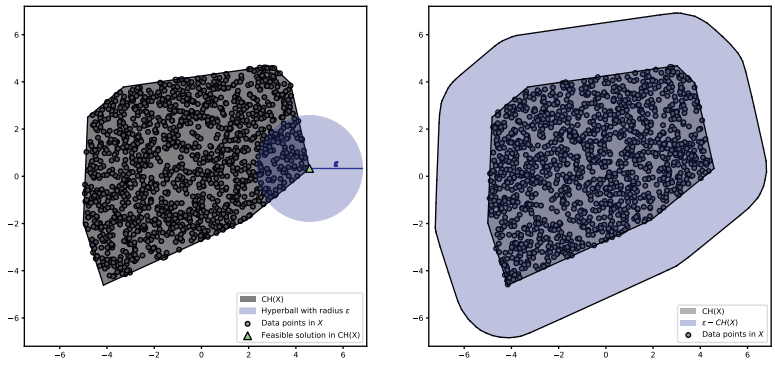


Figure 3.5. Trust region enlarged using an hyperball with radius ϵ around each sample in CH(X).

dress the nutrient gap and are palatable. To guarantee a certain level of palatability, the authors use a number of "unwritten rules" that have been defined in collaboration with nutrition experts. In this case study, we take a step further by inferring palatability constraints directly from data that reflects local people's opinions. We use the specific case of Syria for this example. The conceptual model presents an LO structure with only the food palatability constraint to be learned. Data on palatability is generated through a simulator, but the procedure would remain unchanged if data were collected in the field, for example through surveys. The structure of this problem, which is an LO and involves only one learned constraint, allows the following analyses: (1) the effect of the trust-region on the optimal solution, and (2) the effect of clustering on the computation time and the optimal objective value. Additionally, the use of simulated data provides us with a ground truth to use in evaluating the quality of the prescriptions.

3.4.1 Conceptual model

The optimization model is a combination of a capacitated, multi-commodity network flow model, and a diet model with constraints for nutrition levels and food basket palatability.

The sets used to define the constraints and the objective function are displayed in Table 3.1. We have three different sets of nodes, and the set of commodities contains all the foods available for procurement during the food aid operation.

The parameters used in the model are displayed in Table 3.2. The costs used in the objective function concern transportation (p^T) and procurement (p^P). The amount of

Sets	
$\mathcal{N}_\mathcal{S}$	Set of source nodes
$\mathcal{N}_\mathcal{T}$	Set of transshipment nodes
$\mathcal{N}_\mathcal{D}$	Set of delivery nodes
$\mathcal{K}$	Set of commodities ($k \in \mathcal{K}$)
$\mathcal{L}$	Set of nutrients ($l \in \mathcal{L}$)

Table 3.1. Definition of the sets used in the WFP model.

food to deliver depends on the demand (d) and the number of feeding days $(days)$. The nutritional requirements $(nutreq)$ and nutritional values $(nutrval)$ are detailed in Appendix C. The parameter γ is needed to convert the metric tons used in the supply chain constraints to the grams used in the nutritional constraints. The parameter t is used as a lower bound on the food basket palatability. The values of these parameters are based on those used by Peters et al. (2021).

Parameters	
γ	Conversion rate from metric tons (mt) to grams (g)
d_i	Number of beneficiaries at delivery point $i \in \mathcal{N}_\mathcal{D}$
$days$	Number of feeding days
$nutreq_l$	Nutritional requirement for nutrient $l \in \mathcal{L}$ (grams/person/day)
$nutval_{kl}$	Nutritional value for nutrient $l \in \mathcal{L}$ per gram of commodity $k \in \mathcal{K}$
p^P_{ik}	Procurement cost (in \$ / mt) of commodity k from source $i \in \mathcal{N}_\mathcal{S}$
p^T_{ijk}	Transportation cost (in \$ / mt) of commodity k from node $i \in \mathcal{N}_\mathcal{S} \cup \mathcal{N}_\mathcal{T}$ to node $j \in \mathcal{N}_\mathcal{T} \cup \mathcal{N}_\mathcal{D}$
t	Palatability lower bound

Table 3.2. Definition of the parameters used in the WFP model.

The decision variables are shown in Table 3.3. The flow variables F_{ijk} are defined as the metric tons of a commodity k transported from node i to j. The variable x_k represents the average daily ration per beneficiary for commodity k. The variable y refers to the palatability of the food basket.

Variables	
F_{ijk}	Metric tons of commodity $k \in \mathcal{K}$ transported between node i and node j
x_k	Grams of commodity $k \in \mathcal{K}$ in the food basket
y	Food basket palatability

Table 3.3. Definition of the variables used in the WFP model.

The full model formulation is as follows:

$$\min_{\boldsymbol{x},y,\boldsymbol{F}} \sum_{i\in\mathcal{N}_\mathcal{S}} \sum_{j\in\mathcal{N}_\mathcal{T}\cup\mathcal{N}_\mathcal{D}} \sum_{k\in\mathcal{K}} p^P_{ik} F_{ijk} + \sum_{i\in\mathcal{N}_\mathcal{S}\cup\mathcal{N}_\mathcal{T}} \sum_{j\in\mathcal{N}_\mathcal{T}\cup\mathcal{N}_\mathcal{D}} \sum_{k\in\mathcal{K}} p^T_{ijk} F_{ijk} \tag{3.6a}$$

$$\text{s.t.} \sum_{j \in \mathcal{N}_\mathcal{T}} F_{ijk} = \sum_{j \in \mathcal{N}_\mathcal{T}} F_{jik}, \quad i \in \mathcal{N}_\mathcal{T},\ k \in \mathcal{K}, \tag{3.6b}$$

$$\sum_{j \in \mathcal{N}_\mathcal{S} \cup \mathcal{N}_\mathcal{T}} \gamma F_{jik} = d_i x_k days, \quad i \in \mathcal{N}_\mathcal{D},\ k \in \mathcal{K}, \tag{3.6c}$$

$$\sum_{k \in \mathcal{K}} Nutval_{kl} x_k \geq Nutreq_l, \quad l \in \mathcal{L}, \tag{3.6d}$$

$$x_{salt} = 5, \tag{3.6e}$$

$$x_{sugar} = 20, \tag{3.6f}$$

$$y \geq t, \tag{3.6g}$$

$$y = \hat{h}(\boldsymbol{x}), \tag{3.6h}$$

$$F_{ijk}, x_k \geq 0, \quad i, j \in \mathcal{N},\ k \in \mathcal{K}. \tag{3.6i}$$

The objective function consists of two components, procurement costs and transportation costs. Constraints (3.6b) are used to balance the network flow, namely to ensure that the inflow and the outflow of a commodity are equal for each transhipment node. Constraints (3.6c) state that flow into a delivery node has to be equal to its demand, which is defined by the number of beneficiaries times the daily ration for commodity k times the feeding days. Constraints (3.6d) guarantee an optimal solution that meets the nutrition requirements. Constraints (3.6e) and (3.6f) force the amount of salt and sugar to be 5 grams and 20 grams respectively. Constraint (3.6g) requires the food basket palatability (y), defined by means of a predictive model (3.6h), to be greater than a threshold (t). Lastly, non-negativity constraints (3.6i) are added for all commodity flows and commodity rations.

3.4.2 Dataset and predictive models

To evaluate the ability of our framework to learn and implement the palatability constraints, we use a simulator to generate diets with varying palatabilities. Each sample is defined by 25 features representing the amount (in grams) of all commodities that make up the food basket. We then use a ground truth function to assign each food basket a palatability between 0 and 1, where 1 corresponds to a perfectly palatable basket, and 0 to an inedible basket. This function is based on suggestions provided by WFP experts and complete details are outlined in Appendix C.1. The data is then balanced to ensure that a wide variety of palatability scores are represented in the dataset. The final data used to learn the palatability constraint consists of 121,589 samples. Two examples of daily food baskets and their respective palatability scores are shown in Table 3.4. In this case study, we use a palatability lower bound (t) of 0.5 for our learned constraint.

The next step of the framework involves training and choosing the predictive model that best approximates the unknown constraint. The predictive models used to learn

Table 3.4. Two examples of daily food baskets.

Commodity	Basket 1 Amount (g)	Basket 2 Amount (g)
DSM	31.9	33.9
Chickpeas	–	75.7
Lentils	41	–
Maize meal	48.9	–
Meat	–	17.2
Oil	22	28.6
Salt	5	5
Sugar	20	20
Wheat	384.2	131.2
Wheat flour	–	261.3
WSB	67.3	59.8
Palatability Score	0.436	0.741

DSM=dried skim milk, WSB=wheat soya blend.

the palatability constraints are those discussed in Section 3.2, namely LR, SVM, CART, RF, GBM with decision trees as base-learners, and MLP with ReLU activation function.

3.4.3 Optimization results

The experiments are executed using `OptiCL` jointly with Gurobi v9.1 (Gurobi Optimization, LLC 2022) as the optimization solver. Table 3.5 reports the performances of the predictive models evaluated both for the validation set and for the prescriptions after being embedded into the optimization model. The table also compares the performance of the optimization with and without the trust region. The column "Validation MSE" gives the Mean Squared Error (MSE) of each model obtained in cross-validation during model selection. While all scores in this column are desirably low, the MLP model significantly achieves the lowest error during this validation phase. The column "MSE" gives the MSE of the predictive models once embedded into the optimization problem to evaluate how well the predictions for the optimal solutions match their true palatabilities (computed using the simulator). It is found using 100 optimal solutions of the optimization model generated with different cost vectors. The MLP model exhibits the best performance (0.055) in this context, showing its ability to model the palatability constraint better than all other methods.

Benefit of trust region. Table 3.5 shows that when the trust region is used ("MSE-TR"), the MSEs obtained by all models are now much closer to the results from the validation phase. This shows the benefit of using the trust region as discussed in Section 3.2.3 to prevent extrapolation. With the trust region included, the MLP model

Table 3.5. Predictive models performances for the validation set ("Validation MSE"), and for the prescriptions after being embedded into the optimization model with ("MSE-TR") and without the trust region ("MSE"). The last two columns show the average computation time in seconds and its standard deviation (SD) required to solve the optimization model with ("Time-TR") and without the trust region ("Time").

Model	Validation MSE	MSE	MSE-TR	Time (SD)	Time-TR (SD)
LR	0.046	0.256	0.042	0.003 (0.0008)	1.813 (0.204)
SVM	0.019	0.226	0.027	0.003 (0.0006)	1.786 (0.208)
CART	0.014	0.273	0.059	0.012 (0.0030)	7.495 (5.869)
RF	0.018	0.252	0.025	0.248 (0.1050)	30.128 (13.917)
GBM	0.006	0.250	0.017	0.513 (0.4562)	60.032 (41.685)
MLP	0.001	0.055	0.001	14.905 (41.764)	28.405 (23.339)

Runtimes reported using an Intel i7-8665U 1.9 GHz CPU, 16 GB RAM (Windows 10 environment).

also exhibits the lowest MSE (0.001). The improved performance seen with the inclusion of the trust region does come at the expense of computation speed. The column "Time-TR" shows the average computation time in seconds and its standard deviation (SD) with trust region constraints included. In all cases, the computation time has clearly increased when compared against the computation time required without the trust region (column "Time"). This is however acceptable, as significantly more accurate results are obtained with the trust region.

Benefit of clustering. The large dataset used in this case study makes the use of the trust region expensive in terms of time required to solve the final optimization model. While the column selection algorithm described in Section 3.2.3 is ideal for significantly reducing the computation time, optimization models that require binary variables, either for embedding an ML model or to represent decision variables, would require column selection to be combined with a branch and bound algorithm. However, in this more general MIO case, it is possible to divide the dataset into clusters and solve in parallel an MIO for each cluster. By using parallelization, the total solution time can be expected to be equal to the longest time required to solve any single cluster's MIO. Contrary to column selection, the use of clusters can result in more conservative solutions; the trust region gets smaller with more clusters and prevents the model from finding solutions that are convex combinations of members of different clusters. However, as described in Section 3.2.3, solutions that lie between clusters may in fact reside in low-density areas of the feature space that should not be included in the trust region. In this sense, the loss in the objective value might actually coincide with more trustable solutions.

Figure 3.6 shows the effect of clusters in solving the model (3.6a-3.6i) with GBM as the

predictive model used to learn the palatability constraint. K-means is used to partition the dataset into K clusters, and the reported values are averaged over 100 iterations. In the left graph, we report the maximum runtime distribution across clusters needed to solve the different MIOs in parallel. In the right graph, we have the distributions of optimality gap, *i.e.*, the relative difference between the optimal solution obtained with clusters compared to the solution obtained with no clustering. In this case study, the use of clusters significantly decreases the runtime (89.2% speed up with $K = 50$) while still obtaining near-optimal solutions (less then 0.25% average gap with $K = 50$). We observe that the trends are not necessarily monotonic in K. It is possible that a certain choice of K may lead to a suboptimal solution, whereas a larger value of K may preserve the optimal solution as the convex combination of points within a single cluster.

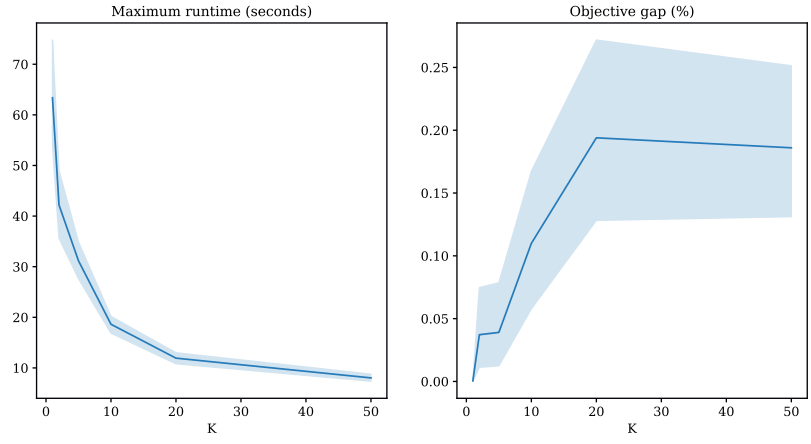


Figure 3.6. Effect of the number of clusters (K) on the computation time and the optimality gap across clusters, with bootstrapped 95% confidence intervals.

3.4.4 Robustness results

In these experiments, we assess the performance of the nominal and robust models. We consider three dimensions of performance: (1) true constraint satisfaction, (2) objective function value, and (3) runtime. The synthetic data used in this case study allows us to evaluate true palatability and constraint satisfaction as these parameters vary. This is the primary goal of the model wrapper ensemble approach, to improve feasibility and make solutions that are robust to any single learned estimator.

We hypothesize that as our models become more conservative, we will more reliably satisfy the desired palatability constraint with some toll on the objective function. Ad-

ditionally, embedding multiple models or characterizing uncertainty sets introduces computational complexity over a single nominal model. In this section, we compare the trade-offs in these metrics as we consider different notions of robustness and vary our conservativeness. We note that we are able to evaluate whether the true palatability meets the constraint threshold since palatability is defined through a known function. As with the experiments above, we solve the palatability problem with 100 different realizations of the cost vector and average the results.

The results below explore the effect of the α (violation limit) on cost and palatability in the WFP case study. Additional results on runtime, and experiments with varied estimators (P), are included in Appendix C.3. As the results demonstrate, the robustness parameters yield solutions that vary in their conservativeness and runtime. There is not a single set of optimal parameters. Rather, it is highly dependent on the use case, including factors like the stakes of the decision and the allowable turnaround time to generate solutions.

Multiple embedded models. We first consider the impact of the model-wrapper approach in the WFP problem. We compare different ways of embedding the palatability constraint, both using multiple estimators of a single model class and an ensemble containing multiple model classes. We run the experiments on a random sample of 1000 observations in the original WFP dataset. Within a single model class, we vary the number of estimators ($P \in [2, 5, 10, 25]$) and the violation limit ($\alpha \in [0, 0.1, 0.2, 0.5]$, or applying a mean constraint). Each estimator is obtained using a bootstrap sample (proportion = 0.5) of the underlying data. We compute metrics (1-3) for each variant to compare the tradeoffs in palatability (constraint satisfaction) and cost (objective function value).

Figure 3.7 presents the results for a decision tree with $P = 25$ and palatability threshold (τ) equal to 0.5. The left figure shows the trade off between palatability and the objective as the violation limit (α) varies. As expected, improvements in palatability (when α decreases) lead to increases in the total cost. However, we observe that a violation limit of 0.0 (vs. 0.5) leads to an 11.3% improvement in real palatability (20.8% improvement in predicted palatability), with a relatively modest 2.5% increase in cost. The center and right figure show how palatability and violations vary with α. Palatability increases and violations decease with lower α. Both the violation rate (proportion of iterations with real palatability < 0.5) and violation margin (average distance to palatability threshold in cases where there is a violation) decrease with lower α. This experiment demonstrates how the α parameter effectively controls the model's robustness as measured by constraint satisfaction. The approach has the advantage of parameterizing the violation limit, allowing us to explicitly control the model's conservativeness and evaluate constraint-objective tradeoffs. Appendix C.3 reports further results for other model classes as well as runtime experiments.

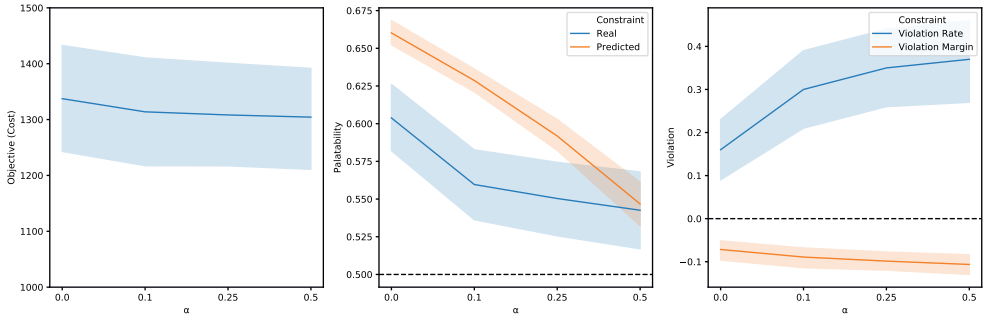


Figure 3.7. Comparison of CART models on objective function and constraint satisfaction.

Enlarged trust region. In order to evaluate the effects of the enlarged trust region on the optimal solution, we use a simplified version of problem (3.6a-3.6i) where the only constraints are on the predictive model embedding, the palatability lower bound, and the ϵ-CH. In Figure 3.8, we show how the objective function value and true palatability score vary according to different values of $\epsilon \in [0, 0.8]$. The results are obtained by averaging over 200 iterations with randomly generated cost vectors and using a decision tree as a predictive model to represent the palatability outcome. As expected, the objective value improves as ϵ increases. More interesting is the true palatability score which stays around the imposed lower bound of 0.5 for values of ϵ smaller than 0.25. This means that the predictive model is able to generalize even outside the CH as long as the optimal solution is not too far from it.

3.5 Case study: chemotherapy regimen design

In this case study, we extend the work of Bertsimas et al. (2016b) in the design of chemotherapy regimens for advanced gastric cancer. Late stage gastric cancer has a poor prognosis with limited treatment options (Yang et al. 2011). This has motivated significant research interest and clinical trials (National Cancer Institute 2021). In Bertsimas et al. (2016b), the authors pose the question of algorithmically identifying promising chemotherapy regimens for new clinical trials based on existing trial results. They construct a database of clinical trial treatment arms which includes cohort and study characteristics, the prescribed chemotherapy regimen, and various outcomes. Given a new study cohort and study characteristics, they optimize a chemotherapy regimen to maximize the cohort's survival subject to a constraint on overall toxicity. The original work uses linear regression models to predict survival and toxicity, and it constrains a single toxicity measure. In this work we leverage a richer class of ML methods and more granular outcome measures. This offers benefits through

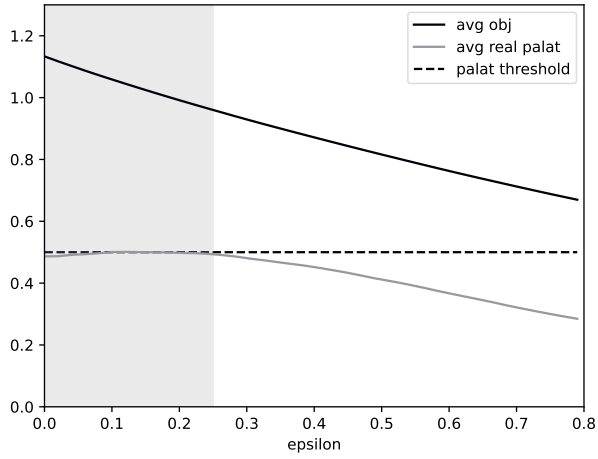


Figure 3.8. Effect of the ϵ-CH on the objective value and the predictive model performance with respect to the optimal solution. The values are obtained as an average of 200 iterations.

higher performing predictive models and more clinically-relevant constraints.

Chemotherapy regimens are particularly challenging to optimize, since they involve multiple drugs given at potentially varying dosages, and they present risks for multiple adverse events that must be managed. This example highlights the generalizability of our framework to complex domains with multiple decisions and learned functions. The treatment variables in this problem consist of both binary and continuous elements, which are easily incorporated through our use of MIO. We have several learned constraints which must be simultaneously satisfied, and we also learn the objective function directly as a predictive model.

3.5.1 Conceptual model

The use of clinical trial data forces us to consider each cohort as an observation, rather than an individual, since only aggregate measures are available. Thus, our model optimizes a cohort's treatment. The contextual variables ($\boldsymbol{w}$) consist of various cohort and study summary variables. The inclusion of fixed, *i.e.*, non-optimization, features allows us to account for differences in baseline health status and risk across study cohorts. These features are included in the predictive models but then are fixed in the optimization model to reflect the group for whom we are generating a prescription. We assume that there are no unobserved confounding variables in this prescriptive setting.

The treatment variables ($\boldsymbol{x}$) encode a chemotherapy regimen. A regimen is defined by a set of drugs, each with an administration schedule of potentially varied dosages throughout a chemotherapy cycle. We characterize a regimen by drug indicators and each drug's average daily dose and maximum instantaneous dose in the cycle:

$$\begin{aligned}
\boldsymbol{x}_b^d &= \mathbb{I}(\text{drug } d \text{ is administered}),\\
\boldsymbol{x}_a^d &= \text{average daily dose of drug } d,\\
\boldsymbol{x}_i^d &= \text{maximum instantaneous dose of drug } d.
\end{aligned}$$

This allows us to differentiate between low-intensity, high-frequency and high-intensity, low-frequency dosing strategies. The outcomes of interest ($\boldsymbol{y}$) consist of overall survival, to be included as the objective (y_{OS}), and various toxicities, to be included as constraints ($y_i,\ i \in \mathcal{Y}_C$).

To determine the optimal chemotherapy regimen $\boldsymbol{x}$ for a new study cohort with characteristics $\boldsymbol{w}$, we formulate the following MIO:

$$\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{y}}\ & y_{OS} & & \\
\text{s.t. } & y_i \le \tau_i, & & i \in \mathcal{Y}_C,\\
& y_i = \hat{h}_i(\boldsymbol{x}, \boldsymbol{w}), & & i \in \mathcal{Y}_C,\\
& y_{OS} = \hat{h}_{OS}(\boldsymbol{x}, \boldsymbol{w}), & & \\
& \sum_d \boldsymbol{x}_b^d \le 3, & & \\
& \boldsymbol{x}_b \in \{0,1\}^d, & & \\
& \boldsymbol{x} \in \mathcal{X}(\boldsymbol{w}). & &
\end{aligned}$$

In this case study, we learn the full objective. However, this model could easily incorporate deterministic components to optimize as additional weighted terms in the objective. We include one domain-driven constraint, enforcing a maximum regimen combination of three drugs.

The trust region, $\mathcal{X}(\boldsymbol{w})$, plays two crucial roles in the formulation. First, it ensures that the predictive models are applied within their valid bounds and not inappropriately extrapolated. It also naturally enforces a notion of "clinically reasonable" treatments. It prevents drugs from being prescribed at doses outside of previously observed bounds, and it requires that the drug combination must have been previously seen (although potentially in different doses). It is nontrivial to explicitly characterize what constitutes a realistic treatment, and the convex hull provides a data-driven solution that integrates directly into the model framework. Furthermore, the convex hull implicitly enforces logical constraints between the different dimensions of $\boldsymbol{x}$. For example, a drug's average and instantaneous dose must be 0, if the drug's binary in-

dicator is set to 0: this does not need to be explicitly included as a constraint, since this is true for all observed treatment regimens. The only explicit constraint required here is that the indicator variables x_b are binary.

3.5.2 Dataset

Our data consists of 495 clinical trial arms from 1979-2012 (Bertsimas et al. 2016b). We consider nine contextual variables, including the average patient age and breakdown of primary cancer site. There are 28 unique drugs that appear in multiple arms of the training set, yielding 84 decision variables. We include several "dose-limiting toxicities" (DLTs) for our constraint set: Grade 3/4 constitutional toxicity, gastrointestinal toxicity, and infection, as well as Grade 4 blood toxicity. As the name suggests, these are chemotherapy side effects that are severe enough to affect the course of treatment. We also consider incidence of any dose-limiting toxicity ("Any DLT"), which aggregates over a superset of these DLTs.

We apply a temporal split, training the predictive models on trial arms through 2008 and generating prescriptions for the trial arms in 2009-2012. The final training set consists of 320 observations, and the final testing set consists of 96 observations. The full feature set, inclusion criteria, and data processing details are included in Appendix D.1.

To define the trust region, we take the convex hull of the treatment variables ($\boldsymbol{x}$) on the training set. This aligns with the temporal split setting, in which we are generating prescriptions going forward based on an existing set of past treatment decisions. In general it is preferable to define the convex hull with respect to both $\boldsymbol{x}$ and $\boldsymbol{w}$ as discussed in Appendix B.1, but this does not apply well with a temporal split. Our data includes the study year as a feature to incorporate temporal effects, and so our test set observations will definitionally fall outside of the convex hull defined by the observed $(\boldsymbol{x}, \boldsymbol{w})$ in our training set.

3.5.3 Predictive models

Several ML models are trained for each outcome of interest using cross-validation for parameter tuning, and the best model is selected based on the validation criterion. We employ function learning for all toxicities, directly predicting the toxicity incidence and applying an upper bound threshold within the optimization model.

Based on the model selection procedure, overall DLT, gastrointestinal toxicity, and overall survival are predicted using GBM models. Blood toxicity and infection are predicted using linear models, and constitutional toxicity is predicted with a RF model. This demonstrates the advantage of learning with multiple model classes; no single

method dominates in predictive performance. A complete comparison of the considered models is included in Appendix D.2.

3.5.4 Evaluation framework

We generate prescriptions using the optimization model outlined in Section 3.5.1, with the embedded model choices specified in Section 3.5.3. In order to evaluate the quality of our prescriptions, we must estimate the outcomes under various treatment alternatives. This evaluation task is notoriously challenging due to the lack of counterfactuals. In particular, we only know the true outcomes for observed cohort-treatment pairs and do not have information on potential unobserved combinations. We propose an evaluation scheme that leverages a "ground truth" ensemble (GT ensemble). We train several ML models using all data from the study. These models are not embedded in an MIO model, so we are able to consider a broader set of methods in the ensemble. We then predict each outcome by averaging across all models in the ensemble. This approach allows us to capture the maximal knowledge scenario. Furthermore, such a "consensus" approach of combining ML models has been shown to improve predictive performance and is more robust to individual model error (Bertsimas et al. 2021). The full details of the ensemble models and their predictive performances are included in Appendix D.3.

3.5.5 Optimization results

We evaluate our model in multiple ways. We first consider the performance of our prescriptions against observed (given) treatments. We then explore the impact of learning multiple sub-constraints rather than a single aggregate toxicity constraint. All optimization models have the following shared parameters: toxicity upper bound of 0.6 quantile (as observed in training data) and maximum violation of 25% for RF models. We report results for all test set observations with a feasible solution. It is possible that an observation has no feasible solution, implying that there is not a suitable drug combination lying within the convex hull for this cohort based on the toxicity requirements. These cases could be further investigated through a sensitivity analysis by relaxing the toxicity constraints or enlarging the trust region. With clinical guidance, one could evaluate the modifications required to make the solution feasible and the clinical appropriateness of such relaxations.

Table 3.6 reports the predicted outcomes under two constraint approaches: (1) constraining each toxicity separately ("All Constraints"), and (2) constraining a single aggregate toxicity measure ("DLT Only"). For each cohort in the test set, we generate predictions for all outcomes of interest under both prescription schemes and compute the relative change of our prescribed outcome from the given outcome predictions.

Benefit of prescriptive scheme. We begin by evaluating our proposed prescriptive scheme ("All Constraints") against the observed actual treatments. For example, under the GT ensemble scheme, 84.7% of cohorts satisfied the overall DLT constraint under the given treatment, compared to 94.1% under the proposed treatment. This yields an improvement of 11.10%. We obtain a significant improvement in survival (11.40%) while also improving toxicity limit satisfaction across all individual toxicities. Using the GT ensemble, we see toxicity satisfaction improvements between 1.3%-25.0%. We note that since toxicity violations are reported using the average incidence for each cohort, and the constraint limits are toxicity-specific, it is possible for a single DLT's incidence to be over the allowable limit while the overall "Any DLT" rate is not.

Table 3.6. Comparison of outcomes under given treatment regimen, regimen prescribed when only constraining the aggregate toxicity, and regimen prescribed under our full model.

		All Constraints		DLT Only	
	Given (SD)	**Prescribed (SD)**	**% Change**	**Prescribed (SD)**	**% Change**
Any DLT	0.847 (0.362)	0.941 (0.237)	11.10%	0.906 (0.294)	6.90%
Blood	0.812 (0.393)	0.824 (0.383)	1.40%	0.706 (0.458)	-13.00%
Constitutional	0.953 (0.213)	1.000 (0.000)	4.90%	1.000 (0.000)	4.90%
Infection	0.882 (0.324)	0.894 (0.310)	1.30%	0.800 (0.402)	-9.30%
Gastrointestinal	0.800 (0.402)	1.000 (0.000)	25.00%	1.000 (0.000)	25.00%
Overall Survival	10.855 (1.939)	12.092 (1.470)	11.40%	12.468 (1.430)	14.90%

We report the mean and standard deviation (SD) of constraint satisfaction (binary indicator) and overall survival (months) across the test set. The relative change is reported against the given treatment.

Benefit of multiple constraints. Table 3.6 also illustrates the value of enforcing constraints on each individual toxicity rather than as a single measure. When only constraining the aggregate toxicity measure ("DLT Only"), the resultant prescriptions actually have lower constraint satisfaction for blood toxicity and infection than the baseline given regimens. By constraining multiple measures, we are able to improve across all individual toxicities. The fully constrained model actually improves the overall DLT measure satisfaction, suggesting that the inclusion of these "sub-constraints" also makes the aggregate constraint more robust. This improvement does come at the expense of slightly lower survival between the "All" and "DLT Only" models (-0.38 months) but we note that incurring the individual toxicities that are violated in the "DLT Only" model would likely make the treatment unviable.

3.6 Discussion

Our experimental results illustrate the benefits of our constraint learning framework in data-driven decision making in two problem settings: food basket recommendations for the WFP and chemotherapy regimens for advanced gastric cancer. The quantitative results show an improvement in predictive performance when incorporating the trust region and learning from multiple candidate model classes. Our framework scales to large problem sizes, enabled by efficient formulations and tailored approaches to specific problem structures. Our approach for efficiently learning the trust region also has broad applicability in one-class constraint learning.

The nominal problem formulation is strengthened by embedding multiple models for a single constraint rather than relying on a single learned function. This notion of robustness is particularly important in the context of learning constraints: whereas misspecfications in learned objective functions can lead to suboptimal outcomes, a misspecified constraint can lead to infeasible solutions. Finally, our software exposes the model ensemble construction and trust region enlargement options directly through user-specified parameters. This allows an end user to directly evaluate tradeoffs in objective value and constraint satisfaction, as the problem's real-world context often shapes the level of desired conservatism.

We recognize several opportunities to further extend this framework. Our work naturally relates to the causal inference literature and individual treatment effect estimation (Athey and Imbens 2016, Shalit et al. 2017). These methods do not directly translate to our problem setting; existing work generally assumes highly structured treatment alternatives (*e.g.*, binary treatment vs. control) or a single continuous treatment (*e.g.*, dosing), whereas we allow more general decision structures. In future work, we are interested in incorporating ideas from causal inference to relax the assumption of unobserved confounders.

Additionally, our framework is dependent on the quality of the underlying predictive models. We constrain and optimize point predictions from our embedded models. This can be problematic in the case of model misspecification, a known shortcoming of "predict-then-optimize" methods (Elmachtoub and Grigas 2022). We mitigate this concern in two ways. First, our model selection procedure allows us to obtain higher quality predictive models by capturing several possible functional relationships. Second, our model-wrapper approach for embedding a single constraint with an ensemble of models allows us to directly control our robustness to the predictions of individual learners. In future work, there is an opportunity to incorporate ideas from robust optimization to directly account for prediction uncertainty in individual model classes. While this has been addressed in the linear case (Goldfarb and Iyengar 2003), it remains an open area of research in more general ML methods.

In this work, we present a unified framework for optimization with learned constraints that leverages both ML and MIO for data-driven decision making. Our work flexibly learns problem constraints and objectives with supervised learning, and incorporates them into a larger optimization problem of interest. We also learn the trust region, providing more credible recommendations and improving predictive performance, and accomplish this efficiently using column generation and unsupervised learning. The generality of our method allows us to tackle quite complex decision settings, such as chemotherapy optimization, but also includes tailored approaches for more efficiently solving specific problem types. Finally, we implement this as a Python software package (`OptiCL`) to enable practitioner use. We envision that `OptiCL`'s methodology will be added to state-of-the-art optimization modeling software packages.

APPENDIX

A Machine learning model embedding

A.1 Linear models

Linear Regression. Linear regression (LR) is a natural choice of predictive function given its inherent linearity and ease of embedding. A regression model can be trained to predict the outcome of interest, y, as a function of $\boldsymbol{x}$ and $\boldsymbol{w}$. The algorithm can optionally use regularization; the embedding only requires the final coefficient vectors $\boldsymbol{\beta_x} \in \mathbb{R}^n$ and $\boldsymbol{\beta_w} \in \mathbb{R}^p$ (and intercept term β_0) to describe the model. The model can then be embedded as

$$y = \beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w}.$$

Support Vector Machines. A support vector machine (SVM) uses a hyper-plane split to generate predictions, both for classification (Cortes and Vapnik 1995) and regression (Drucker et al. 1997). We consider the case of *linear* SVMs, since this allows us to obtain the prediction as a linear function of the decision variables $\boldsymbol{x}$. In linear support vector regression (SVR), which we use for function learning, we fit a linear function to the data. The setting is similar to linear regression, but the loss function only penalizes residuals greater than an ϵ threshold (Drucker et al. 1997). As with linear regression, the trained model returns a linear function with coefficients $\boldsymbol{\beta_x}$, $\boldsymbol{\beta_w}$, and β_0. The final prediction is

$$y = \beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w}.$$

For the classification setting, linear support vector classification (SVC) identifies a hyper-plane that best separates positive and negative samples (Cortes and Vapnik 1995). A trained SVC model similarly returns coefficients $\boldsymbol{\beta_x}$, $\boldsymbol{\beta_w}$, and β_0, where a sample's prediction is given by

$$y = \begin{cases} 1, & \text{if } \beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

In SVC, the output variable y is binary rather than a probability. In this case, the constraint can simply be embedded as $\beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq 0$.

A.2 Decision trees

Consider the leaves in Figure 3.2. An observation will be assigned to the leftmost leaf (node 3) if $A_1^\top \boldsymbol{x} \leq b_1$ and $A_2^\top \boldsymbol{x} \leq b_2$. An observation would be assigned to node 4 if $A_1^\top \boldsymbol{x} \leq b_1$ and $A_2^\top \boldsymbol{x} > b_2$, or equivalently, $-A_2^\top \boldsymbol{x} < -b_2$. Furthermore, we can remove the strict inequalities using a sufficiently small ϵ parameter, so that $-A_2^\top \boldsymbol{x} \leq -b_2 - \epsilon$. We can then encode the leaf assignment of observation $\boldsymbol{x}$ through the following constraints:

$$A_1^\top \boldsymbol{x} - M(1 - l_3) \leq b_1, \tag{7a}$$
$$A_2^\top \boldsymbol{x} - M(1 - l_3) \leq b_2, \tag{7b}$$
$$A_1^\top \boldsymbol{x} - M(1 - l_4) \leq b_1, \tag{7c}$$
$$-A_2^\top \boldsymbol{x} - M(1 - l_4) \leq -b_2 - \epsilon, \tag{7d}$$
$$-A_1^\top \boldsymbol{x} - M(1 - l_6) \leq -b_1 - \epsilon, \tag{7e}$$
$$A_5^\top \boldsymbol{x} - M(1 - l_6) \leq b_5, \tag{7f}$$
$$-A_1^\top \boldsymbol{x} - M(1 - l_7) \leq -b_1 - \epsilon, \tag{7g}$$
$$-A_5^\top \boldsymbol{x} - M(1 - l_7) \leq -b_5 - \epsilon, \tag{7h}$$
$$l_3 + l_4 + l_6 + l_7 = 1, \tag{7i}$$
$$y - (p_3 l_3 + p_4 l_4 + p_6 l_6 + p_7 l_7) = 0, \tag{7j}$$

where l_3, l_4, l_6, l_7 are binary variables associated with the corresponding leaves. For a given $\boldsymbol{x}$, if $A_1^\top \boldsymbol{x} \leq b_1$, Constraints (7e) and (7h) will force l_6 and l_7 to zero, respectively. If $A_2^\top \boldsymbol{x} \leq b_2$, constraint (7d) will force l_4 to 0. The assignment constraint (7i) will then force $l_3 = 1$, assigning the observation to leaf 3 as desired. Finally, constraint (7j) sets y to the prediction of the assigned leaf (p_3). We can then constrain the value of y using our desired upper bound of τ (or lower bound, without loss of generality).

More generally, consider a decision tree $\hat{h}(\boldsymbol{x}, \boldsymbol{w})$ with a set of leaf nodes $\mathcal{L}$ each described by a binary variable l_i and a prediction score p_i. Splits take the form $(A_x)^\top \boldsymbol{x} + (A_w)^\top \boldsymbol{w} \leq b$, where A_x gives the coefficients for the optimization variables $\boldsymbol{x}$ and A_w gives the coefficients for the non-optimization (fixed) variables $\boldsymbol{w}$. Let $\mathcal{S}^l$ be the set of nodes that define the splits that observations in leaf i must obey. Without loss of generality, we can write these all as $(\bar{A}_x)_j^\top \boldsymbol{x} + (\bar{A}_w)_j^\top \boldsymbol{w} - M(1 - l_i) \leq \bar{b}_j$, where $\bar{A}$ is A if leaf i follows the left split of j and $-A$ otherwise. Similarly, $\bar{b}$ equals b if the leaf falls to the left split, and $-b - \epsilon$ otherwise, as established above. This decision tree can then be embedded through the following constraints:

$$(\bar{A}_x)_j^\top \boldsymbol{x} + (\bar{A}_w)_j^\top \boldsymbol{w} - M(1 - l_i) \leq \bar{b}_j,\ i \in \mathcal{L}, j \in \mathcal{S}^l, \tag{8a}$$
$$\sum_{i \in \mathcal{L}} l_i = 1, \tag{8b}$$

$$y - \sum_{i \in \mathcal{L}} p_i l_i = 0. \tag{8c}$$

Here, M can be selected for each split by considering the maximum difference between $(\bar{A}_x)_j^\top \boldsymbol{x} + (\bar{A}_w)_j^\top \boldsymbol{w}$ and b_j. A prescription solution $\boldsymbol{x}$ for a patient with features $\boldsymbol{w}$ must obey the constraints determined by its split path, *i.e.* only the splits that lead to its assigned leaf i. If $l_i = 0$ for some leaf i, the corresponding split constraints need not be considered. If $l_i = 1$, constraint (8a) will enforce that the solution obeys all split constraints leading to leaf i. If $l_i = 0$, no constraints related to leaf i should be applied. When $l_i = 0$, constraint (8a) will be nonbinding at node j if $M \geq (\bar{A}_x)_j^\top \boldsymbol{x} + (\bar{A}_w)_j^\top \boldsymbol{w} - \bar{b}_j$. Thus we can find the minimum necessary value of M by maximizing these expressions over all possible values of $\boldsymbol{x}$ (for the patient's fixed $\boldsymbol{w}$). For a given patient with features $\boldsymbol{w}$ for whom we wish to optimize treatment, EM($\boldsymbol{w}$) is the solution of

$$\max_{\boldsymbol{x}} (\bar{A}_x)_j^\top \boldsymbol{x} + (\bar{A}_w)_j^\top \boldsymbol{w} - \bar{b}_j \tag{9a}$$

$$\text{s.t. } \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w}) \leq 0, \tag{9b}$$

$$\boldsymbol{x} \in \mathcal{X}(\boldsymbol{w}). \tag{9c}$$

Note that the non-learned constraints on $\boldsymbol{x}$, namely constraint (9b), and the trust region constraint (9c) allow us to reduce the search space when determining M.

MIO vs. LO formulation for decision trees. In Section 3.2, we proposed two ways of embedding a decision tree as a constraint. The first uses an LO to represent each feasible leaf node in the tree, while the second directly uses the entire MIO representation of the tree as a constraint. To compare the performance of these two approaches, we learn the palatability constraint using decision trees (CART) grown to have various numbers of leaves, and solve the optimization model with both approaches.

When comparing the solution times (averaged over 10 runs), Figure 9 shows that the MIO approach is relatively consistent in terms of solution time regardless of the number of leaves. With the LO approach however, as the number of leaves grows, the number of LOs to be solved also grows. While the solution time of a single LO is very low, solving multiple LOs sequentially might be heavily time consuming. A way to speed up the process is to solve the LOs in parallel. When only one LO needs to be solved, it takes 1.8 seconds in this problem setting. By parallelizing the solution of the LOs, the total solution time can be expected to take only as long as it takes for the slowest LO to be solved.

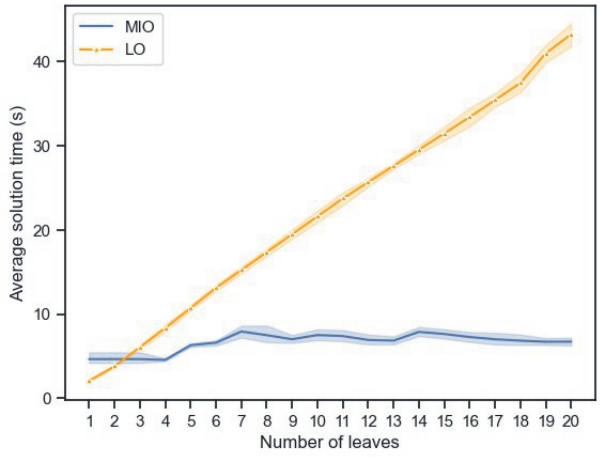


Figure 9. Comparison of MIO and multiple LO approach to tree representation, as a function of the number of leaves.

A.3 Multi-layer perceptrons

MLPs consist of an input layer, $L-2$ hidden layer(s), and an output layer. In a given hidden layer l of the network, with nodes N^l, the value of a node $i \in N^l$, denoted as v_i^l, is calculated using the weighted sum of the previous layer's node values, followed by the ReLU activation function, $\text{ReLU}(x) = \max\{0, x\}$. The value is given as

$$v_i^l = \max\left\{0, \beta_{i0}^l + \sum_{j \in N^{l-1}} \beta_{ij}^l v_j^{l-1}\right\},$$

where $\boldsymbol{\beta}_i^l$ is the coefficient vector for node i in layer l.

The ReLU operator can be encoded using linear constraints:

$$v \geq x, \tag{10a}$$
$$v \leq x - M_L(1-z), \tag{10b}$$
$$v \leq M_U z, \tag{10c}$$
$$v \geq 0, \tag{10d}$$
$$z \in \{0,1\}, \tag{10e}$$

where $M_L < 0$ is a lower bound on all possible values of x, and $M_U > 0$ is an upper bound. While this embedding relies on a big-M formulation, it can be improved in multiple ways. The model can be tightened by careful selection of M_L and M_U. Furthermore, Anderson et al. (2020) recently proposed an additional iterative cut gener-

ation procedure to improve the strength of the basic big-M formulation.

The constraints for an MLP network can be generated recursively starting from the input layer, with a set of ReLU constraints for each node in each internal layer, $l \in \{2, \ldots, L-1\}$. This allows us to embed a trained MLP with an arbitrary number of hidden layers and nodes into an MIO.

Regression. In a regression setting, the output layer L consists of a single node that is a linear combination of the node values in layer $L-1$, so it can be encoded directly as

$$y = v^L = \beta_0^L + \sum_{j \in N^{L-1}} \beta_j^L v_j^{L-1}.$$

Binary Classification. In the binary classification setting, the output layer requires one neuron with a sigmoid activation function, $S(x) = \frac{1}{1+e^{-x}}$. The value is given as

$$v^L = \frac{1}{1 + e^{-(\beta_0^L + \boldsymbol{\beta}^{L\top} \boldsymbol{v}^{L-1})}}$$

with $v^L \in (0, 1)$. This function is nonlinear, and thus, cannot be directly embedded into our formulation. However, if τ is our desired probability lower bound, it will be satisfied when $\beta_0^L + \boldsymbol{\beta}^{L\top} \boldsymbol{v}^{L-1} \geq \ln\left(\frac{\tau}{1-\tau}\right)$. Therefore, the neural network's output, binarized with a threshold of τ, is given by

$$y = \begin{cases} 1, & \text{if } \beta_0^L + \boldsymbol{\beta}^{L\top} \boldsymbol{v}^{L-1} \geq \ln\left(\dfrac{\tau}{1-\tau}\right); \\ 0, & \text{otherwise.} \end{cases}$$

For example, at a threshold of $\tau = 0.5$, the predicted value is 1 when $\beta_0^L + \boldsymbol{\beta}^{L\top} \boldsymbol{v}^{L-1} \geq 0$. Here, τ can be chosen according to the minimum necessary probability to predict 1. As for the SVC case, y is binary and the constraint can be embedded as $y \geq 1$. We refer to Appendix A.3 for the case of neural networks trained for multi-class classification.

Multi-class classification. In multi-class classification, the outputs are traditionally obtained by applying a *softmax* activation function, $S(\boldsymbol{x})_i = e^{x_i} / \left(\sum_{k=1}^K e^{x_k}\right)$, to the final layer. This function ensures that the outputs sum to one and can thus be interpreted as probabilities. In particular, suppose we have a K-class classification problem. Each node in the final layer has an associated weight vector $\boldsymbol{\beta}_i$, which maps the nodes of layer $L-1$ to the output layer by $\boldsymbol{\beta}_i^\top \boldsymbol{v}^{L-1}$. The softmax function rescales

these values, so that class i will be assigned probability

$$v_i^L = \frac{e^{\boldsymbol{\beta_i}^\top \boldsymbol{v}^{L-1}}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta_k}^\top \boldsymbol{v}^{L-1}}}.$$

We cannot apply the softmax function directly in an MIO framework with linear constraints. Instead, we use an *argmax* function to directly return an indicator of the highest probability class, similar to the approach with SVC and binary classification MLP. In other words, the output $\boldsymbol{y}$ is the identity vector with $y_i = 1$ for the most likely class. Class i has the highest probability if and only if

$$\beta_{i0}^L + \boldsymbol{\beta}_i^{L\top} \boldsymbol{v}^{L-1} \geq \beta_{k0}^L + \boldsymbol{\beta}_k^{L\top} \boldsymbol{v}^{L-1}, \quad k = 1, \ldots, K.$$

We can constrain this with a big-M constraint as follows:

$$\beta_{i0}^L + \boldsymbol{\beta}_i^{L\top} \boldsymbol{v}^{L-1} \geq \beta_{k0}^L + \boldsymbol{\beta}_k^{L\top} \boldsymbol{v}^{L-1} - M(1 - y_i), \quad k = 1, \ldots, K, \tag{11a}$$

$$\sum_{k=1}^{K} y_k = 1. \tag{11b}$$

Constraint (11a) forces $y_i = 0$, if the constraint is not satisfied for some $k \in \{1, \ldots, K\}$. Constraint (11b) ensures that $y_i = 1$ for the highest likelihood class. We can then constrain the prediction to fall in our desired class i by enforcing $y_i = 1$.

A.4 Model-wrapper approach

As discussed in Section 3.3.1, we can embed a set of models, rather than a single model, to improve the robustness of constraint satisfaction. This ensemble of P estimators can be obtained through multiple approaches, such as through bootstrapped estimators within a single model class (*e.g.*, P linear models or P decision trees) or by combining estimators across a range of model types (*e.g.*, one linear model, one decision tree, and so on). Given the set of P estimators, we then constrain that at most α proportion of the estimators violate the desired constraint. The α parameter then allows us to control the degree of conservativeness of our solution, with higher α values resulting in more permissive solutions and lower α resulting in more stringent constraint requirements. In order to constrain the violation proportion, we need indicator variables to indicate whether each of the P estimators satisfies the constraint. We use a big-M formulation to obtain these indicators $z_i \; \forall i = 1, \ldots, P$, as outlined in Section 3.3.1. However, this does increase the complexity of the master problem through the introduction of additional binary variables. There are two special cases of the violation limit that circumvent the need for a big-M formulation:

- **No allowable violation:** We can enforce a violation limit of $\alpha = 0\%$, effectively

the most conservative "worst case violation" approach.

- **Average constraint:** Rather than constraining a certain proportion of estimators to obey the constraint, we can enforce that the *average* prediction of all estimators obeys the constraint. This avoids the need for tracking individual constraint satisfaction for each estimator.

In general, we note that the embedded models can be highly nonconvex on their own (*e.g.*, if using an ensemble model as the base estimator, such as Random Forests). Thus, the additional constraints to identify and constrain violating models in this model-wrapper approach are not the primary complexity drivers in the master problem, rather complexity is driven by the individual estimators. The experiments in Appendix C.3 further investigate this latter issue: we explore runtime as the number of estimators (P) increases, the incremental benefit of increasing the number of estimators, and the impact of early stopping conditions.

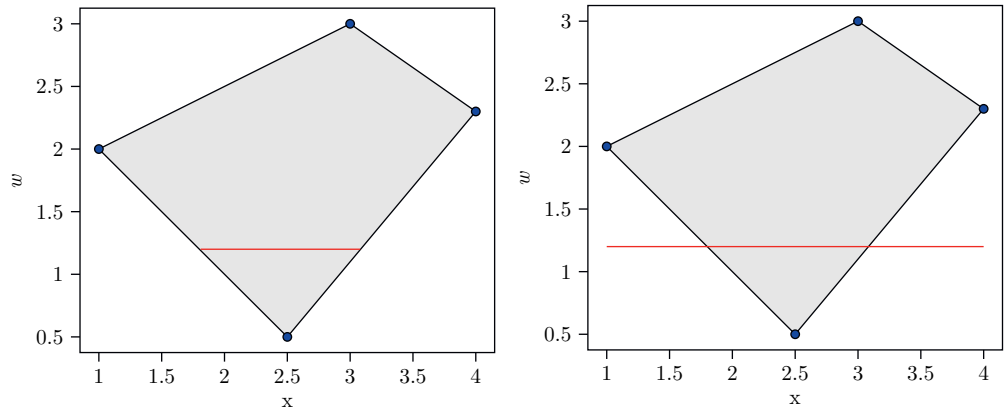
B Trust region

As we explain in Section 3.2.3, the trust region prevents the predictive models from extrapolating. It is defined as the convex hull of the set $\mathcal{Z} = \{(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{w}}_i)\}_{i=1}^{N}$, with $\bar{\boldsymbol{x}}_i \in \mathbb{R}^n$ observed treatment decisions, and $\bar{\boldsymbol{w}}_i \in \mathbb{R}^p$ contextual information. In Section B.1, we explain the importance of using both $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{w}}$ in the formulation of the convex hull. When the number of samples (N) is too large, the optimization model trust region constraints may become computationally expensive. In this case, we propose a column selection algorithm which is detailed in Section B.2.

B.1 Defining the convex hull

We characterize the feasible decision space using the convex hull of our observed data. In general, we recommend defining the feasible region with respect to both $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{w}}$. This ensures that our prescriptions are reasonable with respect to the contextual variables as well. Note that for different values of $\boldsymbol{w}$, the convex hull in the $\boldsymbol{x}$ space may be different. In Figure 10, the shaded region represents the convex hull of $\mathcal{Z}$ formed by the dataset (blue dots), and the red line represents the set of trusted solutions when $\boldsymbol{w}$ is fixed to a certain value. In Figure 10a, we see that the set of trusted solutions (red line) lies within CH($\mathcal{Z}$) when we include $\bar{\boldsymbol{w}}$. If we leave out $\bar{\boldsymbol{w}}$ in the definition of the trust region, then we end up with the undesired situation shown in Figure 10b, where the solution may lie outside of CH($\mathcal{Z}$). We observe that in some cases we must define the convex hull with a subset of variables. This is true in cases where the convex hull constraint leads to excessive data thinning, in which case it may be necessary to define the convex hull on treatment variables only.

Figure 10. Effect of $\bar{w}$ on the trust region.



(a) Solutions lie within trust region.

(b) Solutions may lie outside the trust region.

B.2 Column selection

In this section, we propose a column selection method to deal with a huge set of data points. When objectives and constraints are linear, our method reduces to the Dantzig-Wolfe decomposition method (Dantzig and Wolfe 1960). However, in our case, we do not have to solve the dual problem, since we can just enumerate all the data points. In case the functions f and g in formulation (3.1) are nonlinear and convex the Dantzig-Wolfe method cannot be used. The key point in our approach is the choice of the dual problem. Although the use of Fenchel duality seems a logical way to deal with our problem, it appears that Wolfe duality, which in general leads to nonconvex formulations, is exactly what we need. In (Stoer and Botkin 2005) and (Stoer et al. 2007) another method is described to optimize over the convex hull of a huge set of points. However, the method proposed in these papers is only suitable for problems that have only the convex hull constraint and no additional constraints.

Let $\mathrm{P}_{\mathcal{I}}$ be a convex and continuously differentiable model consisting of an objective function and constraints that may be known a priori as well as learned from data. Like in Section 3.2.3, we denote the index set of samples by $\mathcal{I}$. As part of the constraints, the trust region is defined on the entire set $\mathcal{Z}$. We start with the matrix $\boldsymbol{Z} \in \mathbb{R}^{N\times(n+p)}$, where each row corresponds to a given data point in $\mathcal{Z}$. Then, model $\mathrm{P}_{\mathcal{I}}$ is given as

$$\min_{\boldsymbol{\lambda}} f(\boldsymbol{Z}^\top \boldsymbol{\lambda}) \tag{12a}$$

$$\text{s.t. } g_j(\boldsymbol{Z}^\top \boldsymbol{\lambda}) \leq 0, \quad j = 1, \ldots, m, \qquad \perp \boldsymbol{\mu}, \tag{12b}$$

$$\sum_{i\in\mathcal{I}} \lambda_i = 1, \qquad \perp \rho, \tag{12c}$$

$$\lambda_i \geq 0, \quad i \in \mathcal{I}, \qquad \perp \boldsymbol{v}, \tag{12d}$$

where the decision variable $\boldsymbol{x}$ is replaced by $\boldsymbol{Z}^\top \boldsymbol{\lambda}$. Constraints (12b) include both *known* and *learned* constraints, while constraints (12c) and (12d) are used for the trust region. The dual variables associated with with constraints (12b), (12c), and (12d) are $\boldsymbol{\mu} \in \mathbb{R}^m, \rho \in \mathbb{R}$, and $\boldsymbol{v} \in \mathbb{R}^N$, respectively. Note that for readability, we omit the contextual variables ($\boldsymbol{w}$) without loss of generality.

When we deal with huge datasets, solving $\mathrm{P}_\mathcal{I}$ may be computationally expensive. Therefore, we propose an iterative column selection algorithm (Algorithm 1) that can be used to speed up the optimization while still obtaining a global optima.

Algorithm 1 Column Selection

Require: $\mathcal{I}$ ▷ Index set of columns of $\boldsymbol{Z}^\top$
Ensure: $\boldsymbol{\lambda}^*$ ▷ Optimal solution
1: $\mathcal{I}' \leftarrow \mathcal{I}^0$ ▷ Initial column pool
2: **while** TRUE **do**
3: $\quad \boldsymbol{\lambda}^*, (\boldsymbol{\mu}^*, \rho^*, \boldsymbol{v}^*) \leftarrow \mathrm{P}_{\mathcal{I}'}$
4: $\quad \bar{\mathcal{I}} \leftarrow$ WolfeDual$(\boldsymbol{\lambda}^*, (\boldsymbol{\mu}^*, \rho^*, \boldsymbol{v}^*), \mathcal{I}', \mathcal{I})$ ▷ Column(s) selection
5: $\quad$ **if** $\bar{\mathcal{I}} \neq \emptyset$ **then**
6: $\quad\quad \mathcal{I}' \leftarrow \mathcal{I}' \cup \bar{\mathcal{I}}$
7: $\quad$ **else**
8: $\quad\quad$ Break
9: $\quad$ **end if**
10: **end while**

The algorithm starts by initializing $\mathcal{I}' \subseteq \mathcal{I}$ with an arbitrarily small subset of samples $\mathcal{I}^0$ and iteratively solves the restricted master problem $\mathrm{P}_{\mathcal{I}'}$ and the WolfeDual function. By solving $\mathrm{P}_{\mathcal{I}'}$, we get the primal and dual optimal solutions $\boldsymbol{\lambda}^*$ and $(\boldsymbol{\mu}^*, \rho^*, \boldsymbol{v}^*)$, respectively. The primal and dual optimal solutions, together with $\mathcal{I}$ and $\mathcal{I}'$, are given as input to WolfeDual which returns a set of samples $\bar{\mathcal{I}} \subseteq \mathcal{I} \setminus \mathcal{I}'$ with negative reduced cost. If $\bar{\mathcal{I}}$ is not empty it is added to $\mathcal{I}'$ and a new iteration starts, otherwise the algorithm stops, and $\boldsymbol{\lambda}^*$ (with the corresponding $\boldsymbol{x}^*$) is returned as the global optima of $\mathrm{P}_\mathcal{I}$. A visual interpretation of Algorithm 1 is shown in Figure 3.4.

In function WolfeDual, samples $\bar{\mathcal{I}}$ are selected using the Karush–Kuhn–Tucker (KKT) stationary condition which corresponds to the equality constraint in the Wolfe dual formulation of $\mathrm{P}_\mathcal{I}$ (Wolfe 1961). The KKT stationary condition of $\mathrm{P}_{\mathcal{I}'}$ is

$$\nabla_{\boldsymbol{\lambda}} f(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) + \sum_{i=1}^{m} \mu_i^* \nabla_{\boldsymbol{\lambda}} g_i(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda^*}) - \boldsymbol{e}\rho^* - \boldsymbol{v}^* = \boldsymbol{0}, \tag{13}$$

where $\tilde{\boldsymbol{Z}}$ is the matrix constructed with samples in $\mathcal{I}'$, and $\boldsymbol{e}$ is an N'-dimensional

vector of ones with $N' = |\mathcal{I}'|$. Equation (13) can be rewritten as

$$\tilde{\boldsymbol{Z}}\nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) + \sum_{i=1}^{m} \mu_i^* \tilde{\boldsymbol{Z}}\nabla_{\boldsymbol{x}} g_i(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) - \boldsymbol{e}\rho^* - \boldsymbol{v}^* = \boldsymbol{0}. \tag{14}$$

Equation (14) is used to evaluate the reduced cost related to each sample $\bar{\boldsymbol{z}} \in \mathcal{Z}$ which is not in matrix $\tilde{\boldsymbol{Z}}$. Consider a new sample $\bar{\boldsymbol{z}}$ in (14), with its associated $\lambda_{N'+1}$ set equal to zero. $(\lambda_1^*, \ldots, \lambda_{N'}^*, \lambda_{N'+1})$ is still a feasible solution of the restricted master problem $\mathrm{P}_{\mathcal{I}'}$, since it does not affect the value of $\boldsymbol{x}$. As a consequence, $\boldsymbol{\mu}$ and ρ will not change their value, nor will f and $\boldsymbol{g}$. The only unknown variable is $v_{N'+1}$, namely the reduced cost of $\bar{\boldsymbol{z}}$. However, we can write it as

$$\begin{pmatrix} \boldsymbol{v}^* \\ v_{N'+1} \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{Z}} \\ \bar{\boldsymbol{z}}^\top \end{pmatrix} \nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) + \sum_{i=1}^{t} \mu_i^* \begin{pmatrix} \tilde{\boldsymbol{Z}} \\ \bar{\boldsymbol{z}}^\top \end{pmatrix} \nabla_{\boldsymbol{x}} g_i(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) - \boldsymbol{e}\rho^*. \tag{15}$$

If $v_{N'+1}$ is negative it means that we may improve the incumbent solution of $\mathrm{P}_{\mathcal{I}'}$ by including the sample $\bar{\boldsymbol{z}}$ in $\tilde{\boldsymbol{Z}}$.

Lemma .1. After solving the convex and continuously differentiable problem $\mathrm{P}_{\mathcal{I}'}$, the sample in $\mathcal{I} \setminus \mathcal{I}'$ with the most negative reduced cost is a vertex of the convex hull $\mathrm{CH}(\mathcal{Z})$.

Proof. Proof From equation (15) we have

$$v_{N'+1} = \bar{\boldsymbol{z}}^\top \nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) + \bar{\boldsymbol{z}}^\top \nabla_{\boldsymbol{x}} \boldsymbol{g}(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*)\boldsymbol{\mu}^* - \rho^*. \tag{16}$$

The problem of finding $\bar{\boldsymbol{z}}$, such that its reduced cost is the most negative one, can be written as a linear program where equation (16) is being minimized, and a solution must lie within $\mathrm{CH}(\mathcal{Z})$. That is,

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{\lambda}} \; & \boldsymbol{z}^\top \nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{Z}}^\top \boldsymbol{\lambda}^*) + \boldsymbol{z}^\top \nabla_{\boldsymbol{x}} \boldsymbol{g}(\tilde{\boldsymbol{Z}})\boldsymbol{\mu}^* - \rho^* \\ \text{s.t. } & \boldsymbol{Z}^\intercal \boldsymbol{\lambda} = \boldsymbol{z}, \\ & \sum_{j \in \mathcal{I}} \lambda_j = 1, \\ & \lambda_j \geq 0, \quad j \in \mathcal{I}, \end{aligned} \tag{17}$$

where $\boldsymbol{z}$ and $\boldsymbol{\lambda}$ are the decision variables, and $\boldsymbol{\mu}^*$, $\boldsymbol{\lambda}^*$, ρ^* are fixed parameters. Since the objective function is linear with respect to $\boldsymbol{z}$, the optimal solution of (17) will necessarily be a vertex of $\mathrm{CH}(\mathcal{Z})$. □

To illustrate the benefits of column selection, consider the following convex optimiza-

tion problem that we shall refer to as P_{exp}:

$$\min_{\boldsymbol{x}} \boldsymbol{c}^\top \boldsymbol{x} \tag{18a}$$

$$\text{s.t. } log(\sum_{i=1}^{n} e^{x_i}) \leq t, \tag{18b}$$

$$\boldsymbol{A}\boldsymbol{x} \leq b, \tag{18c}$$

$$\sum_{i=1}^{N} \lambda_i \bar{\boldsymbol{z}}_i = \boldsymbol{x}, \tag{18d}$$

$$\sum_{j=1}^{N} \lambda_j = 1, \tag{18e}$$

$$\lambda_j \geq 0, \quad j = 1 \dots N. \tag{18f}$$

Without a loss of generality, we assume that the constraint (18b) is known a priori, and constraints (18c) are the linear embeddings of learned constraints with $\boldsymbol{A} \in \mathbb{R}^{k \times n}$ and $\boldsymbol{b} \in \mathbb{R}^k$. Constraints (18d-18f) define the trust region based on N datapoints. Figure 11 shows the computation time required to solve P_{exp} with different values of n, k, and N. The "No Column Selection" approach consists of solving P_{exp} using the entire dataset. The "Column Selection" approach makes use of Algorithm 1 to solve the problem, starting with $|\mathcal{I}^0| = 100$, and selecting only one sample at each iteration, *i.e.*, the one with the most negative reduced cost. It can be seen that in all cases, the use of column selection results in significantly improved computation times. This allows us to more quickly define the trust region for problems with large amounts of data.

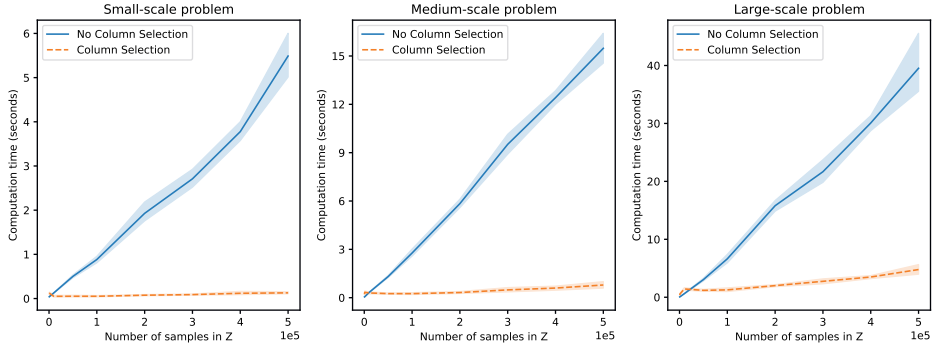


Figure 11. Effect of column selection on computation time. Solution times are reported for three different sizes of problem P_{exp}. Small-scale: $n = 5,\ k = 10$. Medium-scale: $n = 10,\ k = 50$. Large-scale: $n = 20,\ k = 100$. The number of samples goes from 500 to 5×10^5. In each iteration, the sample with most negative reduced cost is selected. The same problem is solved using MOSEK (2019) with conic reformulation for 10 different instances where c, A, and b are randomly generated.

C WFP case study

Table 7 and Table 8 show the nutritional value of each food and our assumed nutrient requirements, respectively. The values adopted are based on the World Health Organization (WHO) guidelines (UNHCR et al. 2002).

Table 7. Nutritional contents per gram for different foods.

Food	Eng(kcal)	Prot(g)	Fat(g)	Cal(mg)	Iron(mg)	VitA(ug)	ThB1(mg)	RibB2(mg)	NicB3(mg)	Fol(ug)	VitC(mg)	Iod(ug)
Beans	335	20	1.2	143	8.2	0	0.5	0.22	2.1	180	0	0
Bulgur	350	11	1.5	23	7.8	0	0.3	0.1	5.5	38	0	0
Cheese	355	22.5	28	630	0.2	120	0.03	0.45	0.2	0	0	0
Fish	305	22	24	330	2.7	0	0.4	0.3	6.5	16	0	0
Meat	220	21	15	14	4.1	0	0.2	0.23	3.2	2	0	0
Corn-soya blend	380	18	6	513	18.5	500	0.65	0.5	6.8	0	40	0
Dates	245	2	0.5	32	1.2	0	0.09	0.1	2.2	13	0	0
Dried skim milk	360	36	1	1257	1	1,500	0.42	1.55	1	50	0	0
Milk	360	36	1	912	0.5	280	0.28	1.21	0.6	37	0	0
Salt	0	0	0	0	0	0	0	0	0	0	0	1000000
Lentils	340	20	0.6	51	9	0	0.5	0.25	2.6	0	0	0
Maize	350	10	4	13	4.9	0	0.32	0.12	1.7	0	0	0
Maize meal	360	9	3.5	10	2.5	0	0.3	0.1	1.8	0	0	0
Chickpeas	335	22	1.4	130	5.2	0	0.6	0.19	3	100	0	0
Rice	360	7	0.5	7	1.2	0	0.2	0.08	2.6	11	0	0
Sorghum/millet	335	11	3	26	4.5	0	0.34	0.15	3.3	0	0	0
Soya-fortified bulgur wheat	350	17	1.5	54	4.7	0	0.25	0.13	4.2	74	0	0
Soya-fortified maize meal	390	13	1.5	178	4.8	228	0.7	0.3	3.1	0	0	0
Soya-fortified sorghum grits	360	360	1	40	2	0	0.2	0.1	1.7	50	0	0
Soya-fortified wheat flour	360	16	1.3	211	4.8	265	0.66	0.36	4.6	0	0	0
Sugar	400	0	0	0	0	0	0	0	0	0	0	0
Oil	885	0	100	0	0	0	0	0	0	0	0	0
Wheat	330	12.3	1.5	36	4	0	0.3	0.07	5	51	0	0
Wheat flour	350	11.5	1.5	29	3.7	0	0.28	0.14	4.5	0	0	0
Wheat-soya blend	370	20	6	750	20.8	498	1.5	0.6	9.1	0	40	0

Eng = Energy, Prot = Protein, Cal = Calcium, VitA = Vitamin A, ThB1 = ThiamineB1, RibB2 = RiboflavinB2, NicB3 = NicacinB3, Fol = Folate, VitC = Vitamin C, Iod = Iodine

Table 8. Nutrient requirements used in optimization model.

Type	Eng(kcal)	Prot(g)	Fat(g)	Cal(mg)	Iron(mg)	VitA(ug)	ThB1(mg)	RibB2(mg)	NicB3(mg)	Fol(ug)	VitC(mg)	Iod(ug)
Avg person day	2100	52.5	89.25	1100	22	500	0.9	1.4	12	160	0	150

Eng = Energy, Prot = Protein, Cal = Calcium, VitA = Vitamin A, ThB1 = ThiamineB1, RibB2 = RiboflavinB2, NicB3 = NicacinB3, Fol = Folate, VitC = Vitamin C, Iod = Iodine

C.1 Food baskets generation and palatability function

Referring to Peters et al. (2021), a food basket $x_k(\forall k \in \mathcal{K})$ is defined as a collection of $\mathcal{K}$ commodities, such as beans, meat, and oil, along with their respective quantities measured in grams. These commodities are classified into five macro-categories: cereals and grains, pulses and vegetables, oils and fats, mixed and blended foods, and meat and fish, as well as dairy. Each macro-category $g \in \mathcal{G}$ is associated with an upper bound (max_g) and a lower bound (min_g), as shown in Table 9. We use the notation $\mathcal{K}_g$ to indicate the set of commodities belonging to category g. In contrast to the approach used in Peters et al. (2021), where bound constraints were utilized to ensure the palatability of the food basket, we expand the notion of palatability by incorporat-

ing a palatability score that ranges from non-negative values and tends towards zero for more enjoyable diets. The score is calculated as:

$$Palatability\ Score = \sqrt{\sum_{g \in \mathcal{G}} (\gamma_g(\widehat{x}_g - Opt_g))^2}, \tag{19}$$

where

$$\widehat{x}_g = \sum_{k \in \mathcal{K}_g} x_k \text{ with } g \in \mathcal{G} \text{ and}$$

$$Opt_g = \frac{max_g + min_g}{2} \text{ with } g \in \mathcal{G}.$$

To account for the different range sizes $(max_g - min_g)$ across the macro-categories, we introduce a scaling parameter γ_g that determines their influence on the score, as presented in Table 9. The resulting score is normalized on a scale of 0 to 1, where a score of 1 represents a perfectly appetizing food basket, while a score of 0 indicates an inedible basket.

macro-category	min	max	γ
Cereals & Grains	200	600	1
Pulses & Vegetables	30	100	5.7
Oils & Fats	15	40	16
Mixed & Blended Foods	0	90	4.4
Meat & Fish & Dairy	0	60	6.6

Table 9. Macro-categories bounds and scaling factor.

The generation of diverse food baskets is done by solving several diet problems whose cost function changes at each run and enforcing constraints on the nutrient requirements as well as on the maximum number of foods belonging to the same category.

C.2 Predictive models

Table 10 shows the structure of the predictive models used in the WFP experiments. For each model, the choice of parameters is based on a cross-validation procedure.

C.3 Effect of robustness parameters

Robustness impact by algorithm. Table 11 reports the change in objective value (cost) and constrained outcome (palatbility) between the nominal and bootstrapped

Model	Parameters
Linear	ElasticNet parameters: 0.1 (alpha), 0.1 (ℓ_1-ratio)
SVM	regularization parameter: 100
CART	max depth: 10, max features: 1.0, min samples leaf: 0.02
RF	max depth : 4, max features: auto, number of estimators: 25
GBM	learning rate: 0.2, max depth: 5, number of estimators: 20
MLP	hidden layers: 1, size hidden layers: (100,) activation: relu

Table 10. Definition of the predictive model parameters used in the WFP case study

solution with 10 estimators and a violation limit of 25%. The goal of the WFP case study is to minimize cost such that palatability is at least 0.5; thus, a smaller cost and larger palatability are better. As expected, the robust solution increases both the cost and palatability of the prescribed diets. We see that the relative increase in cost is consistently lower than the relative increase in real palatability across all methods, indicating that the improvement in palatability exceeds the incremental cost addition. While the acceptable trade off between cost and palatability could differ by use case, this could be further explored with alternative violation limits. Additionally, we compare the single algorithm constraints against an ensemble of all six methods, also with a violation limit of $\alpha = 0.25$. The ensemble with multiple algorithms yields an objective value of 1313 and real palatability of 0.57. This represents a -1.8% to 1% increase in cost and 5.6% to 15.6% increase in real palatability over the nominal solutions. When compared to the bootstrapped single-method models, it is generally *more* conservative. This is consistent with the fact that it must satisfy the constraint estimate across the majority of the individual methods, forcing it to be conservative relative to this set.

	Objective value			Real palatability		
Algorithm	**Nominal**	**Bootstrapped**	**Change**	**Nominal**	**Bootstrapped**	**Change**
Linear	1337	1359	1.6%	0.496	0.512	3.1%
SVM	1306	1308	0.1%	0.541	0.548	1.2%
CART	1301	1307	0.5%	0.539	0.550	2.1%
RF	1305	1306	0.0%	0.543	0.551	1.5%
GBM	1300	1304	0.3%	0.532	0.553	3.9%
MLP	1307	1313	0.5%	0.537	0.587	9.4%

Table 11. Change in cost and palatability from nominal to bootstrapped ($P = 10, \alpha = 0.25$) solution.

Effect of number of estimators. Table 12 compares the runtime as the number of estimators (P) increases up to 25 estimators. We see that the solve time for the linear, SVM, CART, and MLP models are stable as the number of estimators increases. In

contrast, we see that the ensemble algorithms, RF and GBM, have exponential runtime increases as the number of estimators grows. RF and GBM are already comprised of multiple individual learners, so embedding multiple estimators involves adding multiple *sets* of decision trees, which becomes computationally expensive. All results are reported over 100 instances. The experiments were run using a virtual computing environment with 4 CPU and 32 GB total RAM. We also report the runtime for an ensemble of estimators obtained from different model classes ("Ensemble"), using a single model from each class.

We further investigate the runtimes with 25 estimators in Table 13. The left side of the table reports the mean, median, and maximum runtimes for each method on the same 100 experiments as above. We see that the RF and GBM models have reasonable median solve times (6.66 and 18.80 minutes, respectively), but the average solve times are driven up by outlier instances that have significantly higher runtimes (max. 2110 and 1603 minutes, respectively). We propose to use a time limit to control the experiment times. On the right side of the table, we see that using a 4 hour time limit returns optimal solutions for 95% of the RF runs and 82% of the GBM runs, and feasible solutions for all but four GBM instances. In cases where an optimal solution is obtained, the average runtime is less than 40 minutes. In cases where the time limit is hit, the average remaining MIP gap is 1.02% for RF and 5.21% for GBM. The results suggest that imposing this termination condition results in high quality solutions with a modest optimality gap.

Algorithm	**P = 2**	**P = 5**	**P = 10**	**P = 25**
Linear	0.01	0.01	0.02	0.01
SVM	0.01	0.01	0.02	0.01
CART	0.02	0.02	0.03	0.12
RF	0.15	1.34	11.93	44.87
GBM	0.37	3.58	10.71	133.13
MLP	0.01	0.02	0.04	0.48
Ensemble		0.09		

Table 12. Optimization solver runtime (minutes) as a function of number of bootstrapped estimators.

The runtime experiments raise a natural question: what is the impact of embedding a larger number of estimators? We consider the cost-palatability trade off for a decision tree model as we vary the number of estimators from $P = 2$ to $P = 50$, averaged over the candidate violation limits. The results are shown in Figure 12. As the number of estimators increases, the results tend to be more conservative. By 10 estimators, the trade off curve well-approximates the curves for higher estimator up to an inflection point where average cost increases significantly. By $P = 25$ and $P = 50$, the curves closely match, suggesting diminishing value in increasing the number of estimators

	Runtime (mins) to optimality			4 hour time limit			
Algorithm	**Mean**	**Median**	**Max**	**% feasible**	**% optimal**	**Avg. runtime to optimality (mins)**	**Avg. remaining MIP gap**
Linear	0.01	0.01	0.04	100%	100%	0.01	–
SVM	0.01	0.01	0.04	100%	100%	0.01	–
CART	0.12	0.08	0.64	100%	100%	0.12	–
RF	44.87	6.66	2109.72	100%	95%	19.76	1.02%
GBM	133.13	18.80	1603.51	96%	82%	37.79	5.21%
MLP	0.49	0.12	6.89	100%	100%	0.48	–

Table 13. Runtime results for $P = 25$ estimators, both when solved to optimality (left) and with a 4 hour time limit (right).

beyond a certain point.

Finally, Table 14 reports the parameters used in our bootstrapped models. For each method, we report the parameter grid that was used in our model training and selection procedure. Individual estimators use different combinations of these parameters based on the validation performance on the specific bootstrapped samples. We note that for these experiments, we used a default parameter grid implemented in `OptiCL`; this grid can be manually set by a user when specifying each outcome of interest before model training.

Algorithm	**Parameter Grid**
Linear	alpha': [0.1, 1, 10, 100, 1000], 'l1_ratio': np.arange(0.1, 1.0, 0.2)
SVM	'C': [.1, 1, 10, 100]
CART	max_depth': [3, 4, 5, 6, 7, 8, 9, 10], 'min_samples_leaf': [0.02, 0.04, 0.06], 'max_features': [0.4, 0.6, 0.8, 1.0]
RF	n_estimators': [10, 25], 'max_features': ['auto'], 'max_depth': [2, 3, 4]
GBM	learning_rate': [0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2], 'max_depth': [2, 3, 4, 5], 'n_estimators': [20]
MLP	'hidden_layer_sizes': [(10,), (20,), (50,), (100,)]

Table 14. Default parameter grid for supported algorithms.

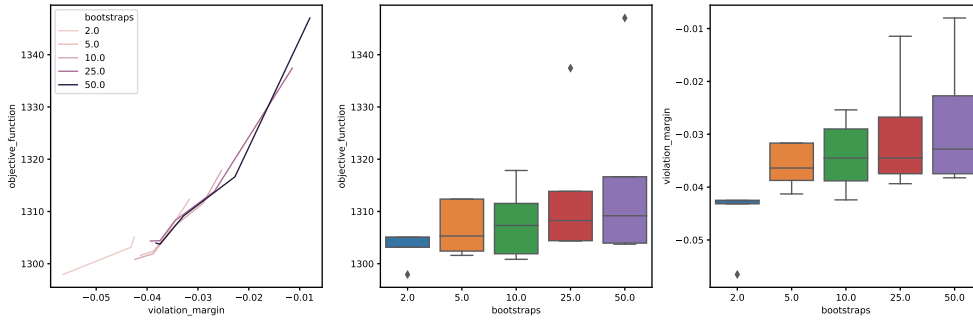


Figure 12. Effect of the number of bootstrapped estimators (P) on the cost and palatability of the prescribed diet.

D Chemotherapy regimen design

D.1 Data processing

The data for this case study includes three components, study cohort characteristics ($\boldsymbol{w}$), treatment variables ($\boldsymbol{x}$), and outcomes ($\boldsymbol{y}$). The raw data was obtained from Bertsimas et al. (2016b), in which the authors manually curated data from 495 clinical trial arms for advanced gastric cancer. Our feature space was processed as follows:

Cohort characteristics. We included several cohort characteristics to adjust for the study context: fraction of male patients, median age, primary site breakdown (Stomach vs. GEJ), fraction of patients receiving prior palliative chemotherapy, and mean ECOG score. We also included variables for the study context: the study year, country, and number of patients. Missing data was imputed using multiple imputation based on the other contextual variables; 20% of observations had one missing feature and 6% had multiple missing features.

Treatment variables. Chemotherapy regimens involve multiple drugs being delivered at potentially varied frequencies over the course of a chemotherapy cycle. As a result, multiple dimensions of the dosage must be encoded to reflect the treatment strategy. As in Bertsimas et al. (2016b), we include three variables to represent each drug: an indicator (1 if the drug is used in the regimen), instantaneous dose, and average dose.

Outcomes. We use Overall Survival (OS) as our survival metric, as reported in the clinical trials. Any observations with unreported OS are excluded. We consider sev-

eral "dose-limiting toxicities" (DLTs): Grade 3/4 constitutional, gastrointestinal, infection, and neurological toxicities, as well as Grade 4 blood toxicities. The toxicities reported in the original clinical trials are aggregated according to the CTCAE toxicity classes (Cancer Therapy Evaluation Program 2006). We also include a variable for the occurrence of any of the four individual toxicities (t_i for each toxicity $i \in T$, called DLT proportion; we treat these toxicity groups as independent and thus define the DLT proportion as

$$DLT = 1 - \prod_{i \in T} (1 - t_i).$$

We define Grade 4 blood toxicity as the maximum of five individual blood toxicities (related to neutrophils, leukocytes, lymphocytes, thrombocytes, anemia). Observations missing all of these toxicities were excluded; entries with partial missingness were imputed using multiple imputation based on other blood toxicity columns. Similarly, observations with no reported Grade 3/4 toxicities were excluded; those with partial missingness were imputed using multiple imputation based on the other toxicity columns. This exclusion criteria resulted in a final set of 461 (of 495) treatment arms.

We split the data into training/testing sets temporally. The training set consists of all clinical trials through 2008, and the testing set consists of all 2009-2012 trials. We exclude trials from the testing set if they use new drugs not seen in the training data (since we cannot evaluate these given treatments). We also identify sparse treatments (defined as being only seen once in the training set) and remove all observations that include these treatments. The final training set consists of 320 observations, and the final testing set consists of 96 observations.

D.2 Predictive models

Table 15 shows the out-of-sample performance of all considered methods in the model selection pipeline. We note that model choice is based on the 5-fold validation performance, so it does not necessarily correspond to the highest test set performance. The final parameters for each model and each outcome, selected through the cross-validation procedure, are shown in Table 16.

D.3 Prescription evaluation

Table 17 shows the performance of the models that comprise the ground truth ensemble used in the evaluation framework. These models trained on the full data. We see that the ensemble models, particularly RF and GBM, have the highest performance. These models are trained on more data and include more complex parameter options

Outcome	Linear	SVM	CART	RF	GBM
Any DLT	0.268	-0.094	-0.016	0.152	0.202
Blood	0.196	-1.102	0.012	0.153	0.105
Constitutional	0.106	0.144	0.157	0.194	0.136
Infection	0.082	-0.511	-0.222	0.070	0.035
Gastrointestinal	0.141	-0.196	-0.023	0.066	0.083
Overall Survival	0.448	0.385	0.474	0.496	0.450

Table 15. Comparison of out-of-sample R^2 all considered models for learned outcomes in chemotherapy regimen selection problem.

Outcome	Model	Parameters
Any DLT	GBM	learning rate: 0.2, max depth: 2, number of estimators: 20
Blood	Linear	ElasticNet parameters: 0.1 (alpha), 0.7 (ℓ_1-ratio)
Constitutional	RF	max depth : 4, max features: 'auto', number of estimators: 25
Infection	Linear	ElasticNet parameters: 1 (alpha), 0.5 (ℓ_1-ratio)
Gastrointestinal	GBM	learning rate: 0.1, max depth: 4, number of estimators: 20
Overall Survival	GBM	learning rate: 0.1, max depth: 3, number of estimators: 20

Table 16. Predictive model parameters used in the chemotherapy case study.

(*e.g.*, deeper trees, larger forests) since they are not required to be embedded in the MIO and are rather used directly to generate predictions. The final parameters for each model and each outcome, selected through the cross-validation procedure, are shown in Table 18. For this reason, the GT ensemble could also be generalized to consider even broader method classes that are not directly MIO-representable, such as neural networks with alternative activation functions, providing an additional degree of robustness.

D.4 Optimization runtimes

Table 19 reports the runtimes of the optimization model results presented in Section 3.5.5, Table 3.6. Results are averaged over all patients in the test set.

E Comparison with JANOS and EML

As mentioned earlier in Section 3.1.1, JANOS and EML are two software frameworks for embedding learned ML models in optimization problems. In this section, we compare the performance of `OptiCL` to those of JANOS and EML using the test problems

outcome	Linear	SVM	CART	RF	GBM	XGB
Any DLT	0.301	0.330	0.250	0.573	0.670	0.323
Blood	0.287	0.351	0.211	0.701	0.813	0.446
Constitutional	0.139	0.224	0.246	0.602	0.682	0.285
Infection	0.217	0.303	0.139	0.514	0.588	0.247
Gastrointestinal	0.201	0.328	0.238	0.563	0.733	0.475
Overall Survival	0.528	0.469	0.421	0.815	0.827	0.756

Table 17. Performance (R^2) of individual models in ground truth ensemble for model evaluation.

Table 18. Predictive model parameters used in the ground truth ensemble for model evaluation.

Algorithm	Parameter	Any DLT	Blood	Const.	Inf.	GI	OS
Linear	alpha	0.1	0.1	1	1	1	0.1
	ℓ_1 ratio	0.6	0.5	0.4	0.3	0.7	0.8
SVM	regularization parameter	100	100	1	10	100	0.1
CART	max depth	3	3	4	3	5	3
	max features	1	1	0.6	0.6	0.6	0.8
	min samples per leaf	0.04	0.06	0.06	0.06	0.06	0.02
RF	max depth	6	8	6	6	6	8
	max features	auto	auto	auto	auto	auto	auto
	number of estimators	500	500	500	250	250	250
GBM	learning rate	0.01	0.025	0.01	6	0.01	0.01
	max depth	5	5	5	auto	6	5
	number of estimators	250	250	250	250	250	250
XGB	cols sampled by tree	0.8	1	0.8	1	0.8	1
	gamma	0.5	0.5	1	1	0.5	10
	max depth	4	5	4	4	5	4
	min child weight	10	1	10	10	1	10
	number of estimators	250	250	250	250	250	250
	subsample	1	0.8	0.8	0.8	0.8	1

Const. = Constitutional, Inf. = Infection, GI = Gastrointestinal, OS = Overall survival.

Model Version	Average Time (SD)
All Constraints	0.511 (0.892)
DLT Only	0.203 (0.433)

Table 19. Average (and standard deviation) of runtimes for gastric cancer case, in seconds.

in Bergman et al. (2022) and Lombardi et al. (2017), respectively. The experiments are conducted using an Intel i7-8665U 1.9 GHz CPU, 16 GB RAM (Windows 10 environment).

E.1 OptiCL vs JANOS

In the Student Enrolment Problem (SEP) in Bergman et al. (2022), a university's admission office seeks to offer scholarships to some of the admitted students in order to bolster the class profile. The objective is to maximize the expected class size subject to budget constraints. This problem is formulated as:

$$\max \sum_{i=1}^{N} y_i \tag{20a}$$

$$\text{s.t.} \sum_{i=1}^{N} x_i \leq \textbf{BUDGET}, \tag{20b}$$

$$y_i = \hat{h}(s_i, g_i, x_i) \qquad \forall i \in \{1, \ldots, N\}, \tag{20c}$$

$$0 \leq x_i \leq 25,000 \qquad \forall i \in \{1, \ldots, N\}, \tag{20d}$$

where x_i is the decision variable indicating the amount of scholarship assigned to each student accepted, s_i is the SAT score of applicant i, and g_i is the GPA score of applicant i. The predicted outcome y_i represents the probability of a candidate i accepting the offer, and $\hat{h}$ is the fitted model used to predict any candidate's probabilities of accepting an offer. The parameters s_i, g_i, and the decision variable x_i are the predictive model's inputs. In order to compare OptiCL and JANOS, we solved the SEP for different student sizes, and compared the objective values and runtimes. Although `OptiCL` and JANOS handle neural network embedding in a similar manner, JANOS uses a parameterized discretization to handle logistic regression predictions. We therefore compared their performances only using the logistic regression models, as we expected to see a difference in performance based on the differences in implementation. In the experiments reported in Figure 13, we discretize the logistic regression (LogReg) in JANOS using three different number of intervals (reported between brackets in the

Figure legend). From the experiments, we can see that `OptiCL` achieves better objective values in all three instances. It can also be seen that for the larger problems, `OptiCL` is much more efficient in terms of optimization runtime than JANOS.

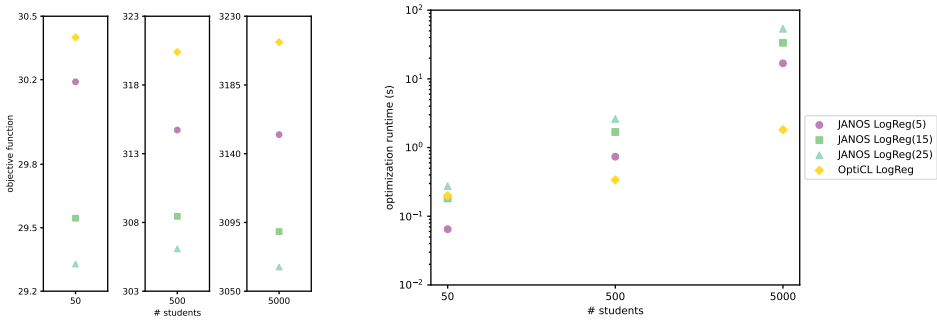


Figure 13. Objective value (right) and runtime (left) comparison between `OptiCL` and `JANOS` for the SEP.

E.2 OptiCL vs EML

In the thermal-aware Workload Dispatching Problem (WDP) in Lombardi et al. (2017), the goal is to assign jobs to the different cores on a multi-core processor. The processor has 24 dual-core tiles arranged in a 4×6 grid, resulting in an arrangement with 48 cores in an 8×6 grid. A direct comparison between `OptiCL` and EML is not possible, as Lombardi et al. (2017) do not use neural networks or decision trees for constraint learning in MIO problems. Their focus for these predictive models are Local Search, Constraint Programming, or SAT Modulo Theory problems. What we do, however, is demonstrate that `OptiCL` is able to solve the example in an MIO setting. The model considered here is the "ANN1" model in Lombardi et al. (2017) given as:

$$\max\ z \tag{21}$$

$$\text{s.t. } z \leq y_k \quad \forall k = 0, \ldots, m-1, \tag{22}$$

$$y_k = \hat{h}_k(avgcpi_k, neighcpi_k, othercpi_k) \quad \forall k = 0, \ldots, m-1, \tag{23}$$

$$\sum_{k=0}^{m-1} x_{ik} = 1 \quad \forall i = 0, \ldots, n-1, \tag{24}$$

$$\sum_{i=0}^{n-1} x_{ik} = \frac{n}{m} \quad \forall k = 0, \ldots, m-1, \tag{25}$$

$$avgcpi_k = \frac{1}{m} \sum_{i=0}^{n-1} cpi_i x_{ik} \quad \forall k = 0, \ldots, m-1, \tag{26}$$

$$neighcpi_k = \frac{1}{m} \sum_{h \in N(k)} avgcpi_h \quad \forall k = 0, \ldots, m-1, \tag{27}$$

$$othercpi_k = \frac{1}{m - 1 - |N(k)|} \sum_{h \neq k, h \notin N(k)} avgcpi_h \quad \forall k = 0, \ldots, m-1, \tag{28}$$

$$x_{ik} \in \{0, 1\} \quad \forall i = 0, \ldots, n-1 \quad \forall k = 0, \ldots, m-1, \tag{29}$$

where x_{ik} is the binary decision variable indicating if a job i is mapped on core k or not. The parameter cpi_i represents the average Clock Per Instructions (CPI) characterizing job i, and is a measure of the difficulty of job i. The objective is to maximize the worst-case core efficiency, and the fitted model $\hat{h}_k$ is used to predict the efficiency of core k that is represented by $y_k \in [0, 1]$.

Constraints (24) ensures that each job is mapped to only one core, and (25) forces the same number of jobs to run on each core. Constraints (26), (27) and (28) are used to compute the average CPI for a core k, the average CPI for the cores in the neighborhood of k ($N(k)$), and the average CPI for cores not in the neighborhood of k respectively. Lombardi et al. (2017) conclude that learning the efficiency function for each core by means of neural networks (with one hidden layer of two nodes and $\tanh$ activation function) is computationally intractable. On the contrary, our experiments show that we are able to solve this problem using neural networks with one hidden layer and 10 nodes in a reasonable amount of time (19.4 seconds). We tried deeper neural networks, but the increase in computational complexity did not lead to a gain in predictive performance.

Chapter 4

Embedding Machine Learning Based Toxicity Models within Radiotherapy Treatment Plan Optimization

4.1 Introduction

Radiation-induced toxicity (RIT), an unavoidable downside of all radiotherapy (RT) treatments, is a critical bottleneck and a major concern throughout patients' treatment course, from treatment planning and delivery through post-RT follow-ups. Traditionally, RIT associated with a given RT plan has been estimated using normal tissue complications probability (NTCP) models such as, Lyman (Lyman 1985), Lyman-Kurcher-Burman (LKB) (Kutcher and Burman 1989), relative seriality (RS) model (Källman et al. 1991), and their variants. The reliance of these models on only external (dosimetric) factors and ignoring the impact of patients' underlying risk factors such as physical health and genetic predispositions limit the potential of these models when applied to individual patients, which is the aim of *personalized radiotherapy* (Baumann et al. 2016). In recent years, ML -based outcome models have been introduced as more powerful tools for estimating RIT. These models are capable of synthesizing (many) dosimetric and non-dosimetric risk factors, often showing superior performance (*i.e.*, predictive power) when compared to their non-ML counterparts (see Isaksson et al. (2020) for an overview of ML-based outcome modeling of RIT).

Regardless of which model is used to estimate the risk of toxicity, in a RIT-cognizant, personalized RT, the estimated risks should be translated into "clinically actionable decisions." That is, one must learn how to adjust the RT treatment plan to keep the estimated *patient-specific* risk within tolerable bounds. Despite this, the field of ML-based outcome modeling remains largely disconnected from clinical RT planning. Using outcome models directly during treatment planning has so far remained limited to NTCP-based RIT models (see Witte et al. (2007), Semenenko et al. (2008), Nahum and Uzan (2012), Kierkels et al. (2014, 2016), for some examples), again making the resulting treatment plans more population-based than personalized. In the work of Ajdari et al. (2022), the authors proposed a framework for "calibrating" the prediction of conventional NTCP model according to the prediction of an ML-based outcome model (Bayesian Networks) for prediction of radiation pneumonitis (RP). The *updated* NTCP model was then embedded within the treatment plan optimization to obtain RT

plans more tailored toward patients' individual RP risk. While the authors effectively integrate a predictive model with treatment planning optimization, the approach separates ML predictions from the optimization module. In other words, the ML model is utilized in the optimization solely indirectly. As a result, the resulting treatment plans are not optimized based on the actual risk but assume a fixed risk during the optimization process, independent of the optimized decision variables.

In this work, we use a novel methodology for *directly* embedding entire ML risk models into treatment planning, thus closing the gap between ML-based outcome prediction and treatment plan optimization. The approach is built upon the concept of *optimization with constraint learning* (OCL) (Maragno et al. 2023, Biggs et al. 2023, Verwer et al. 2017) for learning mathematical representation (constraints) of ML predictive models directly and directly embedding them within the treatment plan optimization. Although generalizable to different types of cancer and toxicity endpoints, we will build our framework on a non-small cell lung cancer (NSCLC) cohort, with the goal of personalizing the treatment plans to minimize the *patient-specific* risk of RP.

By integrating biomarker data, ML-based risk prediction, and treatment plan optimization, our approach provides a computationally tractable means of achieving personalized treatment planning that can be customized to other treatment sites beyond lung cancer. As a result, this study represents a crucial step forward in the pursuit of more efficient and tailored cancer treatment.

4.2 Methodology

Our method is based on the OCL framework (Fajemisin et al. 2024). The framework is generic and can be deployed on a wide range of problems to learn mathematical representations (constraints) from ML models. In the context of RT treatment plan optimization (TPO), OCL can be used to learn a variety of outcome models, such as TCP, NTCP, and any combinations thereof (*e.g.*, uncomplicated tumor control probability, UTCP (Chaikh et al. 2016)). OCL leverages ML to design optimization models in which (parts of the) constraints and objectives are directly learned from data when an explicit expression is unknown or the conventional formulations are deemed inadequate, which is the case for conventional TCP/NTCP models. In the remainder of this section, we describe the general OCL framework in a TPO context. In the last part of this section, we describe the design of the experiments to evaluate our methodology.

4.2.1 General OCL framework

Suppose we have a dataset $\mathcal{D} = \{(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{w}}_i, \hat{\boldsymbol{y}}_i)\}_{i=1}^{N}$ of size N, with $\hat{\boldsymbol{x}}_i$ the observed decision variables (*e.g.*, mean total lung dose), $\hat{\boldsymbol{w}}_i$ the contextual information[1] (*e.g.*, patient's age, gender, or tumor location), and $\hat{\boldsymbol{y}}_i$ the outcomes of interest for sample i (*e.g.*, risk of developing RIT side effects), such that relationship between these variables has to be learned from data. An OCL framework involves three primary steps. First, the conceptual model is designed, including the formulation of decision variables, objective function, and the constraints that are to be learned. Next, one or more predictive models are fitted to historical data to estimate the outcomes of interest. Finally, the predictive models are embedded into the optimization model which is now ready to be solved. Figure 4.1 provides a schematic representation of the OCL framework. In the remainder of this section, we discuss each step of the OCL framework in a radiation therapy setting.

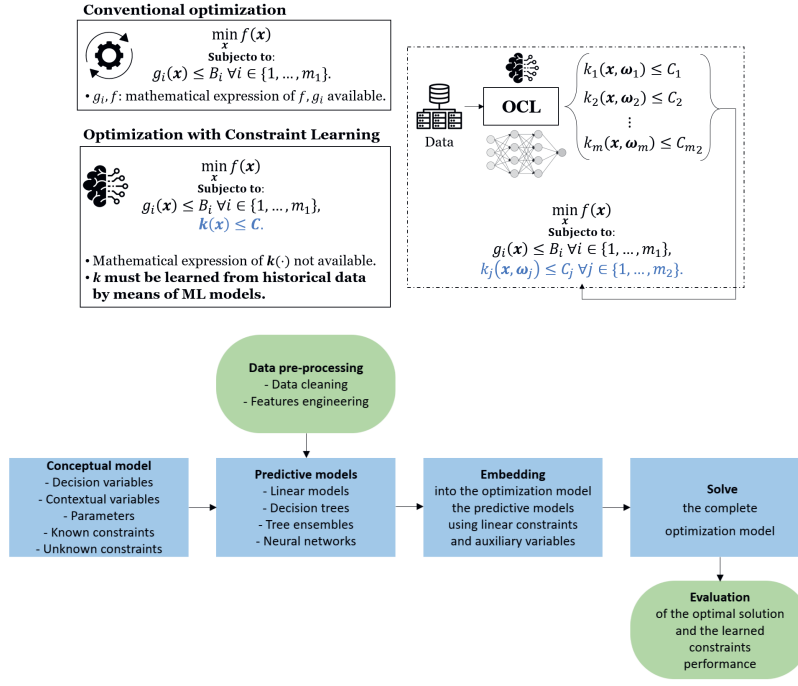


Figure 4.1. Schematic representation of an Optimization with constraint learning framework. The flowchart is based on (Fajemisin et al. 2024).

[1]Contextual features are fixed attributes that provide additional information about the context or environment. These features remain constant throughout the optimization process, providing important context for the predictive model.

Conceptual model

Let n represent the dimension of the beamlet intensity vector $\boldsymbol{x}$, p denote the number of non-decision variable features $\boldsymbol{w}$, and k indicate the dimension of $\boldsymbol{y}$. The dataset $\mathcal{D}$ contains historical solution information and has a dimension of $N \times (n' + p + k)$, where N represents the number of data points, and n' stands for the number of derived features obtained using the function $\boldsymbol{d}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}$. Some examples of derived features include mean dose, max dose, and V20Gy. With the decision variable $\boldsymbol{x} \in \mathbb{R}^n$ and the fixed feature vector $\boldsymbol{w} \in \mathbb{R}^p$, a TPO conceptual model can be defined as

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathbb{R}^n} \; & f(\boldsymbol{x}) \\ \text{s.t. } & g_i(\boldsymbol{x}) \leq 0, \quad \forall i = 1, \ldots, m, \\ & y = \hat{h}(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{w}), \\ & y \leq \tau, \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{aligned} \tag{4.1}$$

where $f(\cdot)$ is a dose fidelity function and $g_i(\cdot)$ represents the conventional dosimetric (mean-dose, max-dose, and dose-volume histogram constraints) and/or *known* biological constraints (*e.g.,* equivalent uniform dose, biologically effective dose, tumor control probability). The learned outcome is represented by the function $\hat{h}(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{w})$ which is an ML model trained on $\mathcal{D}$. We do not use the "hat" notation for $\boldsymbol{x}$, $\boldsymbol{w}$, and y because these variables represent a distinct and unobserved instance, which is not part of the dataset $\mathcal{D}$. The predictive model outcome, denoted as y, corresponds to the probability of a patient experiencing a serious complication or not, subject to the constraint that it remains below a threshold value τ. We note that the embedding of a single learned outcome may require multiple constraints and auxiliary variables as shown in the next section.

Predictive models

The second step of the OCL framework involves exploring various ML models to learn the outcome of interest, denoted as variable y in the conceptual model (4.1). In RT, the goal is to find a predictive model that accurately approximates the probability of a patient experiencing high levels of toxicity. To evaluate each predictive model, we consider four key metrics, summarized in Table 4.1 and described below:

- Interpretability: the selection of a predictive model may depend on whether the practitioner requires the model to be interpretable. Interpretable models, such as linear models or decision trees, can provide insight into the factors that influence

Model	Interpretability	Approximation accuracy	Amount data required	Computational complexity	
				complexity class	embedding
Linear models	high	low	small	LO	# constraints: 1
SVM (nonlinear Kernels)	moderate	moderate	moderate	NLO	*
Decision trees	moderate	moderate	moderate	MIO	# binary vars: 2^k # constraints: $2^{k+1}-2$
Tree ensembles	low	high	moderate	MIO	# binary vars: $2^k T$ # constraints: $(2^{k+1}-2)T$
Bayesian networks	moderate	high	moderate	NLO	*
Neural networks (ReLU)	low	high	high	MIO	# binary vars: 1 per node # constraints: 4 per node
Neural networks	low	high	high	NLO	*
Recurrent neural networks	low	high	high	NLO	*
Convolutional neural networks	low	high	high	NLO	*

* no available literature on the embedding of this predictive model

k is the depth of a complete decision tree. T is the number of trees

LO: linear optimization; NLO: nonlinear optimization; MIO: mixed-integer optimization

Table 4.1. Pros and cons of predictive models commonly used in optimization with constraint learning.

the predictions.

- Approximation accuracy: a predictive model must be able to accurately approximate the underlying function that generates the data. The ability of a model to fit a wide range of functions is an important consideration when selecting a predictive model. More flexible models, like neural networks, can potentially approximate more complex functions.

- Data required: the quantity and quality of data required to fit a model can vary depending on the model's complexity and the amount of information it can extract from the data. More complex models may require larger and higher-quality datasets to achieve good performance.

- Complexity class: the selection of a predictive model impacts the computational complexity class of the conceptual optimization model. Linear models typically do not affect the complexity class, while other models like tree ensembles and neural networks require binary variables to be represented as constraints, which increases the model's complexity. In some cases, such as Bayesian networks and neural networks with activation functions other than rectified linear unit (ReLU), the embedding requires complex (non-convex) constraints.

The interpretability of predictive models is crucial in clinical decision-making for radiation treatment planning. It ensures transparency, trust, and ethical considerations, allowing clinicians to comprehend and validate the model's reasoning for accurate and accountable treatment plans. The outcomes of interest depend on multiple factors, including dosimetric, clinical, and imaging information, making their approximation challenging, especially with scarce, fragmented, and incomplete data. Moreover, the

complexity class of the predictive model significantly impacts the tractability of the final optimization problem. Since the conceptual model may involve a large number of constraints and decision variables, incorporating complex predictive models can lead to problems that cannot be solved within a reasonable timeframe.

Embedding the predictive models

In the final step of our framework, we integrate the trained predictive models into the conceptual model. In this section, we focus on those predictive model belonging to the complexity class LO (linear optimization) and MIO; see Table 4.1. It is important to note that although the embedding of the predictive models is done using piece-wise linear constraints, it does not imply that the function being approximated is linear. This is possible for various types of predictive models, such as linear models, decision trees, tree ensembles, and neural networks, and is applicable to both classification and regression tasks. In light of the paper's objective, the following discussion focuses on binary classification models. For readability, we use the notation $\boldsymbol{x}$ to indicate both the vector of decision variables and the derived features previously denoted as $\boldsymbol{d}(\boldsymbol{x})$.

Below, we show the formulations adopted by Maragno et al. (2023) to embed these predictive models, and we refer to them for further details.

Logistic regression. Logistic regression is a natural choice due to its inherent linearity and ease of integration. Once the model has been trained, only the final coefficient vectors $\boldsymbol{\beta_x} \in \mathbb{R}^n$ and $\boldsymbol{\beta_w} \in \mathbb{R}^p$ (along with the intercept term β_0) are necessary to describe the model. In classification tasks, these models find a hyperplane that separates positive and negative samples. The prediction for a given sample is computed as

$$y = \frac{1}{1 + e^{\beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w}}}.$$

This function, known as the sigmoid function, is nonlinear. However, if we impose the constraint $y \geq \tau$, where τ represents the desired probability lower bound, this constraint is satisfied when $\beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq \ln\left(\frac{\tau}{1-\tau}\right)$. Consequently, the binarized output of logistic regression, using a threshold of τ, can be expressed as follows:

$$y = \begin{cases} 1, & \text{if } \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq \ln\left(\dfrac{\tau}{1-\tau}\right); \\ 0, & \text{otherwise.} \end{cases}$$

For example, at a threshold of $\tau = 0.5$, the predicted value is 1 when $\beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq 0$. Here, τ can be chosen according to the minimum necessary probability to predict 1.

Support vector machines. Similar to logistic regression models, linear support vector machines (SVM) aim to identify the optimal hyperplane that can effectively separate positive and negative samples (Cortes and Vapnik 1995). A trained binary classification SVM also provides coefficients $\boldsymbol{\beta_x}$, $\boldsymbol{\beta_w}$, and β_0, and the prediction for a given sample can be expressed as follows:

$$y = \begin{cases} 1, & \text{if } \beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

In SVM, the output variable y is binary rather than a probability, and as a result, it does not necessitate the use of the threshold parameter τ. In this case, the constraint can simply be embedded as $\beta_0 + \boldsymbol{\beta_x}^\top \boldsymbol{x} + \boldsymbol{\beta_w}^\top \boldsymbol{w} \geq 0$. In the case of SVM with a non-linear kernel (such as the Gaussian kernel or polynomial kernel), the embedding process differs, and its linearization is not possible.

Decision trees. A generic decision tree of depth two is shown in Figure 4.2. A split at node i is described by an inequality $A_i^\top \boldsymbol{x} + A_{iw}^\top \boldsymbol{w} \leq b_i$, where A_i represents the coefficients associated with the input features $\boldsymbol{x}$, and A_{iw} captures the weight of auxiliary variables $\boldsymbol{w}$. To simplify the notation, we include the effect of A_{iw} in the constant term b_i and omit it for the remainder of this paragraph. We assume that A_i can have multiple non-zero elements, in which we have the hyper-plane split setting; if there is only one non-zero element, this creates a parallel (single feature) split. Each terminal node j (*i.e.*, leaf) yields a prediction (p_j) for its observations. In binary classification, the prediction is the proportion of leaf members with the feasible class.

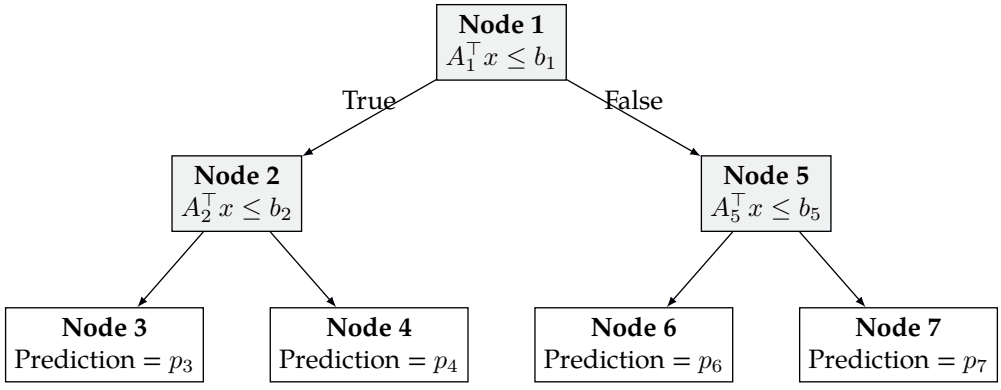


Figure 4.2. Decision tree of depth two.

Suppose that we aim to constrain the predicted value of this tree to be at most τ, a fixed constant between 0 and 1. Once we have trained the tree in Figure 4.2, we can identify which paths satisfy the desired bound ($p_i \leq \tau$). Suppose that p_3 does satisfy

the bound, but p_4, p_6, and p_7 do not. In order to enforce the decision vector $\boldsymbol{x}$ to reach leaf p_3, we can use the constraints:

$$\mathcal{P}_3 = \begin{cases} A_1^\top \boldsymbol{x} \leq b_1, \\ A_2^\top \boldsymbol{x} \leq b_2. \end{cases}$$

When more than a leaf satisfies the condition $p_i \leq \tau$, we can decompose the problem into multiple separate problems, one per feasible leaf. The learned constraints represented by $\hat{h}(\cdot)$ in the conceptual model (4.1) become $(\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{w}) \in \mathcal{P}_i$ where the learned constraints for leaf i's subproblem are implicitly represented by the polyhedron $\mathcal{P}_i$. These subproblems can be solved in parallel, and the minimum across all subproblems is obtained as the optimal solution. An alternative approach is to represent the entire decision tree by utilizing auxiliary binary variables, which assist in determining the leaf to which the solution belongs, see (Verwer et al. 2017, Halilbašić et al. 2018).

The computational complexity arising from the embedding of tree-based models is determined by the total number of nodes and leaves. Therefore, it becomes crucial to leverage the values of the contextual features $\boldsymbol{w}$ to selectively prune unreachable leaves. This is especially significant in radiation therapy since most features are patient-specific and known prior to solving the optimization model. In practical terms, for each patient, we first prune the fitted decision tree by removing leaves that cannot be reached based on the contextual features. Subsequently, we incorporate the pruned tree into the optimization model. An example of pruning is shown in Section 4.2.2.

Tree ensembles. Ensemble methods, such as random forests and gradient-boosting machines, utilize multiple decision trees combined to generate a single prediction for an observation (Breiman 2001). These models can be implemented by embedding multiple sub-models representing each tree. In random forests, predictions are typically obtained by averaging the predictions from individual trees as follows:

$$y = \frac{1}{P} \sum_{i=1}^{P} y_i,$$

where P represents the number of trees in the ensemble, and y_i denotes the predicted value from the i-th tree. On the other hand, gradient-boosting machines calculate the model output by summing the predictions of the individual regression models, weighted by corresponding coefficients:

$$y = \sum_{i=1}^{P} \beta_i y_i,$$

where y_i is the predicted value of the i-th model, and β_i represents the weight associated with the prediction.

Neural networks. Neural networks with a ReLU activation function are known to be linearly representable (Grimstad and Andersson 2019, Anderson et al. 2020). These networks typically consist of an input layer, $L-2$ hidden layers, and an output layer. Importantly, the ReLU operator, $v = \max\{0, x\}$, can be encoded using linear constraints:

$$v \geq x, \tag{4.2}$$
$$v \leq x - M_L(1-z), \tag{4.3}$$
$$v \leq M_U z, \tag{4.4}$$
$$v \geq 0, \tag{4.5}$$
$$z \in \{0,1\}. \tag{4.6}$$

Here, the large numbers M_L and M_U stand for lower and upper bounds on all possible values of x. Although this embedding relies on a big-M formulation, there are ways to enhance it. The constraints for a neural network can be recursively generated, starting from the input layer, allowing the embedding of trained networks with any number of hidden layers and nodes. The output layer of the neural network consists of a single neuron with a sigmoid activation function. Similarly to logistic regression, the sigmoid function can be represented as a linear constraint using the τ bound:

$$y = \begin{cases} 1, & \text{if } \beta_0^L + \boldsymbol{\beta}^{L\top} \boldsymbol{v}^{L-1} \geq \ln\left(\dfrac{\tau}{1-\tau}\right); \\ 0, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{v}^{L-1}$ represent the output vector of the penultimate layer $L-1$.

Literature on predictive models in RIT. Table 4.2 provides an overview of published papers on predictive models for radiation treatment of head and neck (H&N), lung, and prostate cancers, focusing on xerostomia, pneumonitis, and rectal toxicity/bleeding due to their prevalence in ML-based predictions. The last column of the table categorizes the feasibility of embedding the ML models in treatment plan optimization as follows:

- the model is linear or linearly representable, and the methodology proposed in this study can be followed for the embedding,
- the model is linear or could be linearly representable, but this study does not provide the methodology for embedding the learned constraints,

- the model is not linearly representable.

Among the selected toxicities, 19 out of 23 (82.6%) literature models belong to the first category, indicating that they can be embedded in the optimization process following the methodology proposed in this paper. However, it is noteworthy that Linear Discriminant Analysis (LDA) and Distance-based (norm) models, while being MIO-representable, are not incorporated into our methodology due to the absence of documented MIO embeddings in the existing literature. In contrast, both Naïve Bayes and Bayesian Networks cannot be represented using an MIO formulation. Attempting to embed them into the optimization model, though theoretically feasible, would introduce highly non-convex constraints, thereby making the model impractical to solve to optimality.

Cancer site	Toxicity	Classifier	References	Embedding
H&N	Xerostomia	LR	Jiang et al. (2019), Nakatsugawa et al. (2019)	●
		MV-LR	van Dijk et al. (2017)	●
		Naïve Bayes	Pota et al. (2017)	○
		Decision tree	Zhang et al. (2009)	●
		SVM	Gabryś et al. (2018), Zhang et al. (2009)	●
		extra-trees	Gabryś et al. (2018)	●
Lung	Pneumonitis	RF	Luna et al. (2019), Valdes et al. (2016)	●
		LASSO	Krafft et al. (2018)	●
		Decision tree	Valdes et al. (2016), Das et al. (2008, 2007)	●
		RUSBoost	Valdes et al. (2016)	●
		Bayesian network	Lee et al. (2015)	○
		LR	Lee et al. (2015), Naqa et al. (2009)	●
		SVM	Chen et al. (2007), Naqa et al. (2009), Das et al. (2008)	●
		NN	Naqa et al. (2009), Das et al. (2007)	●
Prostate	Rectal toxicity/bleeding	LR	Rossi et al. (2018), Oh et al. (2017), Liu and Li (2016), Tomatis et al. (2012)	●
		Distance-based (norm)	Fargeas et al. (2018, 2015)	◐
		RF	Oh et al. (2017), Ospina et al. (2014)	●
		LDA	Fargeas et al. (2015)	◐
		SVM	Fargeas et al. (2015), Pella et al. (2011)	●
		NN	Tomatis et al. (2012), Pella et al. (2011), Gulliford et al. (2004)	●

Abbreviations: LDA, linear discriminant analysis; LR, logistic regression; MV-LR, multivariate logistic regression; NN, neural network; RF, random forest; SVM, support vector machine; Symbols: ●: linearly representable and considered in the methodology; ◐: linearly representable but not considered in the methodology; ○: not linearly representable and not considered in the methodology

Table 4.2. Supervised ML models for predicting radiation-induced toxicity (adapted from Isaksson et al. (2020)).

4.2.2 Experimental design

The experiments aim to optimize the treatment plan for patients with NSCLC while constraining the risk of symptomatic pneumonitis (grade ≥ 2), referred to as RP2+, to be smaller than a given threshold (τ). The experiments consist of two parts. In the first part, we design a (simplified) toy example with voxel spacing values of [10, 10, 10], which allows for faster optimization even when complex predictive models, like neural networks, are deployed to constrain the RP2+ risk. In the second part, we use voxel spacing of [5, 5, 5] to demonstrate the performance on realistic (clinical) cases and use a decision tree to model the RP2+ outcome. In both scenarios, we use a spot spacing and layer spacing equal to 10, and a target margin equal to 5. In A, we report the size, in terms of beamlets and voxel sizes, of the optimization problems

solved for the clinical study. The first part of the experiments demonstrates how various predictive models, including those discussed in Section 4.2.1, can be used to learn the probability of toxicity. We conduct plan adaptations for patients with an RP2+ predicted risk higher than 0.5 and sensitivity analyses by modifying both the RP2+ threshold and several patient-specific parameters that impact the predictive model's outcome. The second part showcases the suitability of decision trees due to their interpretability and their linear embedding (without the use of binary variables). We present adjusted treatment plans for four patients who would otherwise have a probability greater than 0.5 of experiencing RP2+ according to the fitted decision tree. Since the primary objective of this work is to offer practitioners a structured approach for incorporating predictive models into treatment planning, our discussion of the experiments will emphasize the interconnection between prediction and prescription rather than the fine-tuning of highly performing predictive models. In the remainder of this section, we offer additional information about the conceptual model we adopted, the dataset utilized for training the predictive models, and the specific predictive models employed to assess the RP2+ risk. Finally, we provide a detailed explanation of how a decision tree can be seen as a different set of rules depending on the patient's characteristics.

Conceptual model. The conceptual model used for the experiments is a treatment plan optimization model. A Max-dose constraint of 54 Gy-RBE is imposed on the spinal cord. For heart and total lung, we used mean dose constraints with an upper bound respectively in the range 15.06 - 20.03 Gy-RBE and 19.15 - 21.32 Gy-RBE, depending on the patient. The probability of experiencing RP2+ is constrained to be less than 0.5 (τ), however, it is worth noting that any value between 0 and 1 can be used. We recommend that the threshold value is chosen in consultation with a physician. The reference doses for the gross tumor volume (GTV) and planning target volume (PTV) in the objective function are 66 Gy-RBE and 65 Gy-RBE, respectively. We use the unit "Gy-RBE" to keep the doses consistent between photon and proton treatments in terms of their biological effect. There were no specific constraints on RBE or its range during the treatment planning. As per standard practice, we scale the dosimetric metrics for proton planning using a weighting factor of RBE = 1.1 to accommodate the biological differences between protons and photons. To ensure that the optimal solution yields low mean doses delivered to normal tissues, we introduce a penalty in the objective function by incorporating a weighted sum of the mean doses delivered to the heart, lung, and spinal cord. We give equal weight to all objectives, and this remains consistent in the OCL-adapted plans too.

Dataset. We built our predictive models on a dataset of 73 stage IIB-IIIA NSCLC patients treated with passive scattering proton or intensity-modulated RT (Liao et al. 2018). More details about the dataset can be found in (Ajdari et al. 2022). Among the

patients in the dataset, 27% of them experienced grade 2+ RP, which serves as the outcome for the predictive models. The features used to predict RP2+ can be categorized into three main groups: dosimetric information (*e.g.*, mean lung dose), clinopathological information (*e.g.*, age, smoking status, breathing function), and tumor-related information (*e.g.*, GTV size, histology, tumor location). Before training the predictive model, categorical features are encoded into numerical values using one-hot encoding. We utilized the same dataset throughout the experiments. However, in the first part of the experiments, we excluded a set of features associated with the calculation of the deposition matrix, such as the GTV size, as this was necessary to facilitate the sensitivity analysis.

Predictive models. We trained four predictive models:

- Logistic regression with ℓ_1 penalty term and default regularization coefficient (C=1),
- Decision tree with a maximum depth equal to 4 (max_depth=4),
- Random forest composed of 10 decision trees each with a max depth of 4,
- Neural network with one hidden layer of 10 nodes activated with a ReLU activation function.

In Figure 4.3, we report the fitted decision tree used for the second part of the experiments. Among the 73 patients available in the dataset, 19 of them were selected to evaluate the performance of the model: 14 patients did not develop RP (RP2+ = 0) and 5 did (RP2+ = 1). The fitted decision tree yields a cross-validation AUC of 0.66 (95% confidence interval, CI=[0.65, 0.68]) and a test AUC of 0.66. While we recognize the significance of achieving a predictive model with a high AUC score, improving model accuracy lies beyond the scope of our present work.

From decision tree to treatment constraints In the second part of the experiments, we evaluate our method by conducting tests on several patients. For each patient, we first solve an optimization model (baseline_model) without incorporating the learned constraints. Subsequently, we assess whether the patients, considering their current treatment, had a probability greater than 0.5 of developing pneumonitis within two months. The decision tree shown in Figure 4.3 indicates that a patient receives an RP2+ score smaller than 0.5 if one of the following three scenarios is met:

(i) The GTV is less than or equal to 55.20 cc.

(ii) The GTV is greater than 55.20 cc and the FEV1 pre-bronchodilator (BD) score is greater than 2.52.

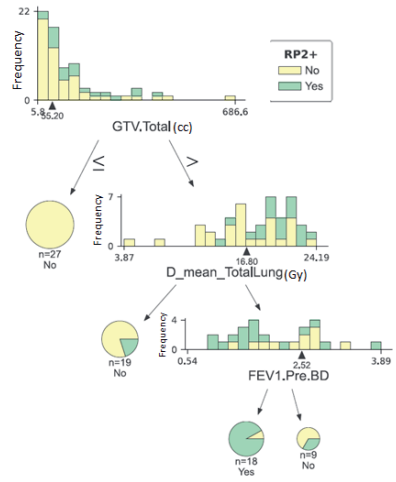


Figure 4.3. Classification tree trained on the radiation pneumonitis dataset. The three split nodes are represented as histograms with the frequency on the y-axis and the following features: total GTV measured in cubic centimeters, mean total lung dose measured in Gy, and FEV1.Pre.BD. Every split node is defined by a threshold, symbolized by the black triangle positioned on each x-axis of the histograms. The histograms are constructed using the training samples able to reach the respective split node.
GTV: gross tumor volume; D_mean_TotalLung: mean dose delivered to the total lung; FEV1.Pre.BD: forced expiratory volume in the first second before bronchodilator administration.

(iii) The GTV is greater than 55.20 cc, the FEV1 pre-bronchodilator (BD) score is less than or equal to 2.52, and the total lung mean dose is greater than 16.80 Gy.

The decision tree structure ensures that only one set of conditions can be active at a time, depending on the patient's information and the treatment plan. If none of these conditions are satisfied, the RP2+ score for the patient, given their treatment plan, will be higher than 0.5 (bottom left leaf). To illustrate how different conditions are active for different patients, let us consider the cases of three patients (see Figure 4.4 for a visual representation). The first patient has a GTV of 45 cc, falling under scenario (i). Regardless of the other features, the decision tree predicts a low probability of RP2+. The second patient has a GTV of 60 cc and an FEV1 pre-BD score of 3, corresponding to scenario (ii). Regardless of the mean dose delivered to the lung, the patient is unlikely to experience side effects according to the DT. Finally, the third patient has a GTV of 60 cc, but this time the FEV1 pre-BD score is 2. This scenario corresponds to (iii), and the only way to achieve an RP2+ score smaller than 0.5 is to design a treatment plan with a mean total lung dose below 16.80 Gy. While some patients fall into scenarios (i) or (ii), where ensuring a low RP2+ score is not necessary to adapt the treatment plan, other patients require a treatment plan that ensures a mean total lung dose smaller than the

threshold suggested by the tree. These three scenarios demonstrate that for different patients, only certain leaves of the decision tree are active, and therefore, reachable while the remaining leaves can be excluded from consideration. In the next section, we will incorporate the learned decision tree into the optimization model and observe how the solution changes for some of the patients.

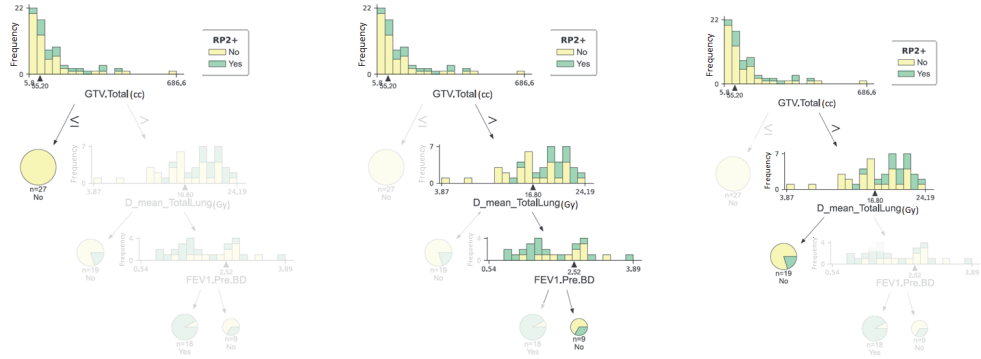


Figure 4.4. Side-by-side comparison of a fitted decision tree illustrating different paths leading to the prevention of radiation pneumonitis development with a probability greater than 0.5 for three distinct patients.

4.3 Results

For the experiments, we use the open-source Python package OpenTPS (Wuyckens et al. 2023) for dose calculation and visualization. We embed the predictive model into the optimization framework using the open-source Python package OptiCL (Maragno and Wiberg 2021). The optimization model is written and solved in Gurobi (Gurobi Optimization, LLC 2022). The training of the predictive models is performed using the Python library Sklearn v1.0.2 (Pedregosa et al. 2011). All computations are performed on a laptop i7-10850H CPU 2.70GHz with 32 GB of RAM.

4.3.1 Toy example

In this first part of the experiments, we use a simplified treatment planning problem and employ three distinct predictive models, namely logistic regression, random forest, and neural network, to learn the RP2+ risk function. In this context, we demonstrate how the adjusted treatment plan varies based on the chosen predictive model used to define the RP2+ risk for a given patient. The dose-volume histograms (DVH) can be found in Figure 4.5, and the dosimetric parameters are detailed in Table 4.3. The

results show that across all three scenarios, adjustments to the plans are necessary to decrease the RP2+ risk below the 0.5 threshold. In fact, the logistic regression model assigns a 57% RP2+ risk probability to the baseline plan, the random forest model assigns 55%, and the neural network model assigns 87%. The adapted plan obtained using logistic regression shows a reduction of 1.25 Gy in mean total lung dose, compared to the 1.68 Gy of the random forest, and the 0.95 Gy of neural network. However, this decrease in mean total lung dose required an increase in mean dose for the heart and spinal cord, ranging between 0.47 Gy and 0.56 Gy for the heart and 0.06 Gy and 0.09 Gy for the spinal cord. The adapted plans exhibit variations in the delivered dose to GTV and PTV, evident especially in terms of D98, which remains below the 2.46 Gy (random forest).

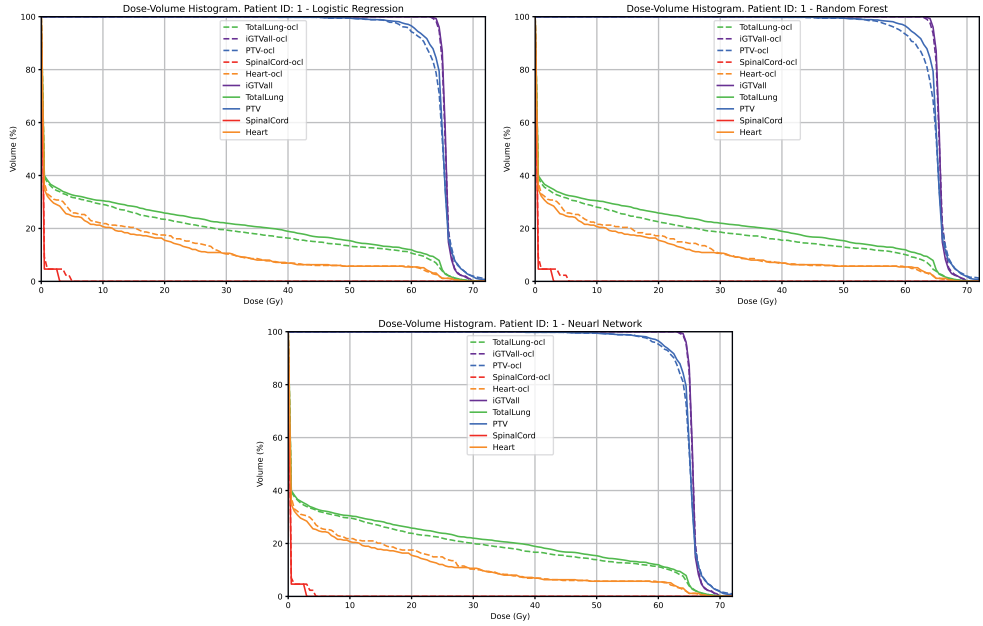


Figure 4.5. Comparison of dose-volume histograms between baseline and OCL adapted plans using logistic regression, random forest, and neural network approaches to capture the radiation pneumonitis risk constraint.

Using the same toy example, we run two sensitivity analyses. The first analysis, shown in Figure 4.6, involves varying the threshold τ in model (4.1) to observe its impact on treatment planning, represented by the PTV D98 and the mean total lung dose. As the threshold is incrementally raised, the plots illustrate the changes in the treatment solution, allowing for higher mean total lung dose and PTV D98 until it plateaus. In the cases of random forest, it is not possible to obtain a feasible solution to the resulting optimization problems for a threshold smaller than 0.5. This analysis provides insights into the behavior of the predictive models concerning the constraint threshold, helping

to identify critical points where the optimal treatment plan attains a steady state and remains insensitive to further changes in the threshold value.

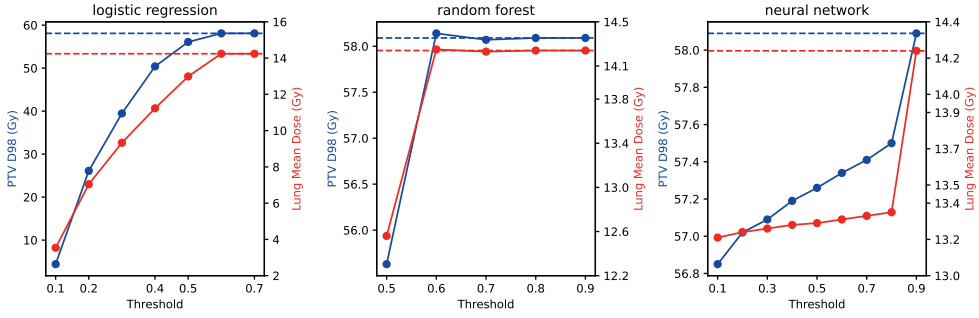


Figure 4.6. Effect of the radiation pneumonitis constraint (upper bound) threshold (horizontal axis) on the optimal treatment plan (vertical axis) before (dashed line) and after (solid line) adaptation.

In Figure 4.7, we report the results of the sensitivity analyses performed using three different predictive models: logistic regression, random forest, and neural network. For each predictive model, we select the three most relevant features that influence the RP2+ risk prediction. These features were perturbed individually to observe their effects on the optimal solution while ensuring a maximum risk threshold of $\tau = 0.5$ for the RP2+ risk. The conducted experiments are patient-specific and offer insights into the treatment's sensitivity to changes in features. Both the logistic regression model and the neural network model demonstrate that a higher Karnofsky score allows for a higher mean total lung dose and PTV dose while ensuring an RP2+ risk remains below the threshold. The logistic regression model indicates that patients with a higher average of daily smoked cigarettes and those who are currently smoking can tolerate a higher radiation dose. Moreover, the neural network model suggests that for older patients, the delivered dose should decrease, while higher values of FVC (forced vital capacity) corresponding to healthier lung function allow for a higher radiation dose.

4.3.2 Clinical study

In the second part of the experiments, we run a clinical study to compare the treatment planning solutions before and after the adaptation for some of the patients falling in scenario (iii); see Section 4.2.2. The adaptation is done using the decision tree in Figure 4.3 to constrain the risk of RP2+ to be smaller than 0.5. In Table 4.4, we summarize the optimal treatment plan for 4 patients using various metrics.

For the first two patients, we display in Figures 4.8-4.9 the DVH, as well as the dose distribution (a) before adaptation, (b) after adaptation, and (c) their voxel-wise differ-

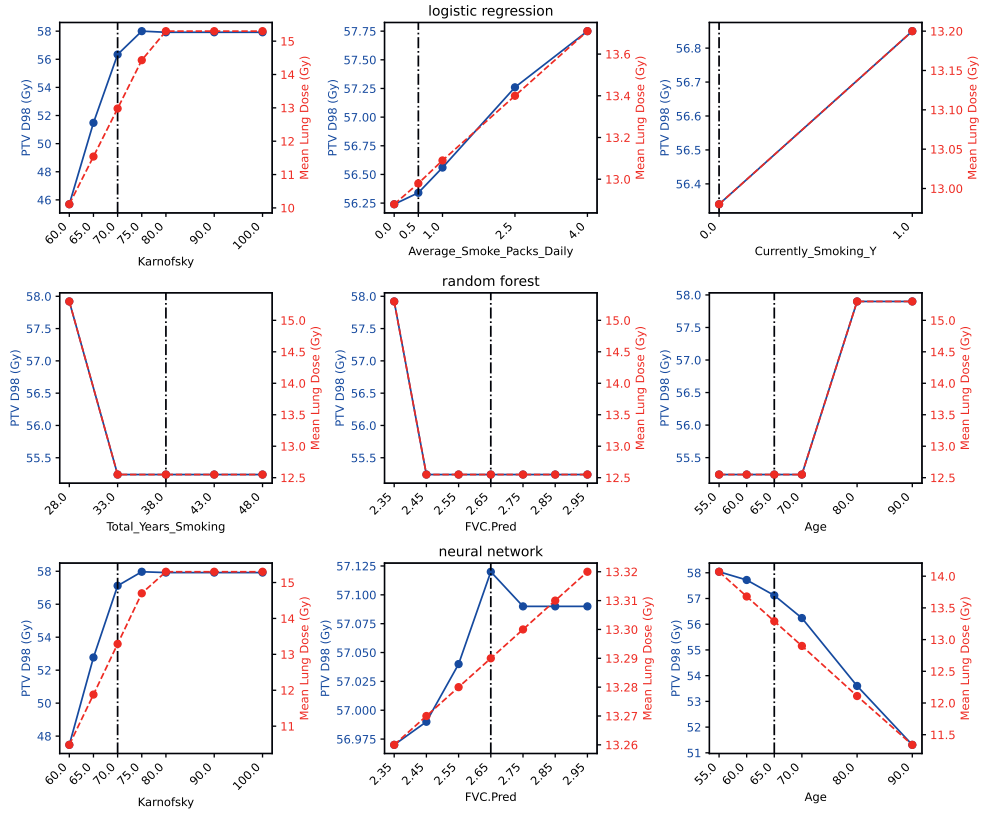
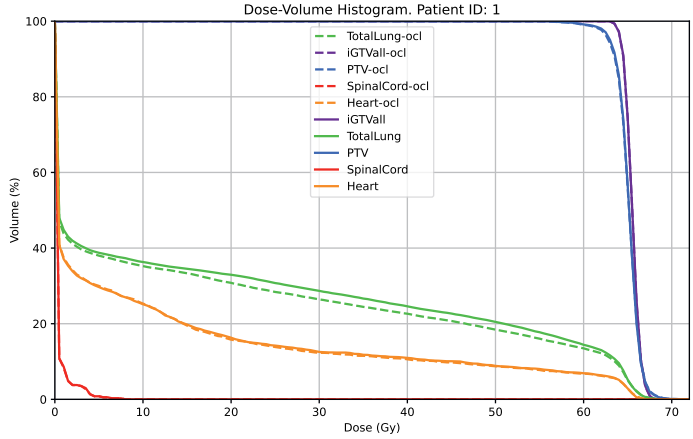


Figure 4.7. Impact of perturbed relevant features (horizontal axis) on the optimal treatment plan (vertical axis) while guaranteeing a radiation pneumonitis risk prediction smaller than 0.5. The black (dashed-dotted) line represents the original value of each feature, while the blue (solid) line depicts the planning target volume D98 value, and the red (dashed) line shows the mean total lung dose.

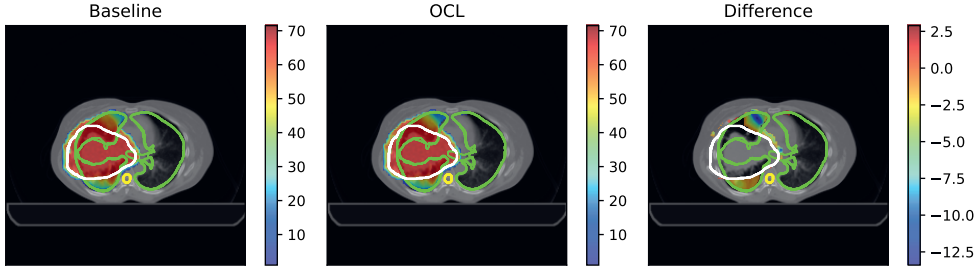
ence. Notably, the lung DVH is significantly reduced in all cases (mean dose reduction of 1.06 Gy, 1.85 Gy, 1.72 Gy, and 2.49 Gy). Occasionally, a lower lung dose may result in trade-offs with other organs at risk. For instance, in the adapted plan for Patient 2, the maximum spinal cord dose limit was respected in all adapted plans. In all the cases, the RP2+ probability prediction drops from 95% (baseline) to 42% (adapted).

4.4 Discussion

Despite the recent breakthroughs in the realm of ML-based outcome modeling in RT, currently, there is no rigorous methodology for integrating these predictions within

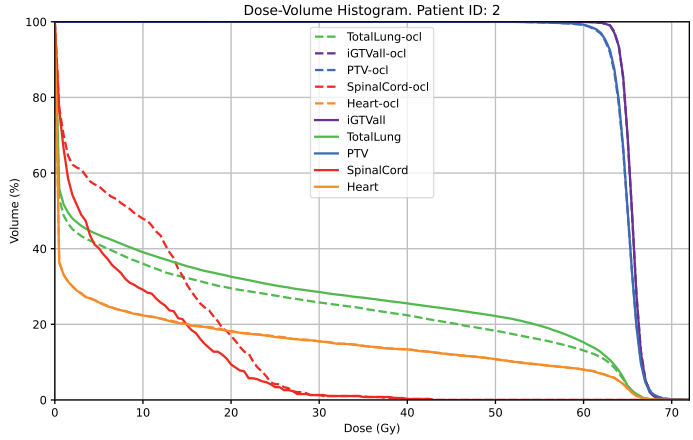


(*a*) Dose-volume histogram

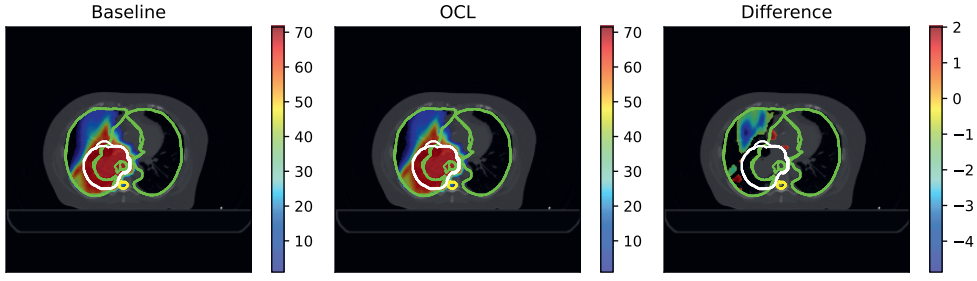


(*b*) Dose distribution (Gy)

Figure 4.8. Baseline vs Optimization with Constraint Learning (OCL) adapted plans: (a) dose-volume histogram for the baseline (solid) and the adapted (dashed) plans (b): CT images overlapped with dose distribution for (left) baseline plan, (mid) adapted plan, and (right) their difference. The contours for the planning target volume, total lung, and spinal cord are respectively colored in white, green, and yellow.



(*a*) Dose-volume histogram



(*b*) Dose distribution (Gy)

Figure 4.9. Baseline vs Optimization with Constraint Learning (OCL) adapted plans: (a) dose-volume histogram for the baseline (solid) and the adapted (dashed) plans (b): CT images overlapped with dose distribution for (left) baseline plan, (mid) adapted plan, and (right) their difference. The contours for the planning target volume, total lung, and spinal cord are respectively colored in white, green, and yellow.

ML model	Time (s)	GTV D2	GTV D98	PTV D2	PTV D98	Total Lung Dmean	Total Lung V5	Total Lung V20	Total Lung V60	Heart Dmean	Heart Dmax	Heart V5	Heart V20	Heart V60	Spinal Cord Dmax
Baseline	1.29	67.98	64.27	69.79	58.09	14.24	32.67	25.81	11.85	7.93	68.48	24.71	15.52	5.46	2.82
Logistic Regression	2.99	68.59	63.98	70.06	56.09	12.99	31.74	23.46	10.63	8.43	68.24	25.86	17.53	5.46	4.60
Random Forest	3.13	68.05	63.93	70.03	55.63	12.56	31.57	22.52	10.06	8.46	68.14	25.86	17.24	5.46	5.26
Neural Network	3.94	68.64	64.05	70.15	57.26	13.29	32.06	23.82	11.12	8.40	68.30	26.15	17.53	5.46	4.26

Table 4.3. Comparison of dosimetric parameters for various organs at risk obtained solving both baseline and adapted models using different predictive models for the radiation pneumonitis constraint with 0.5 as the upper bound. GTV stands for Gross Tumor Volume; PTV stands for Planning Target Volume; D2/98 are the minimum dose delivered to 98%/2% of the volume; V5/20/60 is the percentage of volume receiving at least 5/20/60 Gy; Dmean and Dmax are the mean and max dose in Gy delivered to the volume; Time is computation time, in seconds, required to optimize the treatment plan.

Patient	Model	Time (s)	GTV D2	GTV D98	PTV D2	PTV D98	Total Lung Dmean	Total Lung V5	Total Lung V20	Total Lung V60	Heart Dmean	Heart Dmax	Heart V5	Heart V20	Heart V60	Spinal Cord Dmax
1	baseline	248	67.00	63.67	67.34	61.89	**17.82**	38.75	32.92	14.33	9.80	67.18	29.62	16.30	6.90	7.66
	adapted	320	67.03	63.70	67.42	61.65	**16.76**	37.97	30.76	13.37	9.70	67.35	29.96	15.78	6.76	7.56
2	baseline	256	67.59	63.26	67.64	61.39	**18.65**	43.70	32.76	14.83	10.18	69.10	25.61	17.93	7.68	44.24
	adapted	352	67.58	63.28	67.75	61.24	**16.80**	41.13	29.50	12.97	10.25	69.21	25.79	18.22	7.62	45.03
3	baseline	788	66.50	64.10	66.61	62.95	**18.47**	36.59	33.98	17.38	4.57	66.96	11.87	7.95	3.32	40.00
	adapted	1096	66.60	64.04	66.82	62.37	**16.75**	35.83	30.37	15.57	4.59	66.97	12.07	7.99	3.45	39.40
4	baseline	30	69.98	60.93	70.60	55.19	**19.36**	51.41	38.41	9.92	7.13	76.40	17.88	12.29	5.27	0.93
	adapted	64	69.96	60.74	70.83	54.33	**16.77**	49.53	32.11	8.55	7.15	76.49	17.81	12.36	5.23	1.16

Table 4.4. Comparison of dosimetric parameters for various organs at risk obtained solving both baseline and adapted models for four different patients. GTV stands for Gross Tumor Volume; PTV stands for Planning Target Volume; D2/98 are the minimum dose in Gy delivered to 98%/2% of the volume; V5/20/60 is the percentage of volume receiving at least 5/20/60 Gy; Dmean and Dmax are the mean and max dose delivered to the volume; Time is computation time, in seconds, required to optimize the treatment plan.

RT treatment planning to guide the treatment design. In this study, we set out to address this gap. To the best of our knowledge, our work is the first study to formalize a methodology that directly integrates ML-based outcome models within RT treatment planning. Our methodology is capable of handling the majority of popular ML-based outcome models proposed in the literature (Isaksson et al. 2020), including decision trees, random forests, support vector machines (with linear kernels), neural networks, and general ensemble models. We show that the often complex structure of these models can be mathematically represented by a set of linear, mixed-integer constraints which can, in turn, be included in the conventional treatment plan optimization, thus bridging the gap between data-oriented outcome modeling and optimization-based treatment planning. Our methodology is capable of learning the radiation dose response directly from historical data (as long as the relationship can be captured through one of the proposed models), without the need to rely on over-simplified, *a priori* assumptions on the shape and nature of such response.

Marks et al. (2010) observed that a small reduction in the mean total lung dose had a relatively modest effect on the risk of RP, which might appear in conflict with our results, in which for some cases, even small reduction in MLD led to considerable RP risk reduction. However, we note that the reductions reported in Marks et al. (2010) are averaged over a population, encompassing a spectrum of patients. This means that the reported effects include individuals who do not experience any improvement in outcomes as well as those who are significantly affected, as indicated by the presence of large error bars in the data; second, our predictive model (decision tree) is a classification-based model, meaning that in some patients whose predicted risk lie close to the cutoff point, even small reduction in dose might lead to reclassification (changing from a positive risk group to a negative). We note that this is an inherent property of these classification models and not a kink of our methodology. Nevertheless, we attempted to demonstrate the high sensitivity of the mean total lung dose to variations in input data, as illustrated in Figure 4.7. Whether such "sensitive" models should be used to guide clinical dose-based decisions, though certainly a worthy question, lies beyond the scope of the current article.

Our streamlined approach for *direct* embedding of ML models within treatment plan optimization reduces the uncertainty sources to a single model, as opposed to the two models used in (Ajdari et al. 2022), namely the NTCP model and the predictive model used to learn NTCP parameters. This reduction in complexity and reliance on surrogate models enhances the transparency and robustness of our methodology, making it a valid alternative in personalized treatment planning. Furthermore, it is crucial to recognize the potential extension of this study to different endpoints. The framework is designed to be *site- and endpoint-agnostic*, allowing its application to any endpoint of interest, as long as a predictive (outcome) model is known for that endpoint or could otherwise be learned from available datasets. Despite the advantages, our study has certain limitations.

A side-advantage of our current approach (learning mathematical representation of ML-based outcome models) is that it adds a level of transparency to the more "black-box" ML models. This in turn can enhance the users' ability to detect "nonsensical" decision rules that might have been learned by the models due to, among other factors, model overfitting, insufficient or noisy data, or misrepresentation or bias in the training dataset (Cho 2021, Belenguer 2022, Tasci et al. 2022). This is a crucial step, as prediction uncertainty and the lack of physician trust in ML-based outcome models are among the most important hurdles facing the clinical implementation of outcome-based treatment planning.

Despite its advantages, the limitations of our study must be acknowledged. First, the underlying performance of our tested ML models were not very high (AUC=0.66), most likely due to the small size of our training dataset (n=73) and the quality of the features at our disposal. Prior studies, such as that conducted by Scott et al. (2021a),

have shown that integrating additional features like genomic structures can enhance predictive outcomes. Furthermore, research by Palma et al. (2019) showed a connection between radiation-induced pneumonia and lower lung dose, suggesting potential benefits for predictive models through the inclusion of spatial information in the dataset. This added significant uncertainty to the models' predictions and the resulting risk-adapted plans. However, our primary goal here was to provide a proof-of-concept for the integration of ML-based outcome models into RT treatment planning. Our approach is cancer- and endpoint-agnostic, meaning that it can be easily applied to other, more powerful outcome models. Second, the integration of complex predictive models, such as neural networks and tree ensembles into the treatment planning process increases the computational complexity of the optimization problem, making it difficult to solve in a reasonable timeframe for large optimization models with numerous constraints and (binary) decision variables. Moreover, the use of learned constraints for specific types of toxicities may impact the optimal treatment plan, potentially at the expense of other organs that are not considered in the optimization model. Our feasibility study showed one patient showed an increased dose to the spinal cord to meet the RP2+ threshold. In the case that specific organs might be more susceptible to increased dose delivery, *e.g.* the spinal cord, we can include additional (learned) constraints in the optimization model.

4.5 Conclusion

In this study, we have illustrated the use of the optimization with constraint learning framework to tailor radiotherapy treatment plans based on individually predicted risks of side effects. Our methodology is showcased through its application to non-small cell lung cancer patients, where we employ various predictive models to learn the risk of radiation-induced pneumonitis. The learned function is then incorporated as a constraint within the treatment plan optimization model. This work can be seen as a proof-of-concept for the future development of personalized treatment planning models. The objective of our work is to establish a bridge between prediction and prescription proposing a common framework for both ML and optimization communities within the domain of RT. By effectively integrating predictive insights into the optimization process, our contribution has the potential to replace the current one-size-fits-all methodologies in the RT field.

APPENDIX

A Clinical Study: size of the optimization model

The computational burden associated with solving the optimization problems in Section 3 of the clinical study depends on the number of decision variables and constraints, and consequently, on the number of beamlets and the number of voxels in the significant volumes. Table 5 provides information on beamlet and voxel sizes for each patient.

Patient	Beamlets	Voxels				
		GTV	PTV	Lung	Heart	Spinal Cord
1	4094	2426	7108	19766	2927	370
2	2943	1880	7438	30842	3424	384
3	6578	4070	11293	28659	7017	538
4	3040	1406	8320	43910	2847	431

Table 5. Number of beamlets and voxels per structure for each patient used in the Clinical Study in Section 3. GTV stands for Gross Tumor Volume; PTV stands for Planning Target Volume.

Chapter 5

Counterfactual Explanations Using Optimization With Constraint Learning

5.1 Introduction

Interpretability in ML is an ongoing research field that has received increasing attention in recent years. Off the many approaches and tools for interpretability, counterfactual explanations (CEs) are expected to be especially promising due to their resemblance to how we provide explanations in everyday life (Miller 2018). It has been established that we do not seek to explain the cause of an event *per se*, but *relative* to some other event that did not occur. Typically, we have a factual instance vector $\hat{x}$ for which the (prediction) outcome $\hat{y}$ relative to some other, desired, outcome $\tilde{y}$ should be explained. The key idea for generating a CE is to find a data point $\tilde{\boldsymbol{x}}$ close to the factual instance $\hat{\boldsymbol{x}}$, such that the prediction outcome for $\tilde{\boldsymbol{x}}$ is $\tilde{y}$. The difference in the features constitutes the explanation. As CEs do not try to explain all possible causes of an event but focus on necessary changes to the environment to reach a certain state, they tend to be simpler, and with that, also easier to understand than those methods which communicate explanations based on the entire feature space (Miller 2018).

Wachter et al. (2018) are the first to propose an optimization-based approach for generating CEs. Having a trained classifier $h(\cdot)$, the aim is to find at least one CE, say $\tilde{\boldsymbol{x}}$, which has the closest distance to the original factual instance $\hat{\boldsymbol{x}}$ such that $h(\tilde{\boldsymbol{x}})$ is equal to a different target $\tilde{y}$. Such a CE can be obtained by solving the following mathematical optimization model:

$$\min_{\boldsymbol{x}} \max_{\lambda} \lambda(h(\boldsymbol{x}) - \tilde{y})^2 + d(\hat{\boldsymbol{x}}, \boldsymbol{x}), \tag{5.1}$$

where $d(\cdot, \cdot)$ is a distance function and λ acts as a nonnegative balancing weight to ensure $h(\boldsymbol{x}) = \tilde{y}$. Much work has been devoted to refine this problem such that the generated CEs are useful and attainable in practice. From the literature (*e.g.*, Verma et al. 2020, Wachter et al. 2018, Russell 2019, Navas-Palencia 2021, Mothilal et al. 2020), we can identify the following eight criteria that a generated CE should fulfill in both theory and practice: **Proximity:** The CE should be as close as possible to the factual instance $\hat{\boldsymbol{x}}$ with respect to the feature values. **Validity:** The prediction for the CE $\tilde{\boldsymbol{x}}$

should be equal to $\tilde{y} \neq \hat{y}$. **Coherence.** When one-hot encoding is used for categorical data, we should be able to map it back to the input feature space to obtain coherent explanations. **Sparsity:** The CE should differ from the factual instance in as few features as possible. **Actionability:** We can distinguish between immutable, mutable but not actionable, and actionable features. **Data manifold closeness:** To ensure the generation of realistic and actionable explanations, the generated CEs should be close to the observed (training) data. **Causality:** Any (known) causal relationships in the data should be respected in the proposed CEs to further ensure realistic explanations. **Diversity:** Any algorithm for the generation of CEs should return a set of CEs which differ in at least one feature.

These criteria have been partially addressed in recent work, see Table 5.1. For example, Russell (2019) and Ustun et al. (2019) address coherence and actionability, the latter introducing the notion of immutable, conditionally immutable and mutable features. Further, Russell (2019) focuses on diversity and suggests adding constraints greedily by restricting the state of variables altered in previously generated CEs, while Mothilal et al. (2020) base their approach to diverse CEs on determinantal point processes (Kulesza 2012). Kanamori et al. (2020) attempt to optimize the idea of proximity and data manifold closeness using *Mahalanobis' distance* and the *local outlier factor* to generate CEs close to the empirical distribution of the training data. Poyiadzi et al. (2020) base their work on graph theory, and apply a shortest path algorithm to minimize the f-distance quantifying the trade-off between the path length and the density along this path, by that ensuring a solution that lies in a high density region. To address causality, Kanamori et al. (2021) discuss the use of a structural causal model (SCM), while others advocate a post-hoc filtering approach (Mothilal et al. 2020). We refer to Verma et al. (2020) and Guidotti (2022) for an extensive overview of recent works on counterfactual explanations.

To the best of our knowledge, ours is the first work that addresses all of these criteria in a combined setting. We propose `CE-OCL`, a generic and flexible approach for generating CEs based on optimization with constraint learning (OCL). OCL is a new and fast-growing research field whose aim is to learn parts of an optimization model (*e.g.*, constraints or objective function) using ML models whenever explicit formulae are not available (see Fajemisin et al. (2024) for a recent survey on OCL). We show how all the criteria proposed in the literature can be addressed by an OCL framework. Based on the concept of trust regions, we also propose a new modeling approach to ensure data manifold closeness and coherence. Finally, we propose using incumbent solutions to obtain diverse CEs in a single execution. With our extensive demonstration on standard datasets from the CE literature, we also set new benchmarks for future research.

Table 5.1. State-of-the-art methods to generate CEs

	Proximity	Sparsity	Coherence	Actionability	Data Manifold Closeness	Causality	Diversity
Laugel et al. (2017)	●	●	●	–	–	–	–
Russell (2019)	●	◐	●	–	–	–	●
Ustun et al. (2019)	●	●	●	●	–	–	–
Kanamori et al. (2020)	●	–	●	–	●	–	–
Mahajan et al. (2019)	●	–	●	◐	●	●	–
Karimi et al. (2020b)	●	–	●	–	–	●	–
Kanamori et al. (2021)	●	●	●	●	–	●	●
Mothilal et al. (2020)	●	◐	●	●	–	◐	●
Karimi et al. (2020a)	●	●	●	●	–	–	●
Poyiadzi et al. (2020)	●	–	●	●	●	–	–
`CE-OCL`	●	●	●	●	●	●	●

●: addressed; ◐: partially addressed; –: absent

5.2 Generation of counterfactual explanations

In an OCL framework, ML models are used to design constraint and objective functions of an optimization model when explicit expressions are unknown. First, the predictive model is trained on historical data and then it is embedded into the optimization model using decision variables as inputs (Biggs et al. 2021, Verwer et al. 2017, Villarrubia et al. 2018). Although the interplay between optimization and ML has a different aim in OCL than CE generation, we notice that the two frameworks have a similar structure. Recent advances in OCL successfully reduce the computational burden of embedding fitted ML models into an optimization model (Grimstad and Andersson 2019, Schweidtmann and Mitsos 2019, Mišić 2020) and can be easily transferred to the problem of generating CEs. In this regard, we show how the problem of generating CEs, given a fitted model $h(\cdot)$, a factual instance $\hat{\boldsymbol{x}}$, and the desired outcome $\tilde{y}$, can be seen as a special case of *optimization with constraint learning*. In an OCL setting, a dataset $\mathcal{D} = \{(\bar{\boldsymbol{x}}_{\boldsymbol{i}}, \bar{y}_i)\}_{i=1}^N$ with observed feature vector $\bar{\boldsymbol{x}}_{\boldsymbol{i}}$ and outcome of interest $\bar{y}_i$ for sample i, is used to train predictive models that are to be constrained or optimized in a larger optimization problem. An OCL model is typically presented as

$$\max_{\boldsymbol{x}\in\mathbb{R}^n, y\in\mathbb{R}} f(\boldsymbol{x}, y) \tag{5.2}$$

$$\text{s.t. } \boldsymbol{g}(\boldsymbol{x}, y) \leq \mathbf{0}, \tag{5.3}$$

$$y = h(\boldsymbol{x}), \tag{5.4}$$

$$\boldsymbol{x} \in \mathcal{X}, \tag{5.5}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the decision vector with components $x_i \in \mathbb{R}$, $f(\cdot,\cdot) : \mathbb{R}^{n+1} \mapsto \mathbb{R}$ and $\boldsymbol{g}(\cdot,\cdot) : \mathbb{R}^{n+1} \mapsto \mathbb{R}^m$ are known functions possibly also depending on the predicted outcome y, and $h(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ represents the predictive model[1] trained on $\mathcal{D}$. The set $\mathcal{X}$ defines the trust region, *i.e.*, the set of solutions for which we trust the embedded

[1]To simplify our exposition, we include only one predictive model. However, a general OCL framework admits multiple learned constraints in the model.

predictive models (see below for details). Formulation (5.2-5.5) is quite general and encompasses a large body of work that includes CE generation. Now, we characterize the parallelism between some of the eight criteria listed in Section 5.1 and the structure of the resulting OCL model. We elaborate and discuss the remaining criteria in Appendix A.

Validity. While the trained model $h(\cdot)$ is used in constraint learning to define, completely or partially, the objective function and/or the constraints, in CE generation it is used to enforce the validity constraint. Constraint (5.4) is likely to be an encoding of the predictive model. In other words, embedding a trained ML model requires adding multiple constraints and auxiliary variables. When $h(\cdot)$ is a classification model, the CE validity is obtained by constraining the model prediction to be equal to the desired class $\tilde{y}$; that is, we set $y = \tilde{y}$. If $h(\cdot)$ is a regression model, the OCL framework still applies, and an inequality constraint can be used to enforce validity; *e.g.*, $y \leq \tilde{y} - \delta$ or $y \geq \tilde{y} + \delta$ for some fixed $\delta \in \mathbb{R}_+$.

Data manifold closeness. One of the requirements to obtain plausible CEs is that they are close to the data manifold. For this purpose, we can make use of the *trust region* constraints. Maragno et al. (2023) define the trust region as the convex hull (CH) of $\mathcal{D}$ in the features space, and they use it in OCL to prevent the trained model from extrapolating, therefore, mitigating the deterioration in predictive performance for points that are farther away from the data points in $\mathcal{D}$. In CE generation, the trust region, or rather *data manifold region*, serves the purpose of ensuring solutions in a high-density region. To this end, we can also denote a CE $(\tilde{\boldsymbol{x}})$ as the convex combination of samples in $\mathcal{D}$, in particular samples belonging to the desired class $(\tilde{y})$.

In case the CH is too restrictive, we can use a relaxed formulation to enlarge the data manifold by including those solutions that are in the ϵ-ball surrounding some feasible solutions in the CH:

$$\epsilon\text{-CH} = \left\{ \boldsymbol{x} \middle| \sum_{i \in \mathcal{I}} \lambda_i \bar{\boldsymbol{x}}_i = \boldsymbol{x} + \boldsymbol{s},\ \sum_{i \in \mathcal{I}} \lambda_i = 1,\ \boldsymbol{\lambda} \geq 0,\ ||\boldsymbol{s}||_p \leq \epsilon \right\}, \tag{5.6}$$

where $\lambda_i \in [0, 1]$ and $\boldsymbol{s} \in \mathbb{R}^n$ are auxiliary variables, $\epsilon \geq 0$ is a hyperparameter, and $\mathcal{I}$ denotes the indices corresponding to the subset of samples in $\mathcal{D}$ belonging to the desired class $\tilde{y}$. When $\epsilon = 0$, we obtain the trust region as discussed in Maragno et al. (2023). However, $\epsilon > 0$ leads to a less restrictive set of conditions. Further details are available in Appendix A as well as a graphical representation of the data manifold closeness in Figure 2.

Causality. CEs might be inefficient or unrealistic when causal relations are not considered in the generation process. Both these situations are exemplified in Karimi

et al. (2020b), where the authors show the importance of causal relations to obtain CEs that better answer the question "What *should be done* in the future considering the laws governing the world?" When a causal model is available, we can formulate the causal relations among variables as extra constraints of the optimization model. When there is not an explicit formulation of the causal relations, we are in a typical constraint learning scenario where an ML model can be trained and embedded into the optimization. We provide the formulation of causality constraints in Appendix A.

Diversity. Most of the methods for generating multiple and diverse CEs in the literature require multiple runs and extra constraints to generate diverse CEs for the same input. Following an iterative approach, we can generate diverse CEs using constraints on the actionability of features (Russell 2019), or constraints on the distance between the subsequent CE and all the previously generated ones (Karimi et al. 2020a). Again in an iterative way, we can also use the data manifold constraints to generate diverse CEs (i) by finding one CE for each clustered CH, (ii) by enlarging the CH with increasing ϵ whenever the data manifold constraints are active. The use of diversity constraints offers great flexibility at the expense of computation time. As an alternative, we propose to solve one single optimization model and use the pool of *incumbent solutions* as the set of CEs. In mixed-integer optimization, solvers like Gurobi or CPLEX allow retrieving the sub-optimal solutions found during the tree search procedure (Gurobi Optimization, LLC 2022, CPLEX, IBM ILOG 2009). In this way, collecting a set of CEs comes at no cost in terms of computation time.

5.3 Experiments and results

In this section, we demonstrate the effectiveness of OCL through empirical experiments on multiple datasets and comparing the results with other state-of-the-art methods. The experiments are executed using `OptiCL` [2] (Maragno et al. 2023), an open-source Python package for optimization with constraint learning. `OptiCL` has been originally designed to help practitioners in modeling an optimization problem whose constraints are partially unknown, but where ML models can be deployed to learn them (Maragno et al. 2023). However, as detailed in Section 5.2, the problem of generating CEs directly relates to an OCL problem. `OptiCL` currently supports several MIO-representable predictive models, including logistic regression (lr), support vector machines (svm), (optimal) decision trees (cart), random forests (rf), gradient boosting machines (gbm), and neural networks with ReLU activation functions (MLP). Moreover, `OptiCL` allows for trust region constraints as defined in (5.6). Whenever a causal model is available but the relations are not explicit, `OptiCL` allows representing the relation using one of the MIO-representable ML models. The open-source implemen-

[2] https://github.com/hwiberg/OptiCL, under the MIT license.

tation for reproducing all our results is available at
`https://github.com/tabearoeber/CE-OCL`.

We performed an extensive comparison of our method against four state-of-the-art methods: `Growing Spheres (GS)` (Laugel et al. 2017), `FACE` (Poyiadzi et al. 2020), `Actionable Recourse (AR)` (Ustun et al. 2019), and `DiCE` (Mothilal et al. 2020). Here, we present the results on the COMPAS dataset[3]. We generated CEs for 30 factual instances, then averaged the scores on the evaluation metrics proposed by Mothilal et al. (2020). Our results are presented in Figure 5.1 (see Appendix B for details on the evaluation metrics). For the sake of clarity, we have rescaled values such that they range from 0 (worst) to 1 (best). Since the majority of the other methods do not generate a set of CEs, we chose to generate only one CE for each factual instance and hence do not report any diversity scores (Figure 5.1 left). Furthermore, we compare our approach in terms of diversity by generating three CEs for each of the 30 factual instances and compare the results with `DiCE` (Mothilal et al. 2020) (Figure 5.1 right). Extensive results for all datasets and with different predictive models are included in Appendix B. We further demonstrate the generation of CEs in a step-wise man-

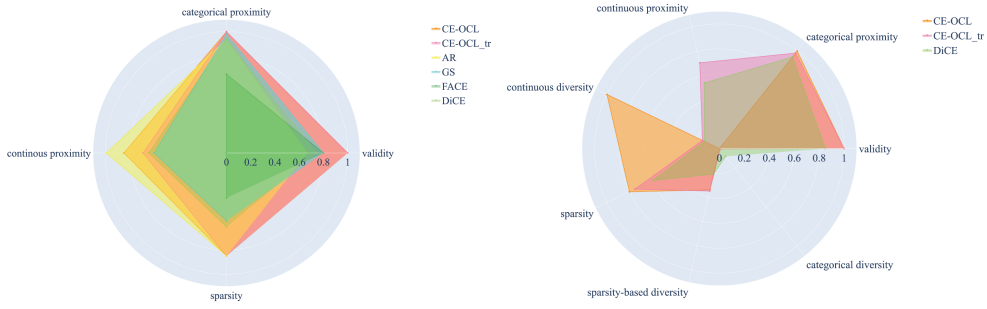


Figure 5.1. Performance of `CE-OCL` and `CE-OCL_tr` (with trust region) compared to performance of current state-of-the-art methods for generating counterfactual explanations on the COMPAS dataset. Left: we generated one counterfactual for each of 30 factual instances; rf as predictive model. Right: we generated three counterfactuals for the same instances; lr as predictive model. Not all methods support generating several counterfactuals.

ner on the Statlog (German Credit Data) dataset (Dua and Graff 2017), which is one of the standard datasets in the CE literature[4]. The German Credit dataset classifies people described by a set of 20 features as good or bad credit risk, see Table 4 in Appendix C.1 for an overview of the features. For this demonstration, we gradually add constraints to the model and present the generated CEs at each step in Table 5 shown in Appendix C.1. The table is divided into six parts (A-F), each showing the set of CEs generated, and a dash is used to represent no change to the corresponding features.

[3]Appendix B includes the results for three further datasets: Adult, Give Me Some Credit, and HELOC.
[4]We also provide another demonstration on the Statlog (Heart) dataset (Dua and Graff 2017) in Appendix C.

In Table 5 in Appendix C.1, we present the evaluation of these CEs using several evaluation metrics proposed by Mothilal et al. (2020): validity, sparsity, categorical and continuous proximity, categorical and continuous diversity, and sparsity-based diversity. The complete mathematical model is detailed in Appendix C.2.

We fit several ML models to the data, all of which performed similarly well. For demonstration purposes, we have chosen a linear support vector machine. The factual instance $\hat{\boldsymbol{x}}$ used for this case study is reported in Table 5. We start the demonstration considering only validity, proximity, and coherence (Part A), and using the ℓ_2-norm as a distance function. The optimal solution suggests several changes in the factual instance and is not actionable in practice due to the negative value for F2 (credit amount). To induce sparsity (Part B), we use auxiliary variables to keep track of the number of features changed and penalize them in the objective function. Multiple and diverse CEs are generated using incumbent solutions (Part C). To ensure that the set of generated CEs is valuable in practice, we add actionability constraints (Part D). Respecting these constraints, the set of generated CEs seems more realistic however, they may still not be attainable in practice. Specifically, if we consider solution (c) of Part D, the only suggested change concerns F4 (age). However, this CE is unlikely to represent a realistic data point, considering the other feature values remain unchanged. In other words, CEs that do not resemble the training data come with the risk of being unattainable in practice. To this end, we use the idea of a *data manifold region*, as detailed in Section 5.2. As a result, in Part E, we obtain a more realistic set of CEs, although at the expense of sparsity and (categorical) proximity (see the scores reported in Table 5, Appendix C.1). From a qualitative point of view, the three CEs show a more sensible combination of feature values compared to those in Part D. Finally, we can leverage the partial SCM provided by Karimi et al. (2020b) for this dataset, which shows that F1 (duration) is causally related to F2 (credit amount). This relationship is learned by an MLP using 5-fold cross validation. In Part F, we display the set of CEs that satisfy also the learned causality constraints.

5.4 Discussion

With this work, we propose `CE-OCL`, a generic approach for generating sensible and practical counterfactual explanations. In Section 5.3, we report the generally superior performance achieved by `CE-OCL` compared to other popular methods. Nevertheless, we acknowledge the limitations of using incumbent solutions as multiple counterfactuals caused by the lack of control over the solutions' diversity. Whenever we have specific diversity requirements to meet, the iterative approaches proposed by Russell (2019) and Karimi et al. (2020a) may suit best. Moreover, owing to the MIO structure of `CE-OCL` and various constraints used to satisfy the established criteria, the feasibility space may shrink to the point of being empty, making the optimization problem infea-

sible. In the infeasibility case, we recommend following an approach similar to that presented in Section 5.3, where constraints are added one at a time. Infeasibility problems due to data manifold constraints can be mitigated by enlarging the data manifold region at the (potential) expense of the sensibility of the CEs. For future research, we plan to investigate the effect of clustering and enlargement of the data manifold region on the CE quality and on diversity. We also intend to extend `CE-OCL` with additional criteria like robustness in the sense that the generated CEs are not point solutions, but that they are defined by ranges in the feature values.

APPENDIX

A Generating counterfactuals

In Section 5.2, we describe how criteria such as validity, closeness, causality, and diversity can be fulfilled exploiting OCL components. Likewise, other criteria can be mathematically represented in the following way:

Proximity. By definition, a CE has to be in the proximity of the factual instance according to some user-defined distance function. To obtain a CE $\tilde{\boldsymbol{x}}$ in the proximity of $\hat{\boldsymbol{x}}$, we can write the objective function (5.2) as a distance function $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$. In the literature, this function is represented by ℓ_1-norm, ℓ_2-norm, or as the Mahalanobis' distance.

Coherence. When one-hot encoding is used to deal with categorical features, we can use the constraints proposed by Russell (2019) to obtain coherent CEs. That is, we write for k categorical features the following constraints:

$$\sum_{j' \in \mathcal{C}_j} x_{j'} = 1, \;\; j = 1, \ldots, k, \tag{7}$$

where $\mathcal{C}_j$ is a set of indices referring to the dummy (binary) variables used to represent the categorical feature j. The use of a data manifold region (with a sufficiently small ϵ) has an interesting impact on CE coherence because constraints (7) become redundant. To exemplify how data manifold constraints guarantee coherence, we consider a set of samples represented by the set of indices $\mathcal{I}$, and a categorical feature *diet* that can assume only three values: *vegan*, *vegetarian*, or *omnivore*. We use one-hot encoding to replace the feature *diet* and describe a CE with the dummy (binary) variables x_{vegan}, $x_{vegetarian}$, $x_{omnivore}$. From (5.6), we have

$$x_j = \sum_{i \in \mathcal{I}} \lambda_i \bar{x}_{i,j}, \;\; j \in \{vegan, vegetarian, omnivore\}, \tag{8}$$

with $\sum_{i \in \mathcal{I}} \lambda_i = 1$. One of the dummy variables, say x_{vegan}, can assume value 1 only if it is the convex combination of data points $\bar{\boldsymbol{x}}_i$ with $\bar{x}_{i,vegan} = 1$ and $\bar{x}_{i,vegeterian} = \bar{x}_{i,omnivore} = 0$. Thus, $\lambda_i > 0$ only when $\bar{x}_{i,vegan} = 1$, and consequently, we obtain $x_{vegetarian} = x_{omnivore} = 0$.

Sparsity. The sparsity can be handled by enforcing the following set of constraints:

$$|x_j - \hat{x}_j| \leq M z_j, \;\; j = 1, \ldots, n, \tag{9}$$

$$\sum_{i=1}^{n} z_i \leq K, \tag{10}$$

where $z_j \in \{0,1\}, j = 1,\ldots,n$ are auxiliary variables that are simply used to count the number of features in $\boldsymbol{x}$ that differ from $\hat{\boldsymbol{x}}$, and K is an upper bound on the number of allowed changes. Alternatively, constraints (10) can be relaxed and moved to the objective function with a scaling penalty factor $\alpha > 0$. That is, we obtain the new objective function $f(\boldsymbol{x}, y) + \alpha \sum_{i=1}^{n} z_i$. Though simpler, this relaxation does not guarantee to lead to an optimal solution with less than or equal to K changes.

Actionability. As a recommended CE should never change the immutable features, we can restrict the CE to be equal to the factual instance for all the immutable features. Suppose that the set of immutable features is represented by $\mathcal{I}_m$, then we simply add the following constraints:

$$x_i = \hat{x}_i, \;\; i \in \mathcal{I}_m. \tag{11}$$

Other feasibility constraints might concern actionable variables that cannot take certain values, such as *age*, which can only be increased, or *has_phd*, which can only change from false to true. These conditions can be added exactly like immutable features.

Data manifold region. Figure 2 shows how the CH of $\mathcal{D}$ in the features space ensures a solution closer to the data manifold, leading to more plausible CEs. In some cases, the CH may be too restrictive, which is why we introduce formulation 5.6 to enlarge the data manifold region by including solutions that are in the ϵ-ball around some feasible solutions in the CH. Being able to enlarge the data manifold region represents a solution to the criticism by Balestriero et al. (2021): "[...] interpolation[5] almost surely never occurs in high-dimensional spaces (> 100) regardless of the underlying intrinsic dimension of the data manifold." Aside from the bound on the norm of $\boldsymbol{s}$, all constraints in (5.6) are linear. Fortunately, the most common norms used to constraint $\boldsymbol{s}$ are ℓ_1-, ℓ_2-, or ℓ_∞-norm. These norms lead to convex conic constraints that can be handled easily with off-the-shelf optimization solvers. The effectiveness of the data manifold region might be hampered by the fact that the CH includes low-density regions. In this case, Maragno et al. (2023) advocate a two-step approach: first, clustering is used to identify distinct high-density regions, and then, the data manifold region is represented as the union of the (enlarged) convex hulls of the individual clusters.

Causality The causality constraints are modelled by applying the Abduction-Action-Prediction steps (Pearl 2013), Karimi et al. (2020b) define the endogenous variables

[5]Interpolation occurs for a sample $\boldsymbol{x}$ whenever this sample belongs to the CH of a set of data points.

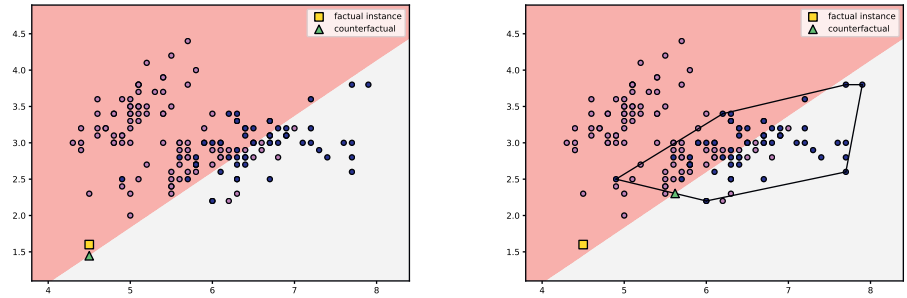


Figure 2. The effect of the data manifold region on the generated CE. The left figure shows the factual instance and its closest counterfactual without closeness constraints. The right figure shows the same factual instance with the CE constrained to be within the data manifold region.

(with indices in the set $\mathcal{E}$) as

$$x_i = \hat{x}_i + c_i(\boldsymbol{p}_i) - c_i(\hat{\boldsymbol{p}}_i), \;\; i \in \mathcal{E}, \tag{12}$$

where $c_i(\boldsymbol{p}_i)$ is a function of the parents of x_i, namely the predecessors of the feature i in the SCM. Both $\hat{x}_i$ and $c_i(\hat{\boldsymbol{p}}_i)$ are known before the optimization and therefore treated as parameters. When there is not an explicit formulation of $c_i(\cdot)$, we are in a constraint learning scenario where an ML model can be trained and embedded into the optimization as $c_i = h_i(\boldsymbol{p}_i)$ for all $i \in \mathcal{E}$.

B Comparison against other methods

We compared `CE-OCL` to four open-source tools for generating CEs: `Growing Spheres` Laugel et al. (2017), `FACE`Poyiadzi et al. (2020), `Actionable Recourse` Ustun et al. (2019), and `DiCE` Mothilal et al. (2020). The experiments are performed using `CARLA` Pawelczyk et al. (2021), a Python library to benchmark counterfactual explanation and recourse models. The predictive model used in the experiments is a random forest and the evaluation is performed by generating a counterfactual for 30 different factual instances on four datasets available in `CARLA`: Adult, Give Me Some Credit, COMPAS, and HELOC. We average the results for the evaluation metrics proposed by Mothilal et al. (2020) and present them together with the standard error (s.e.) in Table 2. Validity, sparsity, categorical proximity, categorical diversity, and sparsity-based diversity range in the interval [0,1], where 0 and 1 represent the worst and the best scores ($\uparrow_0^1$), respectively. Continuous diversity is a positive number, and the higher it is, the better ($\uparrow_0^+$). Continuous proximity is a negative number, and the closer it is to 0, the better

$(\uparrow^0_-)$.

While CE-OCL can deal with causality and closeness constraints, this does not apply to DiCE which uses a post-hoc filtering approach to remove unrealistic CEs. In addition to causality and closeness constraints, Actionable Recourse, and Growing Sphere cannot generate more than one counterfactual for each instance. FACE does not support diversity and causality constraints but it is able to generate CEs close to the data manifold region. Therefore, in Table 2 we report both the results obtained with CE-OCL including validity, proximity, coherence, sparsity, and immutability constraints, and the results obtained including also the closeness constraints, CE-OCL_tr. The results show that, across all datasets, both CE-OCL and CE-OCL_tr exhibit better performance in terms of validity, categorical proximity, and sparsity. Actionable Recourse and CE-OCL/CE-OCL_tr perform equally well in terms of continuous proximity.

Table 2. Comparison of CE-OCL with DiCE (genetic), Algorithmic Recourse, Growing spheres, and FACE using Random Forest as predictive model.

		validity $(\uparrow^1_0)$ mean (s.e.)	**cat. proximity** $(\uparrow^1_0)$ mean (s.e.)	**cont. proximity** $(\uparrow^0_-)$ mean (s.e.)	**sparsity** $(\uparrow^1_0)$ mean (s.e.)
ADULT	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	-4844.35 (575.93)	**0.93** (0.00)
	CE-OCL_tr	**1.00** (0.00)	0.97 (0.02)	-21785.50 (7506.35)	0.86 (0.01)
	DiCE	**1.00** (0.00)	0.74 (0.03)	-84278.61 (11613.30)	0.50 (0.02)
	Actionable Recourse	**1.00** (0.00)	0.78 (0.07)	**0.00** (0.00)	0.89 (0.04)
	Growing Spheres	0.80 (0.07)	0.95 (0.01)	-78901.08 (10395.08)	0.59 (0.01)
	FACE	0.80 (0.07)	0.65 (0.03)	-108614.33 (18804.05)	0.47 (0.02)
COMPAS	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	-15.23 (5.84)	**0.85** (0.01)
	CE-OCL_tr	**1.00** (0.00)	1.00 (0.00)	-30.83 (16.00)	**0.85** (0.01)
	DiCE	0.74 (0.09)	0.94 (0.03)	-35.58 (9.59)	0.61 (0.01)
	Actionable Recourse	0.67 (0.11)	0.94 (0.03)	**-0.87** (0.10)	**0.85** (0.01)
	Growing spheres	0.80 (0.07)	0.98 (0.02)	-39.88 (6.58)	0.56 (0.01)
	FACE	0.80 (0.07)	0.65 (0.05)	-98.83 (21.18)	0.37 (0.03)
HELOC	CE-OCL	**1.00** (0.00)	–	**-12.21** (2.67)	**0.94** (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-92.71 (13.75)	0.75 (0.02)
	DiCE	0.97 (0.03)	–	-203.83 (13.70)	0.22 (0.02)
	Actionable Recourse*	–	–	–	–
	Growing spheres	0.77 (0.08)	–	-87.36 (13.16)	0.00 (0.00)
	FACE	0.77 (0.08)	–	-361.80 (25.20)	0.17 (0.02)
CREDIT	CE-OCL	**1.00** (0.00)	–	**-1.18** (0.66)	**0.90** (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-120.61 (118.54)	0.87 (0.01)
	DiCE	**1.00** (0.00)	–	-1618.33 (305.53)	0.25 (0.02)
	Actionable Recourse	0.83 (0.17)	–	-8.47 (7.96)	0.88 (0.02)
	Growing spheres	0.63 (0.09)	–	-47.73 (27.24)	0.10 (0.00)
	FACE	0.63 (0.09)	–	-3001.73 (430.61)	0.11 (0.02)

For the comparison, one counterfactual was generated for each of 30 factual instances.
The scores were averaged over all instances, and the standard error was derived.
* For the Heloc dataset, Actionable Recourse did not yield any counterfactuals for any of the thirty factual instances.

We performed a more thorough comparison between CE-OCL and DiCE on the same four datasets but this time generating three CEs for each instance and using all the predictive models supported by both OptiCL and DiCE. In Table 3, we report the results obtained with CE-OCL including validity, proximity, coherence, sparsity, diversity, and actionability together with the results obtained considering also the data manifold closeness, (CE-OCL_tr). The results clearly show how CE-OCL outperforms DiCE

in terms of validity, categorical proximity, continuous proximity, and sparsity. While both methods have a categorical diversity score very close to zero in every scenario, `DiCE` has a generally better performance in terms of continuous diversity. Similarly, `DiCE` has a better sparsity-based diversity score with the exception of the COMPAS dataset. The addition of closeness constraints (`CE-OCL_tr`) has a negative effect on the sparsity and proximity scores but it positively affects the diversity scores when compared to `CE-OCL`. This was to be expected, as the data manifold region forces solutions to be located in a high-density region, which might lead to optimal solutions with more feature changes. While the sparsity decreases, this loss comes at a high potential of more valuable counterfactuals.

Table 3. Comparison of CE-OCL, CE-OCL with trust region, and DiCE (genetic) with a range of preditive models.

		validity($\uparrow^1_0$) mean (s.e.)	**cat. proximity** ($\uparrow^1_0$) mean (s.e.)	**cont. proximity**($\uparrow^0_-$) mean (s.e.)	**sparsity**($\uparrow^1_0$) mean (s.e.)	**cat. diversity** ($\uparrow^1_0$) mean (s.e.)	**cont. diversity** ($\uparrow^+_0$) mean (s.e.)	**sparsity-based diversity**($\uparrow^1_0$) mean (s.e.)
Adult dataset								
rf	CE-OCL	**1.00** (0.00)	**0.99** (0.00)	**-6775.61** (953.24)	**0.92** (0.00)	0.01 (0.01)	5796.61 (1056.70)	0.11 (0.01)
	CE-OCL_tr	**1.00** (0.00)	0.97 (0.02)	-25044.04 (7645.82)	0.86 (0.01)	0.00 (0.00)	13288.83 (3236.28)	0.13 (0.01)
	DiCE	1.00 (0.00)	0.74 (0.02)	-81581.97 (8003.12)	0.51 (0.01)	**0.19** (0.02)	**79595.54** (7496.70)	**0.26** (0.02)
lr	CE-OCL	**1.00** (0.00)	**0.99** (0.01)	**-4226.05** (794.33)	**0.89** (0.01)	0.01 (0.01)	8042.59 (1418.66)	0.19 (0.01)
	CE-OCL_tr	**1.00** (0.00)	0.96 (0.02)	-23288.53 (6837.94)	0.84 (0.01)	0.05 (0.02)	23421.06 (7376.79)	0.22 (0.01)
	DiCE	0.71 (0.05)	0.68 (0.03)	-111661.60 (10261.28)	0.47 (0.02)	**0.30** (0.03)	**110668.36** (11019.56)	**0.35** (0.02)
cart	CE-OCL	**1.00** (0.00)	0.88 (0.02)	**-13994.62** (4198.07)	0.83 (0.02)	0.21 (0.04)	26889.72 (8343.13)	**0.27** (0.03)
	CE-OCL_tr	**1.00** (0.00)	**0.94** (0.01)	-19490.98 (5725.22)	**0.84** (0.01)	0.08 (0.02)	20313.32 (5702.44)	0.19 (0.01)
	DiCE	0.65 (0.06)	0.77 (0.02)	-91507.16 (10271.17)	0.55 (0.02)	**0.23** (0.02)	**84111.77** (11802.96)	0.25 (0.01)
mlp	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	**-10553.10** (1842.00)	**0.88** (0.01)	0.00 (0.00)	2538.19 (669.37)	0.15 (0.02)
	CE-OCL_tr	**1.00** (0.00)	0.97 (0.02)	-21467.01 (7465.77)	0.86 (0.01)	0.00 (0.00)	6229.99 (2385.40)	0.14 (0.01)
	DiCE	0.62 (0.06)	0.67 (0.03)	-89314.87 (10646.04)	0.47 (0.02)	**0.26** (0.03)	**83848.59** (11476.56)	**0.31** (0.02)
gbm	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	**-2488.82** (865.72)	**0.91** (0.00)	0.00 (0.00)	4475.59 (1737.16)	0.13 (0.01)
	CE-OCL_tr	**1.00** (0.00)	0.97 (0.01)	-22732.34 (7545.41)	0.88 (0.01)	0.02 (0.01)	10609.72 (3583.71)	0.15 (0.01)
	DiCE	0.91 (0.04)	0.70 (0.02)	-113297.47 (11606.39)	0.49 (0.01)	**0.25** (0.02)	**75728.74** (10042.23)	**0.27** (0.02)
COMPAS dataset								
rf	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	**-18.79** (6.15)	**0.85** (0.00)	0.00 (0.00)	8.87 (4.30)	0.17 (0.01)
	CE-OCL_tr	**1.00** (0.00)	**1.00** (0.00)	-38.02 (16.46)	**0.85** (0.01)	0.00 (0.00)	9.41 (4.49)	0.16 (0.01)
	DiCE	0.81 (0.04)	0.96 (0.01)	-40.94 (7.06)	0.60 (0.01)	**0.06** (0.02)	**22.30** (5.78)	**0.20** (0.01)
lr	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	-121.74 (13.72)	**0.80** (0.01)	0.00 (0.00)	**229.89** (28.27)	0.34 (0.01)
	CE-OCL_tr	**1.00** (0.00)	0.98 (0.01)	**-35.84** (11.03)	0.75 (0.01)	0.01 (0.01)	34.68 (7.79)	**0.35** (0.01)
	DiCE	0.85 (0.05)	0.94 (0.02)	-56.05 (12.83)	0.59 (0.01)	**0.08** (0.02)	30.61 (9.48)	0.21 (0.02)
cart	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	**-23.14** (6.03)	**0.84** (0.01)	0.00 (0.00)	**33.38** (11.83)	**0.19** (0.01)
	CE-OCL_tr	**1.00** (0.00)	**1.00** (0.00)	-28.43 (9.16)	0.83 (0.01)	0.00 (0.00)	31.16 (10.10)	**0.19** (0.01)
	DiCE	0.77 (0.08)	0.96 (0.01)	-32.99 (6.04)	0.60 (0.01)	**0.07** (0.02)	24.31 (6.63)	0.18 (0.01)
mlp	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	**-16.55** (2.11)	0.81 (0.01)	0.00 (0.00)	16.20 (4.62)	**0.22** (0.01)
	CE-OCL_tr	**1.00** (0.00)	**1.00** (0.00)	-27.75 (10.25)	**0.82** (0.01)	0.00 (0.00)	7.29 (4.09)	0.18 (0.01)
	DiCE	0.82 (0.06)	0.96 (0.01)	-59.11 (13.01)	0.58 (0.01)	**0.06** (0.02)	**24.95** (5.78)	**0.22** (0.02)
gbm	CE-OCL	**1.00** (0.00)	**1.00** (0.00)	**-10.10** (2.60)	**0.86** (0.00)	0.00 (0.00)	13.64 (3.09)	**0.21** (0.01)
	CE-OCL_tr	**1.00** (0.00)	**1.00** (0.00)	-25.70 (10.66)	0.85 (0.01)	0.00 (0.00)	13.49 (5.08)	0.20 (0.01)
	DiCE	0.59 (0.07)	0.96 (0.01)	-42.64 (6.32)	0.60 (0.01)	**0.08** (0.02)	**24.51** (5.81)	0.20 (0.01)
Heloc dataset								
rf	CE-OCL	**1.00** (0.00)	–	**-13.53** (2.35)	**0.93** (0.00)	–	9.94 (2.74)	0.09 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-94.24 (13.68)	0.75 (0.02)	–	18.93 (4.62)	0.24 (0.02)
	DiCE	0.90 (0.03)	–	-231.05 (11.17)	0.21 (0.02)	–	**223.91** (14.16)	**0.61** (0.02)
lr	CE-OCL	**1.00** (0.00)	–	**-99.09** (14.22)	**0.88** (0.01)	–	188.16 (28.38)	0.21 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-138.29 (16.52)	0.72 (0.02)	–	72.51 (8.39)	0.34 (0.02)
	DiCE	0.70 (0.06)	–	-232.39 (12.87)	0.21 (0.02)	–	**207.02** (11.34)	**0.61** (0.02)
cart	CE-OCL	**1.00** (0.00)	–	**-13.12** (1.40)	**0.95** (0.00)	–	19.72 (2.41)	0.08 (0.00)
	CE-OCL_tr	**1.00** (0.00)	–	-99.05 (13.45)	0.73 (0.02)	–	41.03 (6.53)	0.31 (0.02)
	DiCE	0.80 (0.07)	–	-216.70 (13.47)	0.22 (0.02)	–	**234.89** (16.10)	**0.61** (0.02)
mlp	CE-OCL	**1.00** (0.00)	–	**-25.09** (7.57)	**0.92** (0.01)	–	21.30 (4.18)	0.12 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-98.94 (15.87)	0.75 (0.02)	–	15.41 (5.52)	0.26 (0.02)
	DiCE	0.67 (0.07)	–	-252.56 (14.17)	0.20 (0.02)	–	**246.96** (16.31)	**0.61** (0.02)
gbm	CE-OCL	**1.00** (0.00)	–	**-8.41** (2.45)	**0.94** (0.00)	–	16.31 (4.92)	0.10 (0.00)
	CE-OCL_tr	**1.00** (0.00)	–	-89.91 (14.70)	0.76 (0.02)	–	18.70 (6.87)	0.25 (0.02)
	DiCE	0.73 (0.08)	–	-234.96 (11.60)	0.22 (0.02)	–	**248.95** (17.34)	**0.59** (0.02)
Give me some credit dataset								
rf	CE-OCL	**1.00** (0.00)	–	**-6.77** (4.43)	**0.90** (0.00)	–	9.14 (5.53)	0.15 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-97.01 (95.21)	0.89 (0.01)	–	115.65 (113.90)	0.16 (0.01)
	DiCE	**1.00** (0.00)	–	-2166.72 (318.36)	0.23 (0.02)	–	**2446.71** (455.17)	**0.32** (0.01)
lr	CE-OCL	**1.00** (0.00)	–	**-3.79** (1.24)	**0.88** (0.01)	–	7.50 (2.49)	0.24 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-614.00 (202.97)	0.83 (0.01)	–	1107.84 (381.92)	0.25 (0.01)
	DiCE	0.92 (0.05)	–	-1946.86 (256.37)	0.21 (0.02)	–	**1909.26** (187.85)	**0.29** (0.01)
cart	CE-OCL	**1.00** (0.00)	–	-1.85 (0.23)	**0.87** (0.00)	–	1.91 (0.23)	0.17 (0.00)
	CE-OCL_tr	**1.00** (0.00)	–	-212.82 (100.91)	0.85 (0.01)	–	285.60 (121.56)	0.22 (0.01)
	DiCE	0.00 (0.00)	–	-1895.95 (230.21)	0.25 (0.02)	–	**2214.51** (319.98)	**0.32** (0.01)
mlp	CE-OCL	**1.00** (0.00)	–	**-24.21** (8.71)	**0.89** (0.00)	–	38.15 (13.75)	0.15 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-996.30 (370.04)	0.85 (0.01)	–	971.37 (447.54)	0.17 (0.01)
	DiCE	0.97 (0.03)	–	-2526.22 (265.46)	0.20 (0.02)	–	**3205.58** (427.42)	**0.32** (0.01)
gbm	CE-OCL	**1.00** (0.00)	–	**-175.98** (74.32)	**0.89** (0.01)	–	296.26 (134.93)	0.17 (0.01)
	CE-OCL_tr	**1.00** (0.00)	–	-219.19 (131.96)	0.87 (0.01)	–	123.61 (82.17)	0.16 (0.01)
	DiCE	0.93 (0.04)	–	-2222.50 (277.89)	0.22 (0.02)	–	**2749.12** (409.75)	**0.31** (0.02)

C Case studies

We reserve this appendix for the details of our case study, Statlog (German Credit Data) dataset, and for the additional demonstration on the Statlog (Heart) dataset.

C.1 German Credit Data tables

We report an overview of the Statlog (German Credit Data) dataset features and the CEs generated at each step detailed in Section 5.3 in Table 4 and Table 5, respectively.

Table 4. Information on Statlog (German Credit Data) Data Set (Dua and Graff 2017)

Label	Variable name	Description	Domain*	Constraint
F1	duration	Duration in months	real	≥ 0
F2	credit_amount	Credit amount	real	≥ 0
F3	instalment_commitment	Installment rate in percentage of disposable income	real	≥ 0
F4	age	Age in years	real	$x_{age} \geq \hat{\boldsymbol{x}}_{age}$
F5	residence_since	Present residence since X years	integer	$x_{residence\_since} \geq \hat{\boldsymbol{x}}_{residence\_since}$
F6	existing_credits	Number of existing credits at this bank	integer	≥ 0
F7	num_dependents	Number of people being liable to provide maintenance for	integer	≥ 0
F8	checking_status	Status of existing checking account, in Deutsche Mark	binary	–
F9	credit_history	Credit history (credits taken, paid back duly, delays, critical accounts)	binary	–
F10	employment	Present employment, in number of years.	binary	conditionally immutable
F11	foreign_worker	Foreign worker (yes,no)	binary	immutable
F12	housing	Housing (rent, own,...)	binary	–
F13	job	Job	binary	–
F14	other_parties	Other debtors / guarantors	binary	–
F15	other_payment_plans	Other installment plans (banks, stores)	binary	–
F16	own_telephone	Telephone (yes,no)	binary	–
F17	personal_status	Personal status (married, single,...) and sex	binary	immutable
F18	property_magnitude	Property (e.g. real estate)	binary	–
F19	purpose	Purpose of the credit (car, television,...)	binary	immutable
F20	saving_status	Status of savings account/bonds, in Deutsche Mark.	binary	–

* All categorical are one-hot encoded and therefore considered binary.

Table 5. `CE-OCL` demo on the Statlog (German Credit Data) Data Set (Dua and Graff 2017).

(a) Counterfactual explanations generated for enriching the optimization model step by step with the constraint presented in Section 5.2

	F1	F2	F3	F4	F8*	F10	F12	F14	F16	F18*	F20*
$\hat{x}$	24.0	1371.26	4.0	25.0	A	1≤X<4	rent	none	none	A	A
Part A: validity, proximity, coherence											
(a)	15.02	-333.52	3.86	27.04	–	–	–	–	–	–	–
Part B: validity, proximity, coherence, sparsity											
(a)	7.12	–	–	–	–	–	–	–	–	–	–
Part C: validity, proximity, coherence, sparsity, diversity											
(a)	7.12	–	–	–	–	–	–	–	–	–	–
(b)	–	-2873.47	–	30.06	–	–	–	–	–	–	–
(c)	–	–	1.96	26.63	–	–	–	–	–	–	–
Part D: validity, proximity, coherence, sparsity, diversity, actionability											
(a)	7.12	–	–	–	–	–	–	–	–	–	–
(b)	–	–	1.96	26.63	–	–	–	–	–	–	–
(c)	–	–	–	75.52	–	–	–	–	–	–	–
Part E: validity, proximity, coherence, sparsity, diversity, actionability, data manifold closeness											
(a)	22.0	1283.52	–	–	B	4≤X<7	–	–	–	B	–
(b)	10	1363.43	2.0	64.0	B	–	own	–	yes	C	B
(c)	12.0	1893.04	–	29.0	–	–	own	guarantor	yes	B	B
Part F: validity, proximity, coherence, sparsity, diversity, actionability, data manifold closeness, causality											
(a)	–	–	–	–	B	4¡=X¡7	–	–	–	B	–
(b)	22.0	990.51	–	–	B	4¡=X¡7	–	–	–	B	–
(c)	26.83	1910.28	–	–	B	4¡=X¡7	–	–	–	B	–

F1–F20 represent the 20 features of the dataset. See Table 4 in Appendix C for a description.
The dash (–) represents no change in a feature with respect to the factual instance.
F5, F6, F7, F9, F11, F13, F15, F17, F19: None of the counterfactual explanations proposed a change in these variables. For space reasons they are not displayed here.
* **F8:** A: ¡0, B: no checking; **F18:** A: real estate, B: life insurance, C: car ; **F20:** A: no known savings, B: ¡100

(b) Evaluation* of counterfactuals generated for a single factual instance, with constraints added gradually.

	categorical proximity($\uparrow_0^1$)	**continuous proximity**($\uparrow_-^0$)	**sparsity**($\uparrow_0^1$)	**categorical diversity**($\uparrow_0^+$)	**continuous diversity**($\uparrow_0^+$)	**sparsity-based diversity**($\uparrow_0^+$)
Part A	1.00	-1715.94	0.8	–	–	–
Part B	1.00	-16.88	0.95	–	–	–
Part C	1.00	-1423.45	0.92	0.00	2845.81	0.15
Part D	1.00	-23.69	0.93	0.00	46.29	0.12
Part E	0.67	-230.12	0.63	0.36	441.68	0.42
Part F	0.77	-308.20	0.78	0.00	616.40	0.10

Part A: validity, proximity, coherence; **Part B:** validity, proximity, coherence, sparsity; **Part C:** validity, proximity, coherence, sparsity, diversity; **Part D:** validity, proximity, coherence, sparsity, diversity, actionability; **Part E:** validity, proximity, coherence, sparsity, diversity, actionability, data manifold closeness; **Part F:** validity, proximity, coherence, sparsity, diversity, actionability. data manifold closeness, causality
* **validity** ($\uparrow_0^1$)**:** 1.00 in all cases

C.2 German Credit Data CE generation model

For the case study in Section 5.3, we made use of the Statlog (German Credit Data) dataset (Dua and Graff 2017) [6]. Table 4 provides an overview of features in this dataset, alongside a short description and the measurement level. For conciseness, we labelled the features F1-F20, and use those labels throughout the manuscript. Table 4 also displays the actionability constraints we imposed on the features. The following mathematical model is used to generate CEs and contains all the constraints – criteria – presented in Section 5.2:

$$\min_{\boldsymbol{x},\boldsymbol{z},\boldsymbol{s}\in\mathbb{R}^n,\boldsymbol{\lambda}\in\mathbb{R}^{|\mathcal{I}|}_{\geq 0}} \ell_2(\boldsymbol{x},\hat{\boldsymbol{x}}) + \alpha\sum_i z_i + \beta\ell_1(\boldsymbol{s},\tilde{\boldsymbol{s}})$$

(13, proximity, sparsity, and closeness)

$$\text{s.t. } h(\boldsymbol{x}) = 1, \qquad \text{(14, validity)}$$

$$|\boldsymbol{x} - \hat{\boldsymbol{x}}| \leq M\boldsymbol{z}, \qquad \text{(15, sparsity)}$$

$$\sum_{i\in\mathcal{I}} \lambda_i \bar{\boldsymbol{x}}_i = \boldsymbol{x} + \boldsymbol{s}, \qquad \text{(16, data manifold closeness)}$$

$$\sum_{i\in\mathcal{I}} \lambda_i = 1, \qquad \text{(17, data manifold closeness)}$$

$$x_i \geq 0, \;\; i \in \{F1, F2, F3, F6, F7\}, \qquad \text{(18, actionability)}$$

$$x_i \geq \hat{x}_i, \;\; i \in \{F4, F5\}, \qquad \text{(19, actionability)}$$

$$x_i = \hat{x}_i, \;\; i \in \{F11, F17, F19\}, \qquad \text{(20, immutability)}$$

$$x_{F10} \in \mathcal{C}_{F10}, \qquad \text{(21, conditional immutability)}$$

$$x_{F1} = \hat{x}_{F1} + h_{causality}(x_{F2}) - h_{causality}(\hat{x}_{F2}), \qquad \text{(22, causality)}$$

$$\boldsymbol{x} \in \mathcal{L}. \qquad \text{(23, Domain (real, integer, binary))}$$

[6] Preprocessed from https://datahub.io/machine-learning/credit-g.

C.3 Heart tables

Similarly to the German Credit Data case study, we report Table 6 with the CEs generated at each step and the scores for the evaluation metrics, and Table 7 with an overview of the features.

Table 6. CE-OCL demo on the Statlog (Heart) Data Set (Dua and Graff 2017)

(a) Counterfactual explanations generated for enriching the optimization model step by step with the constraint presented in Section 5.2

	age	bp	sch	mhrt	opk	chp	ecg	exian	fbs	sex	slope	thal	vessel
$\hat{x}$	49.0	130.0	265.98	171.01	0.6	atypical angina	normal	no	false	male	upsloping	normal	0
Part A: validity, proximity, coherence													
(a)	48.82	139.28	328.09	153.98	1.05	-	-	-	-	-	-	-	-
Part B: validity, proximity, coherence, sparsity													
(a)	-	-	407.24	-	-	-	-	-	-	-	-	-	-
Part C: validity, proximity, coherence, sparsity, diversity													
(a)	-	-	407.24	-	-	-	-	-	-	-	-	-	-
(b)	-	-	393.92	175.14	-	-	-	-	-	-	-	-	-
(c)	-	-	404.04	-	0.47	-	-	-	-	-	-	-	-
Part D: validity, proximity, coherence, sparsity, diversity, actionability													
(a)	-	-	407.24	-	-	-	-	-	-	-	-	-	-
(b)	-	-	393.92	175.14	-	-	-	-	-	-	-	-	-
(c)	-	-	-	124.37	-	-	-	-	-	-	-	-	-
Part E: validity, proximity, coherence, sparsity, diversity, actionability, data manifold closeness													
(a)	-	111.77	253.81	152.7	0.0	nonanginal pain	-	-	-	-	-	-	-
(b)	-	137.0	258.5	147.01	1.55	asymptomatic	left ventricular hypertrophy	-	-	-	flat	reversible defect	-
(c)	-	140.61	274.7	128.61	0.49	asymptomatic	left ventricular hypertrophy	yes	-	-	-	reversible defect	-

See Table 7 for a description of each feature
The dash (–) represents no change in a feature with respect to the factual instance.

(b) Evaluation* of counterfactuals generated for a single factual instance, with constraints added gradually.

	categorical proximity$(\uparrow_0^1)$	**continuous proximity**$(\uparrow_-)$	**sparsity**$(\uparrow_0^1)$	**categorical diversity**$(\uparrow_0^+)$	**continuous diversity**$(\uparrow_0^+)$	**sparsity-based diversity**$(\uparrow_0^+)$
Part A	1.00	-89.05	0.62	-	-	-
Part B	1.00	-141.26	0.92	-	-	-
Part C	1.00	-137.17	0.87	0.00	11.72	0.18
Part D	1.00	-106.66	0.90	0.00	128.02	0.15
Part E	0.62	-50.19	0.46	0.42	50.25	0.56

Part A: validity, proximity, coherence; **Part B:** validity, proximity, coherence, sparsity; **Part C:** validity, proximity, coherence, sparsity, diversity; **Part D:** validity, proximity, coherence, sparsity, diversity, actionability; **Part E:** validity, proximity, coherence, sparsity, diversity, actionability, data manifold closeness; **Part F:** validity, proximity, coherence, sparsity, diversity, actionability. data manifold closeness, causality
*__validity__ $(\uparrow_0^1)$**:** 1.00 in all cases

Table 7. Information on Statlog (Heart) Data Set (Dua and Graff 2017)

Variable name	Description	Domain*	Constraint
age	Patient age in years	real	immutable
sex	Gender	binary	immutable
chp	Chest pain type	binary	-
bp	Resting blood pressure	real	≥ 0
sch	Serum cholesterol	real	≥ 0
fbs	Fasting blood sugar >120 mg/dL	binary	-
ecg	Resting electrocardiographic result	binary	-
mhrt	Maximum heart rate	real	≥ 0
exian	Exercise induced angina	binary	-
opk	Old peak	real	≥ 0
slope	Slope of peak exercise ST segment	binary	-
vessel	Number of major vessels	binary	-
thal	Defect type	binary	-

* All categorical are one-hot encoded and therefore considered binary.

C.4 Heart CE generation model

The following mathematical model is used to generate CEs and contains all the constraints – criteria – presented in the Section 5.2:

$$\min_{\boldsymbol{x},\boldsymbol{z},\boldsymbol{s}\in\mathbb{R}^n,\boldsymbol{\lambda}\in\mathbb{R}^{|\mathcal{I}|}_{\geq 0}} \ell_2(\boldsymbol{x},\hat{\boldsymbol{x}}) + \alpha\sum_i z_i + \beta\ell_1(\boldsymbol{s},\tilde{\boldsymbol{s}})$$

(24, proximity, sparsity, and closeness)

$$\text{s.t. } h(\boldsymbol{x}) = 1, \qquad \text{(25, validity)}$$

$$|\boldsymbol{x} - \hat{\boldsymbol{x}}| \leq M\boldsymbol{z}, \qquad \text{(26, sparsity)}$$

$$\sum_{i\in\mathcal{I}} \lambda_i \bar{\boldsymbol{x}}_i = \boldsymbol{x} + \boldsymbol{s}, \qquad \text{(27, data manifold closeness)}$$

$$\sum_{i\in\mathcal{I}} \lambda_i = 1, \qquad \text{(28, data manifold closeness)}$$

$$x_i \geq 0, \;\; i \in \{bp, sch, mhrt, opk\}, \qquad \text{(29, actionability)}$$

$$x_i = \hat{x}_i, \;\; i \in \{age, sex\}, \qquad \text{(30, immutability)}$$

$$\boldsymbol{x} \in \mathcal{L}. \qquad \text{(31, Domain (real, binary))}$$

The predictive model used for this demo is a neural network with one hidden layer of 50 nodes and ReLU activation functions. A description of the Statlog (Heart) dataset used in the experiment is given in Table 7. The experiments have the same structure described in Section 5.3, and the results are reported in Table 6.

Chapter 6

Finding Regions of Counterfactual Explanations via Robust Optimization

6.1 Introduction

Counterfactual explanations, also known as algorithmic recourse, are becoming increasingly popular as a way to explain the decisions made by black-box ML models. Given a factual instance for which we want to derive an explanation, we search for a counterfactual feature combination describing the minimum change in the feature space that will lead to a flipped model prediction. For example, for a person with a rejected loan application, the CE could be "if the *annual salary* would increase to 50,000$, then the *loan application* would be approved." This method enables a form of user agency and is therefore particularly attractive in consequential decision making, where the user is directly and indirectly impacted by the outcome of the ML model.

The first optimization-based approach to generate CEs has been proposed by Wachter et al. (2018). Given a trained classifier $h : \mathcal{X} \to [0, 1]$ and a *factual instance* $\hat{\boldsymbol{x}} \in \mathcal{X}$, the aim is to find a *counterfactual* $\tilde{\boldsymbol{x}} \in \mathcal{X}$ that has the shortest distance to $\hat{\boldsymbol{x}}$, and has the opposite target. The problem to obtain $\tilde{\boldsymbol{x}}$ can be formulated as

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad d(\hat{\boldsymbol{x}}, \boldsymbol{x}) \tag{6.1}$$

$$\text{s.t. } h(\boldsymbol{x}) \geq \tau, \tag{6.2}$$

where $d(\cdot, \cdot)$ is a distance function, often chosen to be the ℓ_1-norm or the ℓ_2-norm, and $\tau \in [0, 1]$ is a given threshold parameter for the classification decision.

Others have built on this work and proposed approaches that generate CEs with increased practical value, primarily by adding constraints to ensure actionability of the proposed changes and generating CEs that are close to the data manifold (Ustun et al. 2019, Russell 2019, Mahajan et al. 2019, Mothilal et al. 2020, Maragno et al. 2022). Nonetheless, the user agency provided by these methods remains theoretical: the generated CEs are exact point solutions that may remain difficult, if not impossible, to implement in practice. A minimal change to the proposed CE could fail to flip the model's prediction, especially since the CEs are close to the decision boundary due to minimizing the distance between $\hat{x}$ and $\tilde{x}$. As a solution, prior work suggests generat-

ing several CEs to increase the likelihood of generating at least one attainable solution. Approaches generating several CEs typically require solving the optimization problem multiple times (Russell 2019, Mothilal et al. 2020, Kanamori et al. 2021, Karimi et al. 2020a), which might heavily affect the optimization time when the number of explanations is large. Maragno et al. (2022) suggest using incumbent solutions, however, this does not allow to control the quality of sub-optimal solutions. On top of that, the added practical value may be unconvincing: each of the CEs is still sensitive to arbitrarily small changes in the actions implemented by the user (Dominguez-Olmedo et al. 2021, Pawelczyk et al. 2022, Virgolin and Fracaros 2023).

This problem has been acknowledged in prior work and falls under the discussion of robustness in CEs. In the literature, the concept of robustness in CEs has different meanings: (1) robustness to input perturbations (Slack et al. 2021, Artelt et al. 2021) and generating explanations for a group of individuals (Carrizosa et al. 2024), (2) robustness to model changes (Rawal et al. 2020, Forel et al. 2022, Upadhyay et al. 2021, Ferrario and Loi 2022, Black et al. 2021, Dutta et al. 2022, Bui et al. 2022), (3) robustness to hyperparameter selection (Dandl et al. 2020), and (4) robustness to recourse (Pawelczyk et al. 2022, Dominguez-Olmedo et al. 2021, Virgolin and Fracaros 2023). The latter perspective, albeit very user-centered, has so far received only a little attention. Our work focuses on the latter definition of robustness in CEs, specifically, the idea of robustness to recourse. This means that a counterfactual solution should remain valid even if small changes are made to the implemented recourse action. In other words, we aim to define regions of counterfactual solutions that allow the user to choose any point within that region to flip the model prediction. This extends the idea of offering several explanations for the user to choose from, such that not only the suggested point is a counterfactual, but every point in the defined region. Returning to the example we used above, a final explanation may be "if the *annual salary* would increase to anywhere between 50,000$ and 54,200$, then the *loan application* would be approved." While existing research has tackled this problem, their solutions are not comprehensive and have room for further improvements. In the remainder of this section, we will explore the related prior work and present our own contributions to this field.

Pawelczyk et al. (2022) introduce the notion of recourse invalidation rate, which amounts to the proportion of recourse that does not lead to the desired model prediction, *i.e.*, that is invalid. They model the noise around a counterfactual data point with a Gaussian distribution and suggest an approach that ensures the invalidation rate within a specified neighborhood around the counterfactual data point to be no larger than a target recourse invalidation rate. However, their work provides a heuristic solution using a gradient-based approach, which makes it not directly applicable to decision tree models. Additionally, it only provides a probabilistic robustness guarantee. Dominguez-Olmedo et al. (2021) introduce an approach where the optimal solution is surrounded by an uncertainty set such that every point in the set is a feasible solution.

They also model causality between (perturbed) features to obtain a more informative neighborhood of similar points. Given a structural causal model (SCM), they model such perturbations as additive interventions on the factual instance features. The authors design an iterative approach that works only for differentiable classifiers and does not guarantee that the generated recourse actions are adversarially robust. Virgolin and Fracaros (2023) incorporate the possibility of additional intervention to contrast perturbations in their search for CEs. They make a distinction between the features that could be changed and those that should be kept as they are, and introduce the concept of C-setbacks; a subset of perturbations in changeable features that work against the user. Rather than seeking CEs that are not invalidated by C-setbacks, they seek CEs for which the additional intervention cost to overcome the setback is minimal. Perturbations to features that should be kept as they are according to a CE are orthogonal to the direction of the counterfactual, and Virgolin and Fracaros (2023) approximate a robustness-score for such features. A drawback of this method is that it is only applicable in situations where additional intervention is possible, and not in situations where (*e.g.*, due to time limitations) only a single recourse is possible.

Our work addresses robustness to recourse by utilizing a robust optimization approach to generate regions of CEs. For a given factual instance, our method generates a CE that is robust to small perturbations. This gives the user more flexibility in implementing the recourse and reduces the risk of invalidating it. In this work we consider numerical features, ensuring that small perturbations do not affect the recourse validity, while categorical features are treated as immutable based on user preferences. Additionally, the generated CEs are optimal in terms of their objective distance to the factual instance. The proposed algorithm is proven to converge, ensuring that the optimal solution is reached. This is different from prior work that provides only heuristic algorithms which are not provably able to find the optimal (*i.e.*, closest) counterfactual point with a certain robustness guarantee (*e.g.,* Pawelczyk et al. 2022, Dominguez-Olmedo et al. 2021). Unlike prior research in this area, our approach is able to provide deterministic robustness guarantees for the CEs generated. Furthermore, our method does not require differentiability of the underlying ML model and is applicable to the tree-based models, which, to the best of our knowledge, has not been done before.

In summary, we make the following contributions:

- We propose an iterative algorithm that effectively finds global optimal robust CEs for trained decision trees, ensembles of trees, and neural networks.
- We prove the convergence of the algorithm for the considered trained models.
- We demonstrate the power of our algorithm on several datasets and different ML models. We empirically evaluate its convergence performance and compare the robustness as well as the validity of the generated CEs with the prior work

in the literature.

- We release an open-source software called RCE to make the proposed algorithm easily accessible to practitioners. Our software is available in a dedicated repository[1] through which all our results can be reproduced.

6.2 Robust counterfactual explanations

We consider binary classification problems, *i.e.*, we have a trained classifier $h : \mathcal{X} \to [0, 1]$ that assigns a value between zero and one to each data point in the data space $\mathcal{X} \subseteq \mathbb{R}^n$. A point $\boldsymbol{x} \in \mathcal{X}$ is then predicted to correspond to class $+1$, if $h(\boldsymbol{x}) \geq \tau$ and to class -1, otherwise. Here, $\tau \in [0, 1]$ is a given threshold parameter which is often chosen to be $\tau = 0.5$. Given a factual instance $\hat{\boldsymbol{x}} \in \mathcal{X}$ which is predicted to be in class -1, *i.e.*, $h(\hat{\boldsymbol{x}}) < \tau$, the robust CE problem is defined as

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad d(\boldsymbol{x}, \hat{\boldsymbol{x}}) \tag{6.3}$$

$$\text{s.t. } h(\boldsymbol{x} + \boldsymbol{s}) \geq \tau, \quad \forall \boldsymbol{s} \in \mathcal{S}, \tag{6.4}$$

where the $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ represents a distance function, *e.g.*, induced by the ℓ_1-, ℓ_2- or ℓ_∞-norm, and $\mathcal{S} \subset \mathbb{R}^n$ is a given uncertainty set. The idea of the problem is to find a point that is as close as possible to the factual instance $\hat{\boldsymbol{x}}$ such that for all perturbations $\boldsymbol{s} \in \mathcal{S}$, the corresponding point $\boldsymbol{x} + \boldsymbol{s}$ is classified as $+1$ which is enforced by constraints (6.4); see Figure 6.1. This results in a large set of counterfactual explanations.

We consider uncertainty sets of the type

$$\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^n \mid ||\boldsymbol{s}|| \leq \rho\}, \tag{6.5}$$

where $\|\cdot\|$ is a given norm. Popular choices are the ℓ_∞-norm, resulting in a box with upper and lower bounds on features, or the ℓ_2-norm, resulting in a circular uncertainty set. We refer to Ben-Tal et al. (2009) for a discussion of uncertainty sets. From the user perspective, choosing the ℓ_∞-norm has a practical advantage since the region $\mathcal{S}$ is a box, *i.e.*, we obtain an interval for each attribute of $\hat{\boldsymbol{x}}$. Each attribute can be changed in its corresponding interval independently, resulting in a counterfactual explanation. Hence, the user can easily detect if there exists a CE in the region that can be practically reached. The choice of the robustness budget ρ greatly depends on the specific domain. Larger values of ρ are associated with CEs that are more robust but can have a larger distance to the factual instance. However, our approach outlined in the subsequent sections is designed to minimize the proximity to the factual instance for a given robustness parameter ρ. If the perturbation applied to the CE ad-

[1] https://github.com/donato-maragno/robust-CE

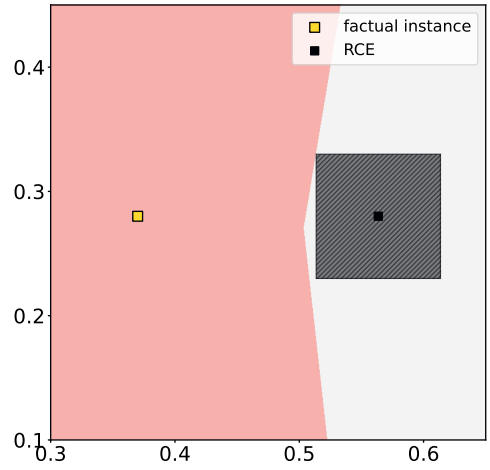


Figure 6.1. Robust CE for a neural network and using a box uncertainty set. All points in the red region are classified as -1, all points in the white region as $+1$.

heres to a known distribution, it becomes feasible to determine ρ in a way that offers a probabilistic guarantee of CE robustness. We refer to the Online Appendix A for a comprehensive guide on how to determine the appropriate value for ρ.

6.2.1 Comparison against model-robustness

There are several works that study the robustness of counterfactual explanations regarding changes in the parameters of the trained ML; see *e.g.*, Rawal et al. (2020), Upadhyay et al. (2021), Black et al. (2021), Ferrario and Loi (2022), Forel et al. (2022), Dutta et al. (2022), Bui et al. (2022). Model changes can appear frequently in real-world applications; *e.g.*, a model used for classification is re-trained with new data or different hyperparameter setup. In these cases the model-parameters can change and a counterfactual explanation for the old model can become invalid for the re-trained model.

Consider a classifier $h_{\hat{\omega}}$ where $\hat{\omega}$ is the vector of model parameters that was determined during the training process. In the case of a neural network, this vector contains all weights of the neural network, or in the case of a linear classifier, ω contains all weight parameters of the linear hyperplane separating the two classes.

Translated into the robust optimization setting we study in this work, a model-robust CE is a point, which remains a CE for all parameter values $\omega \in \Omega$, where
$\Omega = \{\omega : \|\omega - \hat{\omega}\| \leq \rho_{mod}\}$ is a given uncertainty set which contains all possible model parameters which have distance at most ρ_{mod} to the original weights of the classifier. In other words, if x^{CE} is a counterfactual point for the original classifier, *i.e.*, $h_{\hat{\omega}}(x^{CE}) \geq \tau$

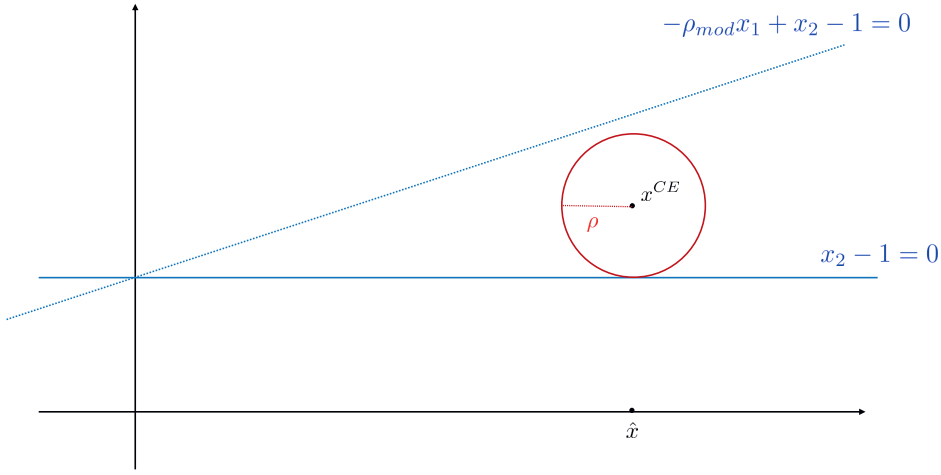


Figure 6.2. Example in Section 6.2.1.

then for each classifier h_ω where $\omega \in \Omega$ this must also be the case, *i.e.*, $h_\omega(x^{CE}) \geq \tau$. Note that defining model-robustness for decision tree models is much more elaborate since a decision tree is not only defined by the parameters of its split-hyperplanes but also by the tree structure which can change after re-training the model. So, defining Ω is not as straightforward as it is for linear models.

However, a natural question arises: Are the two concepts, model-robustness and recourse-robustness, equivalent? In this case concepts from both fields could profit from each other. Unfortunately, this is not the case, which we show in the following example.

Consider the linear classifier given by the hyperplane $0x_1 + x_2 - 1 = 0$, *i.e.*, every point $x \in \mathbb{R}^2$ where $x_2 - 1 \geq 0$ is classified as 1 and all others as 0. Then, for every $\rho, \rho_{mod} > 0$ there exists a recourse-robust CE with radius ρ which is not model-robust with radius ρ_{mod}. The construction works as follows: Let $\rho, \rho_{mod} > 0$ and define the factual instance $\hat{x} = (\frac{\rho}{2\rho_{mod}}, 0)$. Then the closest recourse-robust CE for radius ρ is $x^{CE} = (\frac{\rho}{2\rho_{mod}}, 1+\rho)$ for all relevant norms used in (6.5); see Figure 6.2. Now consider the ρ_{mod}-perturbed hyperplane $-\rho_{mod}x_1 + x_2 - 1 = 0$. For x^{CE}, it holds

$$-\rho_{mod}x_1 + x_2 - 1 = -\rho_{mod}\frac{\rho}{2\rho_{mod}} + 1 + \rho - 1 = -\frac{1}{2}\rho < 0.$$

Hence, x^{CE} is not a counterfactual point for the perturbed model.

A similar setup can be used to show that there exist points that are model-robust but not recourse-robust. Consider the classifier $w_1x_1 + w_2x_2 = 0$, where $w_1 = 0$ and $w_2 =$

1. We assume that only the parameters w_1, w_2 are allowed to change. Let $\rho, \rho_{mod} > 0$. Define the point $\tilde{x}^{CE} = (0, 0)$ which is classified as 1. Clearly, this point is model-robust for radius ρ_{mod}, since for every change of w_1, w_2 it holds $w_1 x_1 + w_2 x_2 = 0$. However, the point is not recourse-robust regarding radius ρ, since the perturbed point $\tilde{x} = (0, -\varepsilon)$ for $\varepsilon < \rho$ is classified as 0.

While the latter examples show that the equivalence of both robustness types does not hold, there can be special cases of models where both concepts are related. However, we place this interesting analysis on our future research agenda.

6.2.2 Algorithm

We note that the model in (6.3)-(6.4) has infinitely many constraints. One approach often used in robust optimization is to rewrite constraints (6.4) as

$$\min_{\boldsymbol{s} \in \mathcal{S}} h(\boldsymbol{x} + \boldsymbol{s}) \geq \tau,$$

and dualize the optimization problem on the left hand side. This leads to a problem with a finite number of constraints. Unfortunately, strong duality is required to perform this reformulation, which does not hold for most classifiers h involving non-convexity or integer variables[2]. In the latter case, we can use an alternative method popular in robust optimization where the constraints are generated iteratively. This iterative approach to solve problem (6.3)-(6.4) is known as the *adversarial approach*. The approach was intensively used for robust optimization problems; see Bienstock and Özbay (2008), Mutapcic and Boyd (2009). In Bertsimas et al. (2016a), the adversarial approach was compared to the classical robust reformulation.

The idea of the approach is to consider a relaxed version of the model, where only a finite subset of scenarios $\mathcal{Z} \subset \mathcal{S}$ is considered:

$$\min_{\boldsymbol{x} \in \mathcal{X}} d(\boldsymbol{x}, \hat{\boldsymbol{x}}) \tag{MP}$$

$$\text{s.t. } h(\boldsymbol{x} + \boldsymbol{s}) \geq \tau, \quad \forall \boldsymbol{s} \in \mathcal{Z}. \tag{6.6}$$

This problem is called the *master problem* (MP), and it only has a finite number of constraints. Note that the optimal value of (MP) is a lower bound of the optimal value of (6.3)-(6.4). However, an optimal solution $\boldsymbol{x}^*$ of (MP) is not necessarily feasible for the original problem, since there may exist a scenario in $\mathcal{S}$ that is not contained in $\mathcal{Z}$ for which the solution is not feasible. More precisely, it may be that there exists an $\boldsymbol{s} \in \mathcal{S}$ such that $h(\boldsymbol{x}^* + \boldsymbol{s}) < \tau$, and hence, $\boldsymbol{x}^*$ is not a robust counterfactual. In this case, we want to find such a scenario $\boldsymbol{s}^*$ that makes solution $\boldsymbol{x}^*$ infeasible. This can be

[2]See the Online Appendix B for the well-known dual approach applied to linear models.

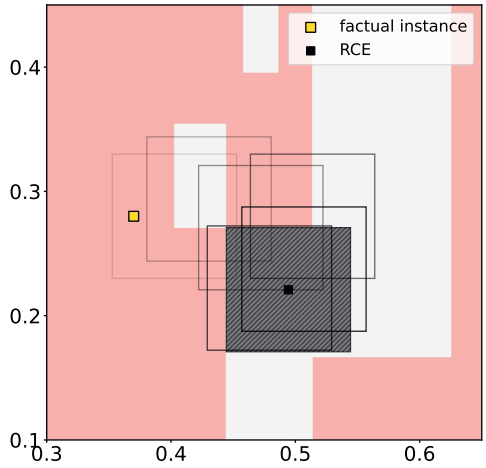


Figure 6.3. Iterations of Algorithm 2 to find the optimal robust CE for a decision tree. For each (MP) solution, we show the uncertainty box around it. As long as the box overlaps with the red region, a new scenario can be found and the solution moves in the next iteration.

done by solving the following, so called, *adversarial problem* (AP):

$$\max_{\boldsymbol{s}\in\mathcal{S}} \tau - h(\boldsymbol{x}^* + \boldsymbol{s}). \tag{AP}$$

The idea is to find a scenario $\boldsymbol{s}^* \in \mathcal{S}$ such that the prediction of classifier h for point $\boldsymbol{x}^* + \boldsymbol{s}^*$ is -1, *i.e.*, $\tau - h(\boldsymbol{x}^* + \boldsymbol{s}^*) > 0$. If we can find such a scenario and add it to the set $\mathcal{Z}$ in the MP, then $\boldsymbol{x}^*$ cannot be feasible anymore for (MP). To find the scenario with the largest impact, we maximize the constraint violation in the objective function in (AP). If the optimal value of (AP) is positive then $\boldsymbol{x}^* + \boldsymbol{s}^*$ is classified as -1, and the optimal solution $\boldsymbol{s}^*$ is added to $\mathcal{Z}$, and we calculate a solution $\boldsymbol{x}^*$ of the updated (MP). We iterate until no violating scenario can be found, that is, until the optimal value of (AP) is smaller or equal to zero. Note that in this case $h(\boldsymbol{x}^* + \boldsymbol{s}) \geq \tau$ holds for all $\boldsymbol{s} \in \mathcal{S}$, which means that $\boldsymbol{x}^*$ is a robust counterfactual. Algorithm 2 shows the steps of our approach, and Figure 6.3 shows its iterative behaviour. Each time a new scenario $\boldsymbol{s}$ is found by solving the AP, it is added to the uncertainty set $\mathcal{Z}$, and the new solution $\boldsymbol{x}^*$ moves to be feasible also for the new scenario. This is repeated until no scenario can be found anymore, *i.e.*, until the full box lies in the correct region. Note that instead of checking for a positive optimal value of (AP), we use an accuracy parameter $\varepsilon > 0$ in Algorithm 2. In this case, we can guarantee the convergence of our algorithm using the following result.

Algorithm 2 Adversarial Algorithm

Input: $\mathcal{S}$, $\hat{\boldsymbol{x}}$, $\varepsilon > 0$
$\mathcal{Z} = \{\mathbf{0}\}$
repeat
  $\boldsymbol{x}^* \leftarrow$ Solve (MP) with $\mathcal{Z}, \hat{\boldsymbol{x}}$
  $\boldsymbol{s}^*, \text{opt} \leftarrow$ Solve (AP) with $\boldsymbol{x}^*, \mathcal{S}$
  $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\boldsymbol{s}^*\}$
until opt $\leq \varepsilon$
Return: x^*

Theorem 6.1 (Mutapcic and Boyd, 2009). If $\mathcal{X}$ is bounded and if h is a Lipschitz continuous function, *i.e.*, there exists an $L > 0$ such that

$$|h(\boldsymbol{x_1}) - h(\boldsymbol{x_2})| \leq L\|\boldsymbol{x_1} - \boldsymbol{x_2}\|$$

for all $\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{X}$. Then, for any tolerance parameter value $\epsilon > 0$, Algorithm 2 terminates after a finite number of steps with a solution $\boldsymbol{x}^*$ such that

$$h(\boldsymbol{x}^* + \boldsymbol{s}) \geq \tau - \varepsilon$$

for all $\boldsymbol{s} \in \mathcal{S}$.

Indeed without Lipschitz continuity, the convergence of Algorithm 2 cannot be ensured. We elaborate on this necessity in the following example, for which Algorithm 2 does not terminate in a finite number of steps.

■ **Example 6.1.** Consider a classifier $h : \mathbb{R}^2 \to [0, 1]$ with $h(\boldsymbol{x}) = 0$, if $x_2 > \frac{1}{2}$ and $h(\boldsymbol{x}) = 1$, otherwise. The threshold is $\tau = 0.5$, *i.e.*, a point is classified as 1, if $x_2 \leq \frac{1}{2}$ and as -1, otherwise. The factual instance is $\hat{\boldsymbol{z}} = (0, 2)$, which is classified as -1. Furthermore, the uncertainty set is given as $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^2 : \|\boldsymbol{s}\|_\infty \leq 1\}$. We can warm-start Algorithm 2 with the (MP) solution $\boldsymbol{x}^1 = (0, 0)$. Now, assume that in iteration i the optimal solution returned by (AP) is $\boldsymbol{s}^i = (1, \frac{1}{2} + \sum_{j=1}^{i} \left(\frac{1}{4}\right)^j)$. Note that in the first iteration $\boldsymbol{s}^1 = (1, \frac{3}{4})$ lies on the boundary of $\mathcal{S}$ and $\boldsymbol{x}^1 + \boldsymbol{s}^1$ is classified as -1, *i.e.*, it is an optimal solution of (AP). We are looking now for the closest point $\boldsymbol{x}^2$ to $\hat{\boldsymbol{x}}$ such that $\boldsymbol{x}^2 + \boldsymbol{s}^1$ is classified as 1, that is, it has a second component of at most $\frac{1}{2}$. This is the point $\boldsymbol{x}^2 = (0, -\frac{1}{4})$ which must be the optimal solution of (MP). Note that $\boldsymbol{s}^2$ is again on the boundary of $\mathcal{S}$ and $\boldsymbol{x}^2 + \boldsymbol{s}^2 = (1, \frac{1}{2} + \frac{1}{8})$ is classified as -1. Hence, $\boldsymbol{s}^2$ is an optimal solution of (AP). We can conclude inductively that the optimal solution of (MP) in iteration i is $\boldsymbol{x}^i = (0, -\sum_{j=1}^{i} \left(\frac{1}{4}\right)^j)$ and that $\boldsymbol{s}^i$ is an optimal solution of (AP) in iteration i. Note that the latter is true, since the value of h is constant in the negative region, and hence, each point in the uncertainty set is an optimal solution of (AP). If the latter solutions are returned by (AP), then the sequence of solutions $\boldsymbol{x}^i$ converges to the point $\bar{\boldsymbol{x}} = (0, -\frac{1}{3})$ which follows from the limit of the geometric series. However,

$\bar{\boldsymbol{x}}$ is not a robust CE regarding S, since for instance, $\bar{\boldsymbol{x}}+(0,1) = (0, \frac{2}{3})$ is classified as -1. Consequently, Algorithm 2 never terminates. This example is illustrated in Figure 6.4.
■

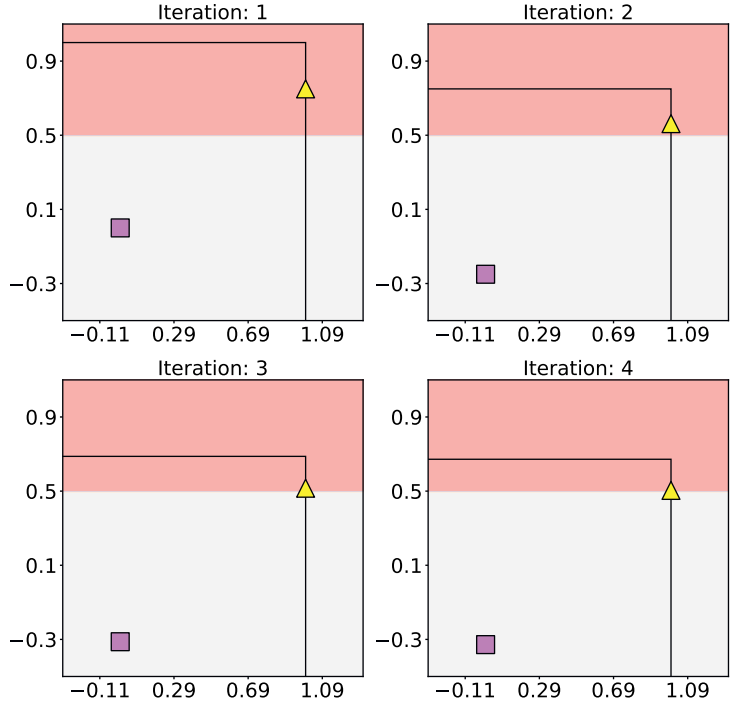


Figure 6.4. Illustration of the iterations of Algorithm 2 for the problem in Example 6.1. The purple square represents the current (MP) solution, while the yellow triangle represents the solution of the adversarial problem. The distance of the adversarial solutions to the decision boundary is $\left(\frac{1}{4}\right)^i$ in iteration i.

One difficulty is modeling the constraints of the form $h(\boldsymbol{x}+\boldsymbol{s}) \geq \tau$ for different trained classifiers. For decision trees, ensembles of decision trees, and neural networks, these constraints can be modeled by mixed-integer linear constraints as we present in the following section. Another difficulty is that h is discontinuous for decision trees and ensembles of decision trees. To handle these models, we have to find Lipschitz continuous extensions of h with equivalent predictions to guarantee convergence of Algorithm 2.

6.3 Trained models

The main contribution of this section is modeling MP and AP for different types of classifiers h by mixed-integer programming formulations. Furthermore, to prove convergence we need to assure that all studied classifiers h are Lipschitz continuous, which is not the case for tree-based models. Hence, we introduce a Lipschitz continuous classifier for tree-based models and derive the MP and AP for it.

We give reformulations for (MP) and (AP) for decision trees, tree ensembles, and neural networks that satisfy the conditions needed in Theorem 6.1 for convergence.

6.3.1 Decision trees

A decision tree (DT) partitions the data samples into distinct *leaves* through a series of *feature splits*. A split at node j is performed by a hyperplane $\tilde{\boldsymbol{a}}^\top \boldsymbol{x} = \tilde{b}$. We assume that $\tilde{\boldsymbol{a}}$ can have multiple non-zero elements, in which we have the hyperplane split setting – if there is only one non-zero element, this creates an orthogonal (single feature) split. Formally, each leaf $\mathcal{L}^i$ of a decision tree is defined by a set of (strict) inequalities

$$\mathcal{L}^i = \{\boldsymbol{x} \in X : \boldsymbol{a}^\top \boldsymbol{x} \leq b,\ \boldsymbol{\alpha}^\top \boldsymbol{x} < \beta; (\boldsymbol{a}, b) \in \mathcal{N}^i_{\leq}, (\boldsymbol{\alpha}, \beta) \in \mathcal{N}^i_{<}\},$$

where $\mathcal{N}^i_{\leq}$ and $\mathcal{N}^i_{<}$ contain all split parameters of the leaf for the corresponding inequality type. For ease of notation in the following, we do not distinguish between strict and non-strict inequalities and define $\mathcal{N}^i = N^i_{\leq} \cup \mathcal{N}^i_{<}$. Furthermore, it holds that $\mathbb{R}^n = \bigcup_{i \in L} \mathcal{L}^i$ where L is the index set of all leaves of the tree. Each leaf i is assigned a weight $p_i \in [0, 1]$, which is usually determined by the fraction of training data of class 1 inside the leaf. The classifier is a piecewise constant function h, where $h(\boldsymbol{x}) = p_i$ if and only if $\boldsymbol{x}$ is contained in leaf i. Since, h is a discontinuous step-function, and it is not Lipschitz continuous. To achieve convergence of our algorithm, we have to find a Lipschitz continuous function assigning the same classes to each data point as h. To this end we define the function

$$\tilde{h}(\boldsymbol{x}) = \begin{cases} \tau, & \boldsymbol{x} \in \mathcal{L}_i, p_i \geq \tau; \\ \tau - \min\limits_{(\boldsymbol{a}, b) \in \mathcal{N}^i} b - \boldsymbol{a}^\top \boldsymbol{x}, & \boldsymbol{x} \in \mathcal{L}_i, p_i < \tau. \end{cases}$$

We choose this function to have a constant value of τ for all leaves with $p_i \geq \tau$ while for a point $\boldsymbol{x}$ in one of the other leaves, we subtract from τ the minimum slack-value of the point over all leaf-defining constraints. Since the minimum slack on the boundary of the leaves is zero, $\tilde{h}$ is a continuous function and it holds $\tilde{h}(\boldsymbol{x}) < \tau$ in the interior of the latter leaves. Note that the value of h decreases if a point is farther away from the boundary of the leaf. Figure 6.5 illustrates this construction on a one-dimensional fea-

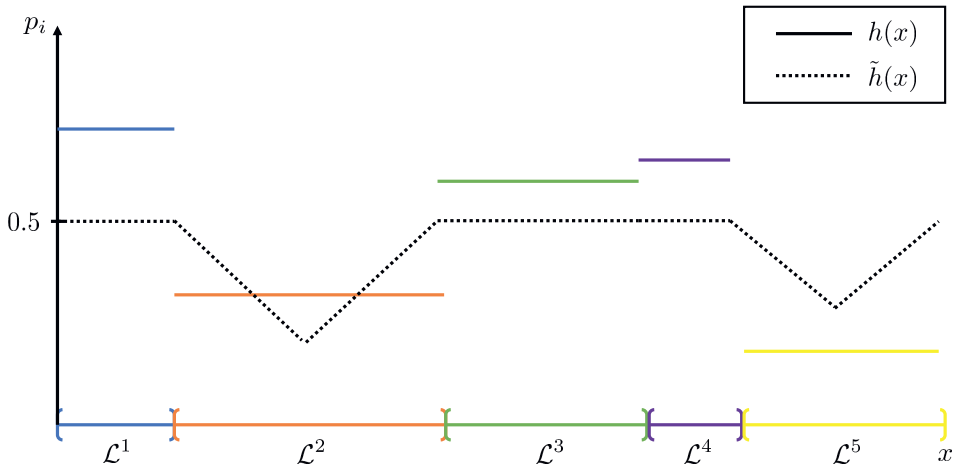


Figure 6.5. Example of the functions h and $\tilde{h}$ in a one-dimensional feature space.

ture space. Theoretically other function classes than piecewise linear functions could be used to connect the leaf values, as long as these functions are Lipschitz continuous on each leaf region. However, this would lead to non-linear problem formulations for the adversarial problem (see below), which would increase the computational effort of our method.

Unfortunately, due to imposed continuity, the predictions on the boundaries of the leaves can be different than the original predictions of h. We show in the following lemma that $\tilde{h}$ is Lipschitz continuous and, except on the leaf boundaries, the same class is assigned to each data point as it is done by the original classifier h.

Lemma 6.1. The function $\tilde{h}$ is Lipschitz continuous on $\mathcal{X}$ and $\text{int}\left(\{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \geq \tau\}\right) \subseteq \{\boldsymbol{x} : h(\boldsymbol{x}) \geq \tau\} \subseteq \{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \geq \tau\}$, where $\text{int}(\cdot)$ denotes the interior of the set.

Proof. Proof. We first show, that $\tilde{h}$ is Lipschitz continuous. To this end, let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$. Consider the following three cases.

Case 1: Both points are contained in a leaf with prediction 1, *i.e.*, $\boldsymbol{x} \in \mathcal{L}_i$ and $\boldsymbol{y} \in \mathcal{L}_{i'}$ with $p_i, p_{i'} \geq \tau$. In this case we have

$$|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| = |\tau - \tau| = 0 \leq \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{6.7}$$

Case 2: Point $\boldsymbol{x}$ is in a leaf with prediction 1, and point $\boldsymbol{y}$ is in a leaf with prediction -1, *i.e.*, $\boldsymbol{x} \in \mathcal{L}_i$ and $\boldsymbol{y} \in \mathcal{L}_{i'}$ with $p_i \geq \tau$ and $p_{i'} < \tau$. Since $\boldsymbol{x}$ is not contained in

$\mathcal{L}_{i'}$, there must be split parameters (a_*, b_*) (without loss of generality, we assume that it is related to a non-strict inequality) such that $\boldsymbol{a}_*^\top \boldsymbol{y} \leq b_*$ and $\boldsymbol{a}_*^\top \boldsymbol{x} > b_*$. It holds $\tilde{h}(\boldsymbol{x}) = \tau \geq \tilde{h}(\boldsymbol{y})$, and we obtain

$$
\begin{aligned}
|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| &= \tau - (\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y}) && (6.8)\\
&= \min_{(\boldsymbol{a},b)\in\mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y} && (6.9)\\
&\leq b_* - \boldsymbol{a}_*^\top \boldsymbol{y} && (6.10)\\
&< b_* - \boldsymbol{a}_*^\top \boldsymbol{y} + \boldsymbol{a}_*^\top \boldsymbol{x} - b_* && (6.11)\\
&= \boldsymbol{a}_*^\top (\boldsymbol{x} - \boldsymbol{y}) && (6.12)\\
&\leq \|\boldsymbol{a}_*\| \|\boldsymbol{x} - \boldsymbol{y}\|, && (6.13)
\end{aligned}
$$

where the first inequality follows from $(\boldsymbol{a}_*, b_*) \in \mathcal{N}^{i'}$, the second inequality follows from $\boldsymbol{a}_*^\top \boldsymbol{x} > b_*$, and for the last inequality we apply the Cauchy-Schwarz inequality.

Case 3: Both points are contained in a leaf with prediction -1, *i.e.*, $\boldsymbol{x} \in \mathcal{L}_i$ and $\boldsymbol{y} \in \mathcal{L}_{i'}$ with $p_i, p_{i'} < \tau$. First assume that $i \neq i'$. Without loss of generality, we also assume that $\tilde{h}(\boldsymbol{x}) \geq \tilde{h}(\boldsymbol{y})$. In this case, we have

$$
|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| \leq \tau - (\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y}), \tag{6.14}
$$

which follows from $\tilde{h}(\boldsymbol{x}) \leq \tau$ for all $\boldsymbol{x} \in \mathcal{X}$. We can prove Lipschitz continuity in this case by following the same steps as in Case 2. When $i = i'$, we designate $(\boldsymbol{a}_*, b_*)$ as the parameters which attain the minimum in

$$
\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^{i}} b - \boldsymbol{a}^\top \boldsymbol{x}. \tag{6.15}
$$

Then, we have

$$
\begin{aligned}
|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| &= \tau - b_* + \boldsymbol{a}_*^\top \boldsymbol{x} - (\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y}) && (6.16)\\
&\leq -b_* + \boldsymbol{a}_*^\top \boldsymbol{x} + b_* - \boldsymbol{a}_*^\top \boldsymbol{y} && (6.17)\\
&= \boldsymbol{a}_*^\top (\boldsymbol{x} - \boldsymbol{y}) && (6.18)\\
&\leq \|\boldsymbol{a}_*\| \|\boldsymbol{x} - \boldsymbol{y}\|, && (6.19)
\end{aligned}
$$

where we use $(\boldsymbol{a}_*, b_*) \in \mathcal{N}^{i'}$ for the first inequality, and the Cauchy-Schwarz inequality for the last one.

Following the three cases above, we show that $\tilde{h}$ is Lipschitz continuous with Lipschitz constant $L = \max_{i\in\mathcal{L}} \max_{(\boldsymbol{a},b)\in\mathcal{N}^i} \|\boldsymbol{a}\|$.

We now show the second part of the result. First, assume for $\boldsymbol{x}$ that $h(\boldsymbol{x}) \geq \tau$. This

implies that $\boldsymbol{x}$ is contained in a leaf $\mathcal{L}_i$ with $p_i \geq \tau$, and hence, $\tilde{h}(\boldsymbol{x}) = \tau$ showing the second inclusion. For the first inclusion, let now $\boldsymbol{x}$ be a point in the interior of the set $\{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \geq \tau\}$. Assume the contrary of the statement, *i.e.*, it is contained in a leaf $\mathcal{L}_i$ with $p_i < \tau$. We can assume that the leaf is full-dimensional, since otherwise the interior is empty. Then, by definition of $\tilde{h}$, it must hold that

$$\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^i} b - \boldsymbol{a}^\top \boldsymbol{x} \geq \tau. \tag{6.20}$$

That is, there is a $(\boldsymbol{a}, b) \in \mathcal{N}^i$ such that $\boldsymbol{a}^\top \boldsymbol{x} = b$. Since the leaf is a full-dimensional polyhedron, there exists a $\bar{\delta} > 0$ and a direction $\boldsymbol{v}$ such that $\boldsymbol{a}^\top(\boldsymbol{x} + \delta\boldsymbol{v}) < b$ for all $0 < \delta < \bar{\delta}$ and all $(\boldsymbol{a}, b) \in \mathcal{N}^i$. Consequently, $\tilde{h}(\boldsymbol{x} + \delta\boldsymbol{v}) < \tau$ for all $0 < \delta < \bar{\delta}$. This implies that $\boldsymbol{x}$ cannot be in the interior of the set $\{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \geq \tau\}$, which is a contradiction. Thus, $\boldsymbol{x}$ must be contained in a leaf with $p_i \geq \tau$ which proves the result. □

We can now derive the formulations for (MP) and (AP) for our tree model. By using Lemma 6.1, we can use h instead of $\tilde{h}$ to model Constraints (6.6) in (MP). Then, we can adapt the decision tree formulation proposed by Maragno et al. (2022) and reformulate Constraint (6.6) of (MP) as

$$\boldsymbol{a}^\top(\boldsymbol{x} + \boldsymbol{s}) - M(1 - l_i(\boldsymbol{s})) \leq b,\ i \in \mathcal{L}, (\boldsymbol{a}, b) \in \mathcal{N}^i_{\leq}, \boldsymbol{s} \in \mathcal{Z}, \tag{6.21}$$

$$\boldsymbol{a}^\top(\boldsymbol{x} + \boldsymbol{s}) - M(1 - l_i(\boldsymbol{s})) < b,\ i \in \mathcal{L}, (\boldsymbol{a}, b) \in \mathcal{N}^i_{<}, \boldsymbol{s} \in \mathcal{Z}, \tag{6.22}$$

$$\sum_{i\in\mathcal{L}} l_i(\boldsymbol{s}) = 1, \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.23}$$

$$\sum_{i\in\mathcal{L}} l_i(\boldsymbol{s}) p_i \geq \tau, \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.24}$$

$$l_i(\boldsymbol{s}) \in \{0, 1\}, \quad i \in \mathcal{L}, \boldsymbol{s} \in \mathcal{Z}, \tag{6.25}$$

where M is a predefined large-enough constant. The variables $l_i(\boldsymbol{s})$ are binary variables associated with the corresponding leaf i and scenario $\boldsymbol{s}$, where $l_i(\boldsymbol{s}) = 1$, if solution $\boldsymbol{x} + \boldsymbol{s}$ is contained in leaf i. Constraints (6.23) ensure that each scenario gets assigned to exactly one leaf. Constraints (6.21) and (6.22) ensure that only if leaf i is selected for scenario $\boldsymbol{s}$, *i.e.*, $l_i(\boldsymbol{s}) = 1$, then $\boldsymbol{x} + \boldsymbol{s}$ has to fulfill the corresponding constraints of $\mathcal{L}_i$ while the constraints for all other leaves can be violated, which is ensured by the big-M value. Note that in our computational experiments we use an $\tilde{\varepsilon}$-accuracy parameter to reformulate the strict inequalities as non-strict inequalities. Finally, Constraints (6.24) ensure that the chosen leaf has a weight p_i which is greater than or equal to the threshold τ. We can remove all variables and constraints of the problem related to leaves with $p_i < \tau$ together with constraint (6.24), since only leafs which correspond to label $+1$ can be chosen to obtain a feasible solution.

Using the Lipschitz continuous function $\tilde{h}$, (AP) can be reformulated as

$$\tau + \max_{\boldsymbol{s} \in \mathcal{S}} -\tilde{h}(\boldsymbol{x}^* + \boldsymbol{s}). \tag{6.26}$$

Optimizing $-\tilde{h}(\boldsymbol{x}^* + \boldsymbol{s})$ over $\mathcal{S}$ is equivalent to iterating over all leaves $\mathcal{L}_i$ with $p_i < \tau$ and maximizing the same function over the corresponding leaf. The problem is formulated as:

$$\max \; -\tau + \min_{(a,b) \in \mathcal{N}^i} \{b - \boldsymbol{a}^\top (\boldsymbol{x}^* + \boldsymbol{s})\} \tag{6.27}$$

$$\text{s.t. } \boldsymbol{a}^\top (\boldsymbol{x}^* + \boldsymbol{s}) \leq b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_{\leq}, \tag{6.28}$$

$$\boldsymbol{a}^\top (\boldsymbol{x}^* + \boldsymbol{s}) < b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_{<}, \tag{6.29}$$

$$\boldsymbol{s} \in \mathcal{S} \tag{6.30}$$

for each such leaf. Using a level-set transformation and substituting the latter problem in (6.26) leads to

$$\max \alpha \tag{6.31}$$

$$\text{s.t. } \alpha \leq w_{(\boldsymbol{a},b)}, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i, \tag{6.32}$$

$$\boldsymbol{a}^\top (\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a},b)} \leq b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_{\leq}, \tag{6.33}$$

$$\boldsymbol{a}^\top (\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a},b)} < b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_{<}, \tag{6.34}$$

$$\boldsymbol{s} \in \mathcal{S}, \boldsymbol{w} \geq 0, \tag{6.35}$$

which is equivalent to maximizing the minimum slacks of the constraints corresponding to the leaves. Geometrically this means that we try to find a perturbation $\boldsymbol{s}$ such that $\boldsymbol{x}^* + \boldsymbol{s}$ is as deep as possible in one of the negative leaves; see Figure 6.6. Note that the problems (6.31) are continuous optimization problems that can be solved efficiently by state-of-the-art solvers, such as, Gurobi Gurobi Optimization, LLC (2022) or CPLEX CPLEX, IBM ILOG (2009).

Heuristic variant. Using Algorithm 2 can be computationally demanding, since it requires solving (MP) and (AP) many times in an iterative manner. An alternative and more efficient approach can be conducted, where we try to find a CE $\boldsymbol{x}^*$ that is robust only regarding to one leaf of the tree. More precisely, this means that $\boldsymbol{x}^* + \boldsymbol{s}$ is contained in the same leaf for all $\boldsymbol{s} \in \mathcal{S}$. This is an approximation, since for each scenario $\boldsymbol{s}$, the point $\boldsymbol{x}^* + \boldsymbol{s}$ could be contained in a different neighboring leaf leading to better CEs; see Figure 6.8. Hence, the solutions of the latter approach may be non-optimal. When restricting to one leaf, we can iterate over all possible leaves $\mathcal{L}_i$ with $p_i < \tau$ and solve the resulting (MP):

$$\min d(\boldsymbol{x}, \hat{\boldsymbol{x}}) \tag{6.36}$$

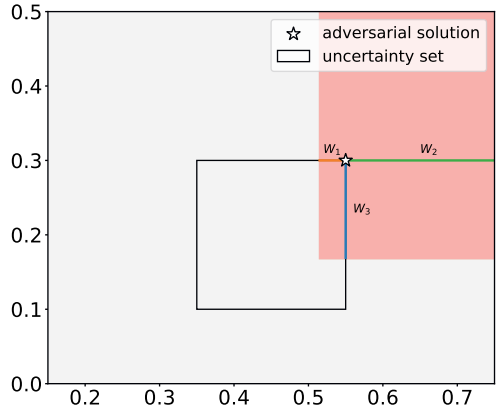


Figure 6.6. Slack values of a solution regarding the leaf-defining constraints where the minimum slack is maximized.

$$\text{s.t. } \boldsymbol{a}^\top \boldsymbol{x} + \rho||\boldsymbol{a}||^* \leq b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_{\leq}, \tag{6.37}$$

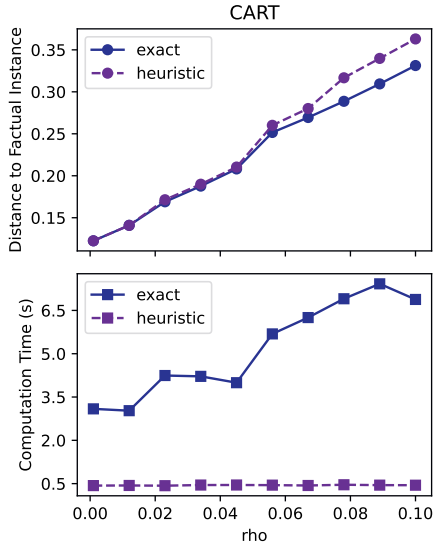
$$\boldsymbol{a}^\top \boldsymbol{x} + \rho||\boldsymbol{a}||^* < b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_{<}, \tag{6.38}$$

$$\boldsymbol{x} \in \mathcal{X}, \tag{6.39}$$

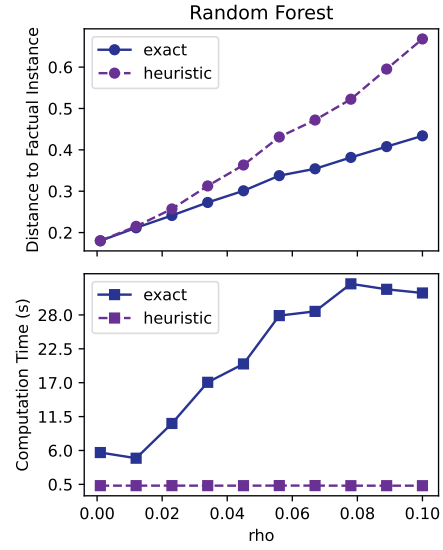
and choose the solution $\boldsymbol{x}^*$ for the leaf which yields the best optimal value. Note that, in that case, we do not need binary assignment variables anymore, since we only consider one leaf for (MP). Alternatively, we can obtain the same result modelling the entire decision tree using auxiliary binary variables, one for each leaf i with $p_i \geq \tau$. In Figure 6.7, we present the computation time and the distance of the calculated CE to the factual instance using both the heuristic and the (exact) adversarial algorithm. The results show that the heuristic approach outperforms the exact method in terms of speed, and its computation time remains unaffected by the robustness budget ρ. However, it is noteworthy that the CEs generated through the heuristic method have a larger distance to the factual instance where the difference to the optimal distance provided by our exact algorithm increases with increasing ρ.

6.3.2 Tree ensembles

In the case of tree ensembles like random forest (RF) and gradient boosting machines (GBM), we model the validity constraints by formulating each base learner separately. Assume we obtain K base learners. Since each base learner is a decision tree, we can use the construction of the constraints (6.21)-(6.25) and apply them to all base learners separately. Then, we add the constraints to the master problem, where each base learner k gets a separate copy $l_i(\boldsymbol{s})(k)$ of the assignment variables and has its own set of node inequalities given by $\boldsymbol{a}(\boldsymbol{k})$ and $b(k)$. Additionally, we have to replace



(*a*) DT with max. depth of 5.



(*b*) RF with 10 DTs.

Figure 6.7. Comparison of counterfactual explanations generated using the heuristic method and the adversarial algorithm (exact method) in terms of computation time and distance between the factual instance and the counterfactual explanations. The results are obtained using the Diabetes dataset and are averaged over 10 distinct factual instances.

constraint (6.24) by

$$\frac{\sum_{k=1}^{K}\sum_{i\in\mathcal{L}} l_i(\boldsymbol{s})(k)p_i(k)}{K} \geq \tau, \tag{6.40}$$

where $p_i(k)$ is the weight of leaf i in base learner k. This constraint forces the average prediction value of the tree ensemble to be larger than or equal to τ. Note that to model a majority vote, we can use $p_i(k) \in \{0,1\}$. Since a random forest is equivalent to a decision tree, the same methodology for (AP) can be used as in Section 6.3.1. Note that for classical DTs we may iterate over all leaves and solve Problem (6.31). However, deriving the polyhedral descriptions of all leaves for an ensemble of trees is very time-consuming. Instead (AP) can be reformulated as

$$\max \alpha \tag{6.41}$$
$$\text{s.t. } \alpha \leq w_{(\boldsymbol{a}(\boldsymbol{k}),b(k))}, \quad (\boldsymbol{a},b) \in \mathcal{N}^i(k),\ \forall k \in [K], \tag{6.42}$$
$$\boldsymbol{a}(\boldsymbol{k})^\top(\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a}(\boldsymbol{k}),b(k))} \leq b(k), \quad (\boldsymbol{a}(\boldsymbol{k}),b(k)) \in \mathcal{N}^i_{\leq}(k),\ \forall k \in [K], \tag{6.43}$$
$$\boldsymbol{a}(\boldsymbol{k})^\top(\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a}(\boldsymbol{k}),b(k))} < b(k), \quad (\boldsymbol{a}(\boldsymbol{k}),b(k)) \in \mathcal{N}^i_{<}(k),\ \forall k \in [K], \tag{6.44}$$
$$\boldsymbol{s} \in \mathcal{S},\ \ \boldsymbol{w} \geq 0, \tag{6.45}$$

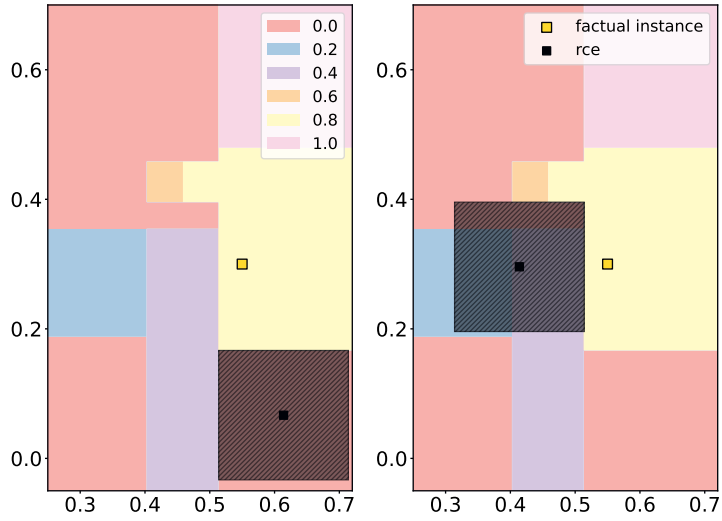


Figure 6.8. (Left) the CE of the heuristic approach where the whole set is restricted to be contained in one leaf. (Right) an optimal CE where the uncertainty set can overlap over different leaves of a decision tree.

where $\mathcal{N}^i_{\leq}(k), \mathcal{N}^i_{<}(k)$ contain the split parameters $(\boldsymbol{a(k)}, b(k))$ of the nodes of tree k as defined in Section 6.3.1 and we use $[K]$ to denote the set of the first K positive integers, that is $[K] = \{1, \ldots, K\}$.

Finally, note that since the classifier of an ensemble of trees is equivalent to a classical decision tree classifier, the convergence analysis presented in Section 6.3.1 holds also for the ensemble case.

6.3.3 Neural networks

In the case of neural networks convergence of Algorithm 2 is immediately guaranteed when we consider ReLU activation functions. More precisely, the evaluation function $h : \mathcal{X} \to [0, 1]$ of a trained neural network with rectified linear unit (ReLU) activation functions is Lipschitz continuous, since it is a concatenation of Lipschitz continuous functions; see the Online Appendix C for a formal proof.

Moreover, neural networks with ReLU activation functions belong to the MIP-representable class of ML models (Grimstad and Andersson 2019, Anderson et al. 2020), and we adopt the formulation proposed by Fischetti and Jo (2018). The ReLU operator of a neuron in layer l is given by

$$v_i^l = \max \left\{ 0, \beta_{i0}^l + \sum_{j \in N^{l-1}} \beta_{ij}^l v_j^{l-1} \right\}, \tag{6.46}$$

where $\boldsymbol{\beta}_i^l$ is the coefficient vector for neuron i in layer l, β_{i0}^l is the bias value and v_j^{l-1} is the output of neuron j of layer $l-1$. Note that in our model the input of the neural network can be a data point $\boldsymbol{x}$ perturbed by a scenario $\boldsymbol{s}$, *i.e.*, all variables depend on the perturbation $\boldsymbol{s}$. The input in the first layer is $\boldsymbol{v}^0(\boldsymbol{s}) = \boldsymbol{x} + \boldsymbol{s}$ and the output of layer l is denoted as $v_j^l(\boldsymbol{s})$. The ReLU operator (6.46) can then be linearly reformulated as

$$v_i^l(\boldsymbol{s}) \geq \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1,s}, \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.47}$$

$$v_i^l(\boldsymbol{s}) \leq \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1,s} + M_{LB}(1 - l_i^l(\boldsymbol{s})), \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.48}$$

$$v_i^l(\boldsymbol{s}) \leq M_{UB} l_i^l(\boldsymbol{s}), \ \boldsymbol{s} \in \mathcal{Z}, \tag{6.49}$$

$$v_i^l(\boldsymbol{s}) \geq 0, \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.50}$$

$$l_i^l(\boldsymbol{s}) \in \{0, 1\}, \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.51}$$

where M_{LB} and M_{UB} are big-M values.

The following is a complete formulation of the master problem (MP) in the case of neural networks with ReLU activation functions:

$$\min_{\boldsymbol{x} \in \mathcal{X}} d(\boldsymbol{x}, \hat{\boldsymbol{x}}) \tag{6.52}$$

$$\text{s.t.} \sum_{j \in \mathcal{N}^L} \beta_j^L v_j^{L-1}(\boldsymbol{s}) \geq \tau, \quad \boldsymbol{s} \in \mathcal{Z}, \tag{6.53}$$

$$v_i^l(\boldsymbol{s}) \geq \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1}(\boldsymbol{s}), \quad \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \ \forall l \in [L], \tag{6.54}$$

$$v_i^l(\boldsymbol{s}) \leq \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1}(\boldsymbol{s}) + M_{LB}(1 - l_i^l(\boldsymbol{s})), \quad \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \ \forall l \in [L], \tag{6.55}$$

$$v_i^l(\boldsymbol{s}) \leq M_{UB} l_i^l(\boldsymbol{s}), \ \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \ \forall l \in [L], \tag{6.56}$$

$$v_i^0(\boldsymbol{s}) = x_i + s_i, \quad \boldsymbol{s} \in \mathcal{Z}, \ \forall i \in [n], \tag{6.57}$$

$$v_i^l(\boldsymbol{s}) \geq 0, \quad \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \ \forall l \in [L], \tag{6.58}$$

$$l_i^l(\boldsymbol{s}) \in \{0, 1\}, \quad \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \ \forall l \in [L], \tag{6.59}$$

where L represents the number of layers with $[L] = \{1, \ldots, L\}$ and $\mathcal{N}^l$ the set of neurons in layer l. The first $L-1$ layers are activated by a ReLU function except for the output layer, which consists of a single node that is a linear combination of the node values in layer $L-1$. The variable $v_i^l(\boldsymbol{s})$ is the output of the activation function in node i, layer l, and scenario $\boldsymbol{s}$.

Likewise, the adversarial problem (AP) is formulated as

$$\max_{\boldsymbol{s}\in\mathcal{S}} \tau - \sum_{j\in\mathcal{N}^L} \beta_j^L v_j^{L-1}, \tag{6.60}$$

$$\text{s.t. } v_i^l \geq \beta_{i0}^l + \sum_{j\in\mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1}, \quad i \in \mathcal{N}^l,\ \forall l \in [L], \tag{6.61}$$

$$v_i^l \leq \beta_{i0}^l + \sum_{j\in\mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1} + M_{LB}(1-l_i^l), \quad i \in \mathcal{N}^l,\ \forall l \in [L], \tag{6.62}$$

$$v_i^l \leq M_{UB} l_i^l,\ i \in \mathcal{N}^l,\ \forall l \in [L], \tag{6.63}$$

$$v_i^0 = s_i + x_i^*, \quad i = 1, \ldots, n, \tag{6.64}$$

$$v_i^l \geq 0, \quad i \in \mathcal{N}^l,\ \forall l \in [L], \tag{6.65}$$

$$l_i^l \in \{0,1\}, \quad i \in \mathcal{N}^l,\ \forall l \in [L], \tag{6.66}$$

$$\boldsymbol{s} \in \mathcal{S}, \tag{6.67}$$

where $\boldsymbol{x}^*$ is the counterfactual solution of (MP).

6.4 Experiments

In this section, we aim to illustrate the effectiveness of our method by conducting empirical experiments on various datasets. The mixed-integer optimization formulations of the ML models used in our experiments are based on Maragno et al. (2023). In our experiments, we consistently set ϵ to 1e-7 and utilize a big M value of 1e3 for decision trees and tree ensembles, while for neural networks, we employ a big M value of 100. The experiments were conducted on a computer with an Apple M1 Pro processor and 16 GB of RAM. For reproducibility, our open-source implementation can be found at our repository[3]. It is important to note that, to the best of our knowledge, the present work is the first approach that generates a region of CEs for a range of different models, involving non-differentiable models. Therefore, a comparison to prior work is only possible for the case of neural networks with ReLU activation functions. In the last part of the experiments, we compare our method against the one proposed by Dominguez-Olmedo et al. (2021) in terms of CE validity and robustness.

In the first part of the experiments, we analyze our method using three well-known datasets: BANKNOTE AUTHENTICATION, DIABETES, and IONOSPHERE (Dua and Graff 2017). Before training the ML models, we scaled each feature to be between zero and one. None of the datasets contain categorical features, which otherwise would have been considered immutable or fixed according to the user's preference. We apply our algorithm to generate robust CEs for 20 factual instances randomly selected from the dataset. We use ℓ_∞-norm as uncertainty set with a radius (ρ) of 0.01 and 0.05. The

[3] https://github.com/donato-maragno/robust-CE

		BANKNOTE AUTHENTICATION 4 features			DIABETES 8 features			IONOSPHERE 34 features		
Model	**Specs**	**Comp. time (s)**	**# iterations**	**# early stops**	**Comp. time (s)**	**# iterations**	**# early stops**	**Comp. time (s)**	**# iterations**	**# early stops**
					$\rho = 0.01$					
Linear	ElasticNet	0.25 (0.01)	–	–	0.23 (0.01)	–	–	0.25 (0.01)	–	–
DT	max depth: 3	1.53 (0.04)	1.00 (0.00)	–	1.66 (0.07)	1.20 (0.09)	–	1.66 (0.07)	1.10 (0.07)	–
	max depth: 5	1.86 (0.09)	1.10 (0.07)	–	2.29 (0.11)	1.20 (0.09)	–	1.96 (0.08)	1.10 (0.07)	–
	max depth: 10	3.90 (0.88)	2.00 (0.58)	–	5.77 (0.48)	1.40 (0.13)	–	2.86 (0.16)	1.20 (0.09)	–
RF*	# est.: 5	3.50 (0.36)	1.75 (0.22)	–	5.89 (1.98)	2.60 (0.83)	–	3.79 (0.24)	1.70 (0.13)	–
	# est.: 10	6.74 (1.03)	2.60 (0.40)	–	7.20 (0.94)	2.35 (0.29)	–	8.76 (0.89)	2.85 (0.33)	–
	# est.: 20	21.21 (4.61)	4.55 (0.99)	–	33.55 (7.48)	6.15 (0.79)	–	22.33 (2.83)	4.40 (0.51)	–
	# est.: 50	115.79 (34.35)	7.80 (1.65)	–	110.24 (32.43)	6.47 (1.39)	3 ($\bar{\rho} = 0.007$)	137.26 (33.37)	8.20 (1.20)	–
	# est.: 100	214.38 (65.07)	6.44 (1.31)	2 ($\bar{\rho} = 0.009$)	274.09 (71.93)	8.87 (1.49)	5 ($\bar{\rho} = 0.004$)	285.62 (95.57)	8.27 (2.02)	9 ($\bar{\rho} = 0.004$)
GBM**	# est.: 5	2.70 (0.22)	1.20 (0.14)	–	2.76 (0.22)	1.85 (0.17)	–	2.37 (0.15)	1.60 (0.13)	–
	# est.: 10	3.20 (0.30)	1.45 (0.15)	–	2.72 (0.29)	1.50 (0.24)	–	4.35 (0.44)	2.75 (0.30)	–
	# est.: 20	5.94 (0.50)	2.60 (0.23)	–	4.25 (0.45)	2.15 (0.28)	–	9.01 (1.11)	3.85 (0.50)	–
	# est.: 50	18.38 (1.62)	4.05 (0.35)	–	24.60 (8.21)	5.85 (1.41)	–	81.33 (28.39)	8.90 (1.36)	–
	# est.: 100	87.11 (26.24)	7.28 (0.77)	2 ($\bar{\rho} = 0.006$)	164.32 (42.12)	11.63 (2.00)	1 ($\bar{\rho} = 0.004$)	137.98 (22.74)	10.33 (0.97)	2 ($\bar{\rho} = 0.007$)
NN	(10,)	1.63 (0.04)	1.00 (0.00)	–	1.55 (0.06)	1.00 (0.00)	–	2.22 (0.19)	1.80 (0.21)	–
	(10, 10, 10)	2.98 (0.15)	1.15 (0.08)	–	2.40 (0.12)	1.15 (0.08)	–	13.01 (3.57)	2.30 (0.40)	–
	(50,)	2.60 (0.14)	1.00 (0.00)	–	2.09 (0.12)	1.05 (0.05)	–	5.75 (0.75)	1.20 (0.12)	–
	(100,)	3.53 (0.15)	1.00 (0.00)	–	4.36 (0.62)	1.10 (0.07)	–	61.31 (33.11)	1.50 (0.22)	10 ($\bar{\rho} = 0.000$)
					$\rho = 0.05$					
Linear	ElasticNet	0.13 (0.00)	–	–	0.15 (0.01)	–	–	0.14 (0.00)	–	–
DT	max depth: 3	1.00 (0.06)	1.70 (0.15)	–	1.09 (0.07)	1.60 (0.15)	–	1.02 (0.07)	1.30 (0.15)	–
	max depth: 5	1.18 (0.11)	1.80 (0.22)	–	1.63 (0.13)	2.05 (0.22)	–	1.33 (0.10)	1.60 (0.18)	–
	max depth: 10	2.22 (0.41)	2.95 (0.61)	–	8.84 (1.49)	4.45 (0.59)	–	3.68 (2.14)	2.95 (1.49)	–
RF*	# est.: 5	2.85 (0.47)	3.25 (0.54)	–	4.29 (0.73)	5.25 (0.86)	–	2.27 (0.19)	2.35 (0.23)	–
	# est.: 10	13.51 (3.03)	8.95 (1.60)	–	14.71 (3.53)	8.35 (1.24)	–	7.41 (1.22)	5.15 (0.67)	–
	# est.: 20	13.86 (3.58)	5.53 (0.92)	1 ($\bar{\rho} = 0.041$)	89.00 (28.17)	14.00 (2.28)	2 ($\bar{\rho} = 0.045$)	47.44 (26.35)	9.50 (2.52)	–
	# est.: 50	101.21 (24.90)	11.37 (1.53)	1 ($\bar{\rho} = 0.048$)	303.27 (93.95)	16.73 (2.55)	9 ($\bar{\rho} = 0.034$)	307.72 (72.52)	19.31 (3.15)	7 ($\bar{\rho} = 0.044$)
	# est.: 100	156.28 (33.21)	8.70 (1.02)	–	453.67 (111.27)	15.43 (2.19)	13 ($\bar{\rho} = 0.032$)	156.45 (116.11)	5.75 (2.29)	16 ($\bar{\rho} = 0.029$)
GBM**	# est.: 5	1.57 (0.15)	1.75 (0.24)	–	1.50 (0.10)	2.05 (0.20)	–	1.97 (0.32)	2.65 (0.50)	–
	# est.: 10	4.84 (0.58)	3.55 (0.46)	–	8.87 (4.44)	8.55 (3.21)	–	11.76 (6.51)	8.85 (3.27)	–
	# est.: 20	12.72 (2.22)	8.28 (0.85)	2 ($\bar{\rho} = 0.038$)	41.81 (17.86)	17.05 (4.37)	1 ($\bar{\rho} = 0.025$)	19.20 (6.32)	9.45 (1.80)	–
	# est.: 50	73.23 (27.00)	13.76 (1.82)	3 ($\bar{\rho} = 0.039$)	223.87 (125.98)	19.86 (4.23)	13 ($\bar{\rho} = 0.027$)	139.18 (56.80)	16.31 (3.03)	7 ($\bar{\rho} = 0.023$)
	# est.: 100	274.22 (51.40)	17.25 (1.57)	4 ($\bar{\rho} = 0.040$)	–	–	20 ($\bar{\rho} = 0.022$)	537.13 (295.04)	15.00 (2.08)	17 ($\bar{\rho} = 0.022$)
NN	(10,)	0.90 (0.01)	1.00 (0.00)	–	1.00 (0.04)	1.15 (0.08)	–	2.96 (0.23)	3.00 (0.25)	–
	(10, 10, 10)	1.48 (0.03)	1.00 (0.00)	–	2.02 (0.26)	1.65 (0.21)	–	229.69 (104.80)	4.90 (0.67)	10 ($\bar{\rho} = 0.039$)
	(50,)	1.39 (0.06)	1.00 (0.00)	–	1.75 (0.13)	1.35 (0.11)	–	19.06 (6.63)	2.37 (0.24)	1 ($\bar{\rho} = 0.049$)
	(100,)	1.83 (0.05)	1.00 (0.00)	–	5.88 (1.19)	1.80 (0.16)	–	289.62 (125.61)	2.70 (0.30)	10 ($\bar{\rho} = 0.000$)

* max depth of each decision tree equal to 3.; ** max depth of each decision tree equal to 2.

Table 6.1. Generation of robust CEs for 20 factual instances, using ℓ_∞-norm as uncertainty set.

distance function adopted is the ℓ_1-norm, which can be linearly expressed within the optimization model. For each instance, we use a time limit of 1000 seconds. Although less practical from a user's perspective, we also report the results using ℓ_2-norm as uncertainty set in Table 6.2. The datasets used in our analysis do not require any additional constraints, such as *actionability*, *sparsity*, or *data manifold closeness*. However, it is important to note that these types of constraints can be added to our master problem when needed by using constraints like the ones proposed in (Maragno et al. 2022). The accuracy of each trained ML model is provided in the Online Appendix D.

In Table 6.1, we show (from left to right) the type of ML model, model-specific hyperparameters, and for each dataset, the average computation time (in seconds), the number of iterations performed by the algorithm, and the number of instances where the algorithm hits the time limit without providing an optimal solution. For the computation time and the number of iterations, we show the standard error values in parentheses. For the early stops, we show the (average) maximum radius of the uncertainty set, which is feasible for the generated counterfactual solutions in each iteration of the algorithm. The latter value gives a measure for the robustness of the returned solution. As for hyperparameters, we report the maximum tree depth for DT, the number of generated trees for RF and GBM, and the depth of each layer in the NN. The results

		BANKNOTE AUTHENTICATION 4 features			DIABETES 8 features			IONOSPHERE 34 features		
Model	**Specs**	**Comp. time (s)**	**# iterations**	**# early stops**	**Comp. time (s)**	**# iterations**	**# early stops**	**Comp. time (s)**	**# iterations**	**# early stops**
					$\rho = 0.01$					
Linear	ElasticNet	0.24 (0.01)	–	–	0.24 (0.01)	–	–	0.26 (0.01)	–	–
DT	max depth: 3	2.09 (0.11)	1.45 (0.11)	–	2.18 (0.13)	1.70 (0.15)	–	5.54 (0.38)	1.20 (0.09)	–
	max depth: 5	2.64 (0.14)	1.53 (0.12)	1 ($\bar{\rho} = 0.000$)	4.21 (0.39)	2.00 (0.23)	–	10.17 (0.78)	1.56 (0.16)	4 ($\bar{\rho} = 0.000$)
	max depth: 10	4.61 (0.88)	2.70 (0.58)	–	14.17 (2.64)	2.63 (0.50)	1 ($\bar{\rho} = 0.000$)	21.75 (2.58)	2.06 (0.26)	2 ($\bar{\rho} = 0.000$)
RF*	# est.: 5	5.20 (0.62)	2.35 (0.34)	–	7.96 (1.65)	3.17 (0.55)	2 ($\bar{\rho} = 0.004$)	69.94 (10.77)	4.41 (0.54)	3 ($\bar{\rho} = 0.006$)
	# est.: 10	10.83 (2.68)	3.41 (0.78)	3 ($\bar{\rho} = 0.006$)	15.20 (3.17)	4.20 (0.68)	–	237.03 (41.12)	5.50 (0.70)	4 ($\bar{\rho} = 0.003$)
	# est.: 20	22.59 (7.10)	4.15 (1.22)	7 ($\bar{\rho} = 0.004$)	104.42 (27.35)	9.81 (1.78)	4 ($\bar{\rho} = 0.007$)	370.21 (41.84)	7.33 (0.73)	5 ($\bar{\rho} = 0.004$)
	# est.: 50	61.93 (14.29)	3.89 (1.22)	11 ($\bar{\rho} = 0.004$)	137.37 (39.96)	6.50 (1.51)	10 ($\bar{\rho} = 0.004$)	570.88 (111.94)	9.70 (1.61)	10 ($\bar{\rho} = 0.001$)
	# est.: 100	103.33 (31.75)	3.20 (0.89)	10 ($\bar{\rho} = 0.004$)	177.47 (41.01)	3.80 (0.86)	15 ($\bar{\rho} = 0.002$)	531.13 (121.00)	6.17 (0.98)	14 ($\bar{\rho} = 0.001$)
GBM**	# est.: 5	4.50 (0.50)	2.50 (0.34)	–	4.67 (0.47)	2.45 (0.26)	–	11.47 (1.48)	4.42 (0.66)	8 ($\bar{\rho} = 0.002$)
	# est.: 10	6.87 (0.88)	3.15 (0.42)	–	6.11 (1.04)	2.70 (0.48)	–	42.68 (6.67)	6.88 (0.86)	3 ($\bar{\rho} = 0.001$)
	# est.: 20	14.02 (1.09)	4.40 (0.34)	–	10.97 (1.45)	3.65 (0.49)	–	58.16 (7.01)	9.62 (1.00)	4 ($\bar{\rho} = 0.004$)
	# est.: 50	34.03 (3.15)	5.21 (0.44)	1 ($\bar{\rho} = 0.010$)	76.04 (33.09)	9.35 (2.21)	–	192.70 (42.52)	14.77 (2.47)	7 ($\bar{\rho} = 0.006$)
	# est.: 100	133.53 (24.40)	8.50 (0.98)	6 ($\bar{\rho} = 0.007$)	201.13 (77.68)	12.08 (2.92)	8 ($\bar{\rho} = 0.005$)	415.96 (80.09)	17.29 (3.36)	13 ($\bar{\rho} = 0.003$)
NN	(10,)	1.64 (0.09)	1.00 (0.00)	–	1.73 (0.05)	1.00 (0.00)	–	4.73 (0.57)	1.27 (0.19)	9 ($\bar{\rho} = 0.004$)
	(10, 10, 10)	2.54 (0.15)	1.00 (0.00)	–	2.04 (0.10)	1.00 (0.00)	6 ($\bar{\rho} = 0.000$)	282.24 (193.65)	1.50 (0.50)	18 ($\bar{\rho} = 0.003$)
	(50,)	2.42 (0.14)	1.00 (0.00)	–	2.00 (0.08)	1.00 (0.00)	1 ($\bar{\rho} = 0.000$)	48.13 (12.76)	2.40 (0.51)	15 ($\bar{\rho} = 0.008$)
	(100,)	3.57 (0.14)	1.00 (0.00)	–	5.28 (0.83)	1.15 (0.11)	–	352.64 (153.96)	1.09 (0.09)	9 ($\bar{\rho} = 0.000$)
					$\rho = 0.05$					
Linear	ElasticNet	0.15 (0.00)	–	–	0.34 (0.02)	–	–	0.26 (0.01)	–	–
DT	max depth: 3	1.72 (0.17)	2.25 (0.19)	–	6.71 (0.65)	2.40 (0.24)	–	4.40 (1.35)	2.60 (0.90)	–
	max depth: 5	2.69 (0.52)	3.95 (0.74)	1 ($\bar{\rho} = 0.026$)	20.28 (2.88)	5.00 (0.68)	1 ($\bar{\rho} = 0.000$)	5.91 (1.09)	2.26 (0.40)	1 ($\bar{\rho} = 0.000$)
	max depth: 10	4.44 (0.80)	4.95 (0.93)	–	178.35 (34.48)	9.11 (1.30)	1 ($\bar{\rho} = 0.045$)	20.91 (10.20)	5.05 (1.51)	–
RF*	# est.: 5	4.49 (0.64)	4.40 (0.64)	–	117.95 (28.96)	9.63 (1.82)	1 ($\bar{\rho} = 0.049$)	28.79 (2.76)	6.35 (0.51)	–
	# est.: 10	12.39 (2.57)	6.82 (1.14)	3 ($\bar{\rho} = 0.048$)	200.39 (69.46)	12.92 (2.93)	7 ($\bar{\rho} = 0.043$)	232.63 (36.46)	10.79 (1.36)	1 ($\bar{\rho} = 0.042$)
	# est.: 20	51.29 (18.03)	10.17 (2.32)	8 ($\bar{\rho} = 0.049$)	149.39 (46.57)	12.75 (2.78)	12 ($\bar{\rho} = 0.042$)	450.46 (68.06)	8.50 (1.06)	10 ($\bar{\rho} = 0.027$)
	# est.: 50	93.18 (51.21)	7.78 (2.17)	11 ($\bar{\rho} = 0.048$)	560.78 (–)	6.00 (–)	19 ($\bar{\rho} = 0.031$)	510.69 (341.34)	7.50 (3.50)	18 ($\bar{\rho} = 0.021$)
	# est.: 100	108.47 (29.76)	5.25 (0.80)	8 ($\bar{\rho} = 0.047$)	868.10 (–)	4.00 (–)	19 ($\bar{\rho} = 0.024$)	495.03 (268.88)	9.00 (2.08)	17 ($\bar{\rho} = 0.014$)
GBM**	# est.: 5	2.67 (0.33)	3.55 (0.47)	–	8.89 (1.29)	3.68 (0.53)	1 ($\bar{\rho} = 0.000$)	11.38 (1.83)	5.74 (0.68)	1 ($\bar{\rho} = 0.029$)
	# est.: 10	5.77 (0.45)	5.55 (0.44))	–	46.46 (14.33)	12.00 (3.12)	–	72.15 (15.31)	15.19 (2.62)	4 ($\bar{\rho} = 0.005$)
	# est.: 20	23.89 (3.55)	10.41 (1.00)	3 ($\bar{\rho} = 0.046$)	153.43 (43.11)	18.76 (3.73)	3 ($\bar{\rho} = 0.036$)	156.17 (19.16)	16.42 (1.71)	8 ($\bar{\rho} = 0.005$)
	# est.: 50	123.75 (51.10)	18.14 (4.47)	6 ($\bar{\rho} = 0.044$)	243.24 (80.72)	24.50 (6.31)	14 ($\bar{\rho} = 0.029$)	894.23 (100.35)	32.50 (6.50)	18 ($\bar{\rho} = 0.013$)
	# est.: 100	389.06 (124.83)	20.25 (3.32)	12 ($\bar{\rho} = 0.044$)	– (–)	– (–)	20 ($\bar{\rho} = 0.020$)	– (–)	– (–)	20 ($\bar{\rho} = 0.010$)
NN	(10,)	0.91 (0.00)	1.00 (0.00)	–	4.13 (0.13)	1.00 (0.00)	–	36.34 (5.22)	1.68 (0.19)	1 ($\bar{\rho} = 0.000$)
	(10, 10, 10)	1.45 (0.03)	1.00 (0.00)	–	8.02 (0.86)	1.31 (0.18)	4 ($\bar{\rho} = 0.024$)	328.70 (72.59)	2.38 (0.35)	4 ($\bar{\rho} = 0.012$)
	(50,)	1.36 (0.03)	1.00 (0.00)	–	11.27 (1.10)	1.50 (0.15)	–	57.07 (7.28)	1.22 (0.10)	2 ($\bar{\rho} = 0.000$)
	(100,)	2.26 (0.04)	1.00 (0.00)	–	26.82 (3.39)	1.30 (0.13)	–	111.85 (26.18)	1.44 (0.24)	11 ($\bar{\rho} = 0.001$)

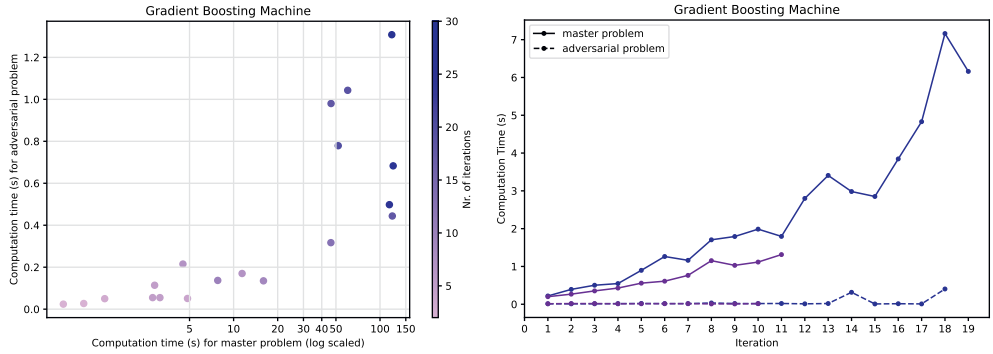
* max depth of each decision tree equal to 3.; ** max depth of each decision tree equal to 2.

Table 6.2. Generation of robust CEs for 20 factual instances, using ℓ_2-norm as uncertainty set.

indicate that the computation time and the number of iterations increase primarily due to the complexity of the ML models rather than the number of features in the datasets. We further inspect the computation time by looking at the time needed to solve the MP and AP at each iteration. To do so, we use the DIABETES dataset and train a GBM with 100 estimators. In Figure 6.9a we plot the overall computation time of the AP versus the overall computation time of the MP for 20 instances. In Figure 6.9b we inspect the time required per iteration for two of those instances. We can see that the time required to solve the AP remains small and stable while the time required to solve the MP increases exponentially with each iteration, *i.e.*, as more scenarios are added.

Decision trees tend to overfit as their depth increases. In Figure 6.10, we show iterations for a decision tree with maximum depths of (a) 10 and (b) 3. We observe that the more complex model substantially increases the number of iterations, and hence the computation time.

When the time limit is reached, we can still provide a list of solutions generated by solving the (MP) at each iteration. Each one of these solutions comes with information on the distance to the actual instance and the radius of the uncertainty set for which the model predictions still belong to the desired class for all perturbations. Therefore,



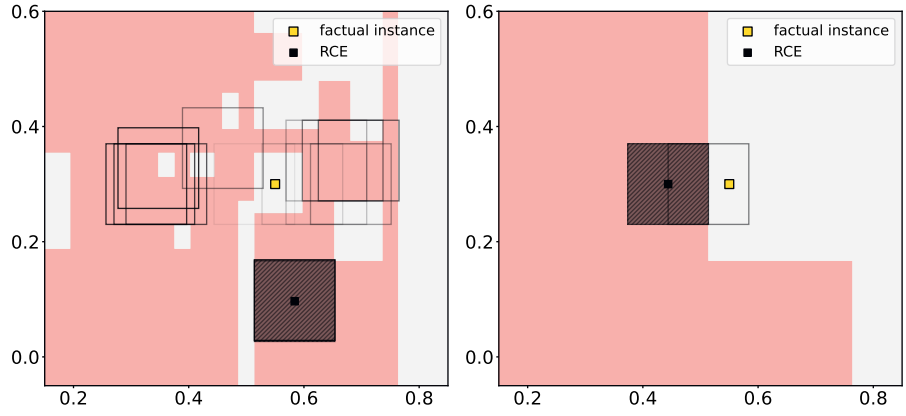
(*a*) Overall computation time for 20 instances

(*b*) Computation time per iteration for 2 instances

Figure 6.9. Visual analysis of the computation time required to solve the MP and AP. The trained model is a GBM with 100 estimators. On the left, we show the computation times for MP and AP summed over all iterations, for each of 20 instances. On the right, we have visualized the computation times in each iteration of two selected instances.

the decision-makers still have the chance to select CEs from a region, albeit slightly smaller than intended. Overall the algorithm converges relatively quickly, however, our experiments show that it converges slower when using ℓ_2-norm as uncertainty set; see Table 6.2. Furthermore, as expected, for a smaller radius of $\rho = 0.01$ we reach the global optimum faster. Figure 6.11 shows the convergence behavior of various predictive models trained on the Diabetes dataset and evaluated on a specific instance. Although the distance to the factual instance (solid line) follows a monotonic increasing trend, the robustness (dashed line) exhibits both peaks and troughs. This behavior can be attributed to the objective function of (MP), which seeks to minimize the distance between the CE and the factual instance. With each iteration of our algorithm, a new scenario/constraint is introduced to (MP), resulting in a worse objective value (higher), but not necessarily an improvement in robustness.

In the latter part of the experiment, we train a neural network with one hidden layer containing 50 nodes using the same three datasets. For the Ionosphere dataset, we also trained another neural network with one hidden layer containing 10 nodes. To generate robust counterfactuals, we used the ℓ_2-norm uncertainty set and tested our algorithm against the one proposed by Dominguez-Olmedo et al. (2021) on 10 instances. In Table 6.3, we report the percentage of times each algorithm was able to find a counterfactual that was at least ρ distant from the decision boundary (robustness) and the percentage of times the counterfactual was valid (validity). We test ρ values of 0.1 and 0.2 for each dataset. Our approach consistently generated counterfactual explanations that were optimal in terms of distance to the factual instance and valid. In the Diabetes dataset, our algorithm reached the time limit without finding a counterfactual with robustness of at least 0.2 in 20% of cases. Nevertheless, the gener-



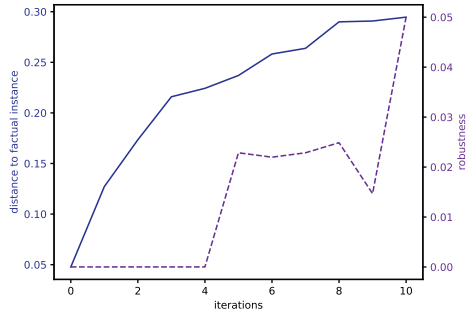
(*a*) Decision tree with max depth = 10 (*b*) Decision tree with max depth = 3

Figure 6.10. Comparison of decision trees with different maximum depths. The decision tree on the right has a maximum depth of 3, while the one on the right has a maximum depth of 10 and is overfitted. The deeper decision tree leads to a higher number of iterations needed to find a robust solution, as shown by the larger number of boxes/solutions generated before converging.

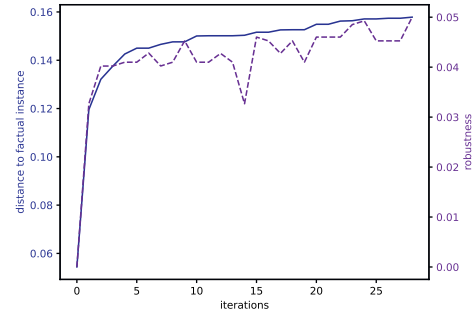
ated counterfactuals were still valid, with a ρ value smaller than 0.2. In contrast, the algorithm proposed by Dominguez-Olmedo et al. (2021) often returned solutions that were not entirely robust with respect to ρ, particularly with the increase in ρ and the complexity of the predictive model. Even more concerning, their algorithm generated invalid solutions in 10% of the cases for the IONOSPHERE dataset. Note that our experiments only cover neural network models since the method in (Dominguez-Olmedo et al. 2021) cannot be used for non-differentiable predictive models.

	Dominguez-Olmedo et al. (2021)		Our algorithm	
ρ	**robustness**	**validity**	**robustness**	**validity**
		BANKNOTE NN(50)		
0.1	80%	100%	100%	100%
0.2	0%	100%	100%	100%
		DIABETES NN(50)		
0.1	60%	100%	100%	100%
0.2	0%	100%	80%	100%
		IONOSPHERE NN(10)		
0.1	70%	90%	100%	100%
0.2	40%	90%	100%	100%
		IONOSPHERE NN(50)		
0.1	50%	100%	100%	100%
0.2	30%	100%	100%	100%

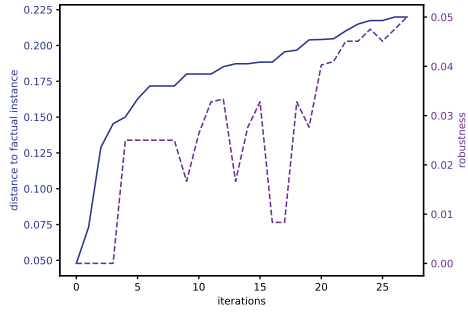
Table 6.3. Comparison between (Dominguez-Olmedo et al. 2021) and our algorithm regarding the percentage of times each algorithm was able to find a counterfactual at least ρ distant from the decision boundary, and the percentage of times the counterfactual explanations were valid and therefore resulting in a flip in the prediction.



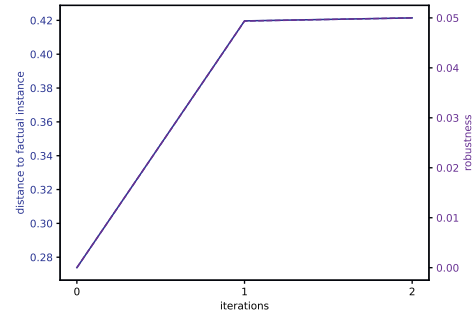
(a) Decision Tree



(b) Random Forest



(c) Gradient Boosting Machines



(d) Neural Network

Figure 6.11. Convergence plots for DT with max depth 10 (a), RF with 50 trees (b), GBM with 50 trees (c), and NN with one hidden layer containing 100 nodes (d). The models are trained on the Diabetes dataset and the uncertainty set is the ℓ_∞-norm.

6.5 Discussion and future work

In this paper, we propose a robust optimization approach for generating regions of CEs for tree-based models and neural networks. We have also shown theoretically that our approach converges. This result has also been supported by empirical studies on different datasets. Our experiments demonstrate that our approach is able to generate explanations efficiently on a variety of datasets and ML models. Our results indicate that the proposed method scales well with the number of features and that the main computational challenge lies in solving the master problem as it becomes larger with every iteration. Overall, our results suggest that our proposed approach is a promising method for generating robust CEs for a variety of ML models.

As part of our future work, we plan to investigate methods for speeding up the calculations of the master problem. One approach involves exploring more efficient formulations of predictive models, such as the one proposed by Parmentier and Vidal (2021) for tree-based models. We also plan to focus on tightening the big M formula-

tions in the future. For instance, this can be achieved for tree-based models by solving a maximum violation problem per node.

We also aim to evaluate the user perception of the robust CEs generated by our approach and investigate how the choice of the uncertainty set affects the quality of the solution. Another research direction is the implementation of categorical and immutable features into the model. In the current version, categorical features are considered immutable, following user preferences. Adhering to user preferences by keeping some feature values constant, whether they are categorical or numerical, offers a practical strategy for generating sparse counterfactual explanations. This approach assumes that only a subset of the features (the ones that are not considered immutable) is susceptible to perturbations. Considering mutable categorical features would require a reformulation of the uncertainty set to account for the different feature types. This would lead to non-convex, discrete, and lower-dimensional uncertainty sets. However, this change would only affect the adversarial problem, while the master problem remains the same.

APPENDIX

A How to choose the robustness budget

When determining the robustness budget ρ, it is essential to weigh the balance between the distance of the resulting CE to the factual instance and its robustness against small perturbations. When the underlying distribution of the CE perturbations is known, we can select ρ such that a certain probabilistic guarantee is satisfied. However, in many real-world situations, we do not have access to distributional information regarding the perturbations. In this case, decision-makers can derive a Pareto front based on the CE's proximity to the factual instance and its robustness against changes.

A.1 Probability guarantee

Probability guarantees regarding the invalidation of recourse depend on both the specific predictive model and the type of uncertainty set in use. For a comprehensive overview of robust reformulations with probabilistic guarantees in the context of linear models, we refer to (Ben-Tal et al. 2009). In this section, we present a more generic formulation that is applicable to both linear and nonlinear models. However, it should be noted that it is a more conservative approach since it only considers the probability that the realized recourse will be in the uncertainty set while (large) part(s) of the uncertainty set may not be near the decision boundary, depending on the model structure.

If we assume that the realized/actual perturbation $\hat{\boldsymbol{s}}$ is distributed according to $\hat{\boldsymbol{s}} \sim N(0, \sigma^2 I)$, where I is the identity matrix, then we can approach the probabilistic guarantee from three angles:

1. What is the probability (α) that the realized recourse is in the defined uncertainty set $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^n \mid ||\boldsymbol{s}|| \leq \rho\}$, given ρ and σ?

2. What value of ρ is required to have α probability that the realized recourse is in the uncertainty set $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^n \mid ||\boldsymbol{s}|| \leq \rho\}$, given α and σ?

3. How large can the standard deviation of the perturbations (σ) be to have α probability of the realized recourse being in the uncertainty set $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^n \mid ||\boldsymbol{s}|| \leq \rho\}$, given α and ρ?

These answers depend on the chosen norm for the uncertainty set. Note that a prediction region with probability α for $\hat{\boldsymbol{s}}$ is given by $\tilde{\mathcal{S}} = \{\boldsymbol{s} \in \mathbb{R}^n \mid \boldsymbol{s}^T \boldsymbol{s} \leq \sigma^2 \chi^2_k(\alpha)\}$ (Chew

1966), such that for the l_2 norm the questions above boil down to

$$\alpha = F_{\chi^2_k}\left(\frac{\rho^2}{\sigma^2}\right), \tag{68}$$

$$\rho = \sigma\sqrt{\chi^2_k(\alpha)}, \tag{69}$$

$$\sigma = \rho/\sqrt{\chi^2_k(\alpha)}, \tag{70}$$

where $\chi^2_k(\alpha)$ and is the quantile-function for the probability α for the chi-square distribution with k degrees of freedom, k is the dimension (number of features) of $\hat{\boldsymbol{s}}$, and $F_{\chi^2_k}$ is the cumulative distribution function for the chi-square distribution with k degrees of freedom.

For the l_∞-norm, first note that $\hat{\boldsymbol{s}}$ is a vector of k i.i.d. $N(0, \sigma^2)$ variables. The probability that the realized recourse is in the defined uncertainty set for this norm is then equal to the probability that all k variables are between $-\rho$ and ρ, i.e.: $P(\hat{\boldsymbol{s}} \in \mathcal{S}) = [\Phi(\rho/\sigma) - \Phi(-\rho/\sigma)]^k = [2\Phi(\rho/\sigma) - 1]^k$, where $\Phi(x)$ represents the standard normal cumulative distribution function. Equating $P(\hat{\boldsymbol{s}} \in \mathcal{S})$ to α then yields

$$\alpha = \left[2\Phi\left(\frac{\rho}{\sigma}\right) - 1\right]^k, \tag{71}$$

$$\rho = \sigma\Phi^{-1}\left(\frac{\alpha^{1/k} + 1}{2}\right), \tag{72}$$

$$\sigma = \rho/\Phi^{-1}\left(\frac{\alpha^{1/k} + 1}{2}\right), \tag{73}$$

where $\Phi^{-1}(x)$ represents the inverse of the standard normal cumulative distribution function.

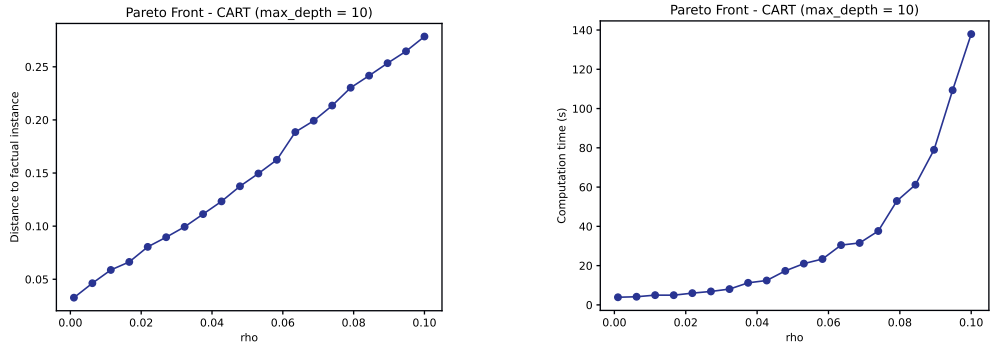
A.2 Pareto front

The Pareto front represents a set of CE solutions that cannot be improved in one objective without degrading performance in the other. In our context:

- Objective 1: Proximity of the CE to the Factual Instance – This measures the ℓ_1-norm distance between the CE and the factual instance.
- Objective 2: Robustness (ρ) - This quantifies the robustness of the CE against perturbations.

In Figure 12a, we report the Pareto front for a decision tree with a max depth of 10 and ℓ_∞-norm uncertainty set. The outcomes are averaged across 20 different cases. When we increase the value of ρ, the distance to the factual instance also increases, occasion-

ally experiencing jumps, see for example the jump in distance for values of ρ around 0.06, which can be attributed to changes in the leaf nodes. Also, the computation time increases with ρ given the increasing difficulty in finding a region that is large enough to contain the ρ-robust CE, see Figure 12b.



(a) (b)

Figure 12. Pareto fronts obtained comparing (a) distance to the factual instance (y-axis) and ρ (x-axis) and (b) computation time required to find a robust counterfactual (y-axis) and ρ (x-axis). The results reported are averaged over 20 factual instances and obtained using a decision tree with a max-depth of 10 trained on the diabetes dataset.

B Linear models

Although Algorithm 1 could still be used for the linear models, there is a well-known easier, and more efficient dual approach to solve model (3)-(4). We review this approach briefly here for the completeness of our discussion.

In the case of linear models, such as logistic regression (LR) or linear support vector machines (SVM), the validity constraint (4) can be formulated as

$$\boldsymbol{\beta}^\top(\boldsymbol{x}+\boldsymbol{s})+\beta_0 \geq \tau, \quad \forall \boldsymbol{s} \in \mathcal{S}, \tag{74}$$

where $\boldsymbol{\beta} \in \mathbb{R}^n$ is the coefficient vector and $\beta_0 \in \mathbb{R}$ is the intercept. Then, these constraints can be equivalently reformulated as

$$\boldsymbol{\beta}^\top \boldsymbol{x} + \beta_0 + \min_{\boldsymbol{s}\in\mathcal{S}} \beta^\top \boldsymbol{s} \geq \tau.$$

Since $\mathcal{S}$ is given in the form (5), the latter is equivalent to

$$\boldsymbol{\beta}^\top \boldsymbol{x} - \rho||\boldsymbol{\beta}||^* + \beta_0 \geq \tau, \tag{75}$$

where $\|\cdot\|^*$ is the dual norm of the norm used in the definition of $\mathcal{S}$. The constant term $\rho||\boldsymbol{\beta}||^*$ ensures that the constraint (74) holds for all $\boldsymbol{s} \in \mathcal{S}$. Note that constraint (75) remains linear in $\boldsymbol{x}$ independently of $\mathcal{S}$. For more details see, *e.g.*, Dominguez-Olmedo et al. (2021), Bertsimas et al. (2019), Xu et al. (2008).

C Proof of Lipschitz continuity of neural networks

Suppose that we have a trained ℓ-layer neural network constructed with ReLU activation functions. If we denote the resulting functional by $f : \mathbb{R}^{n_0} \mapsto \mathbb{R}^{n_\ell}$, then we can write

$$f(\boldsymbol{x}) = \sigma_\ell(W_\ell \sigma_{\ell-1}(W_{\ell-1}\sigma_{\ell-2}\dots W_2\sigma_1(W_1\boldsymbol{x})\dots)), \tag{76}$$

where $\sigma_m : \mathbb{R}^{n_m} \mapsto \mathbb{R}^{n_m}$, $m = 1, \dots, \ell$, stands for the vectorized ReLU functions, and $W_m \in \mathbb{R}^{n_m} \times \mathbb{R}^{n_{m-1}}$, for $m = 1, \dots, \ell$, are the weight matrices. This shows that f is simply the composition of linear and component- as well as piece-wise linear functions.

Given any two Lipschitz continuous functions $g : \mathbb{R}^p \mapsto \mathbb{R}^q$ and $h : \mathbb{R}^q \mapsto \mathbb{R}^s$ with respective Lipschitz constants L_g and L_h, the composition $h \circ g : \mathbb{R}^p \mapsto \mathbb{R}^p$ is Lipschitz continuous with Lipschitz constant $L_h L_g$. This simply follows from observing for a pair of vectors $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^p$ that

$$\|h \circ g(\boldsymbol{y}) - h \circ g(\boldsymbol{z})\| \leq L_h \|g(\boldsymbol{y}) - g(\boldsymbol{z})\| \leq L_h L_g \|\boldsymbol{y} - \boldsymbol{z}\|. \tag{77}$$

Next, for any $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{n_m}$, we have

$$\|W_m \boldsymbol{u} - W_m \boldsymbol{v}\| \leq \|W_m\|_s \|\boldsymbol{u} - \boldsymbol{v}\|, \tag{78}$$

where $\|\cdot\|_s$ is the spectral norm. As the Lipschitz constant for any vectorized ReLU function is one, we obtain the desired result by

$$\|f(\bar{\boldsymbol{x}}) - f(\tilde{\boldsymbol{x}})\| \leq \prod_{m=1}^{\ell} \|W_m\|_s \|\bar{\boldsymbol{x}} - \tilde{\boldsymbol{x}}\| \tag{79}$$

for any pair $\bar{\boldsymbol{x}}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{n_0}$.

D Performances of predictive models

	LR	CART			RF					GBM					NN			
		3	5	10	5	10	20	50	100	5	10	20	50	100	(10)	(10, 10, 10)	(50)	(100)
Banknote																		
Train	0.97	0.94	0.98	1.00	0.96	0.97	0.96	0.97	0.97	0.99	0.99	1.00	1.00	1.00	0.98	0.99	1.00	1.00
Test	0.96	0.89	1.00	1.00	0.96	0.93	0.93	0.96	0.93	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Diabetes																		
Train	0.77	0.77	0.84	0.97	0.77	0.77	0.79	0.80	0.80	0.80	0.83	0.87	0.93	0.99	0.78	0.79	0.80	0.82
Test	0.72	0.77	0.72	0.69	0.72	0.69	0.69	0.67	0.71	0.59	0.67	0.69	0.59	0.62	0.69	0.64	0.73	0.77
Ionosphere																		
Train	0.88	0.93	0.97	1.00	0.93	0.95	0.94	0.96	0.96	0.99	1.00	1.00	1.00	1.00	0.98	0.99	0.99	0.99
Test	0.83	0.88	0.83	0.83	0.88	0.92	0.92	0.92	0.92	0.88	0.88	0.92	0.92	0.92	0.89	0.92	0.88	0.88

Table 4. Train and test accuracy scores of the predictive models used for the experiments.

Table 4 displays the accuracy score of each predictive model employed in the experiments, with their performance being reported for both the training and testing sets.

Chapter 7

Conclusion

7.1 The implications of OCL

In the era of big data, where data generation and collection are priorities for companies and organizations, this thesis provides researchers and practitioners with tools to exploit data in designing accurate and personalized mathematical models. We introduce a framework called Optimization with Constraint Learning (OCL), aimed at training ML models to approximate unknown functions within mathematical optimization models. These predictive models are integrated into optimization models using MIO formulations, preserving model complexity. The works presented in this thesis have already sparked numerous subsequent studies investigating the use of OCL in three different areas: (i) solving complex mathematical optimization, (ii) applying a data-driven approach to formulate optimization models characterized by unknown functions, and (iii) enhancing the OCL framework itself.

First, OCL can be used to tackle important challenges in mathematical optimization. For instance, Bertsimas and Margaritis (2023) use OCL in global optimization to approximate nonconvex constraints using ML models trained on generated data. Their results show that an OCL approach often leads to superior solutions compared to state-of-the-art solvers like BARON. On a different note, Dumouchelle et al. (2024) apply OCL to bilevel optimization problems (and to two-stage robust optimization), approximating the leader's or follower's value function with neural networks resulting in significantly reduced optimization time and minimal gaps in solution optimality.

Second, OCL can be applied to design optimization models with unknown constraints. For example, in energy optimization problems some constraints are complex to model and often present intractable nonlinear relationships between variables. In this context, Dolányi et al. (2024) use neural networks to model electricity market dynamics for optimal bid decisions. Sang et al. (2024) employ neural networks to model carbon emission flows in energy management. A successful demonstration of leveraging OCL in practice is given by Papalexopoulos et al. (2023). The authors designed a framework to design efficient, equitable, and inclusive deceased-donor organ allocation policies. In March 2023, a lung allocation policy designed using the authors' methodology was implemented in the US, leading to a reduction in waiting list mortality by approxi-

mately 20%.

Third, the broad applicability of OCL has gained interest from multiple research groups aiming to enhance it from various perspectives. To improve the robustness of the solutions generated via OCL, Zhu and Burer (2024) propose a new definition of trust region using a convex hull in an extended space. Wang et al. (2023) pursue another approach using an ensemble of neural networks which would lead to better performances of the learned constraints. The authors also suggest preprocessing steps to tighten the bounds of critical neurons to speed up the optimization process. In this regard, Cacciola et al. (2023) recommend pruning neural networks to reduce computational complexity, demonstrating a significant decrease in computation time without compromising decision quality. On a higher level, Davidovich et al. (2022) present a decision-optimization (DO) framework for evaluating OCL frameworks beyond the single optimal solution. The DO framework generates multiple random optimization problems, computes the optimal solutions using the OCL framework, and compares these results with the ground truth to derive a comprehensive prediction profile of the framework. Other works, such as Zhang et al. (2023), investigate efficient embeddings of previously unexplored ML models, such as graph neural networks (GNNs). The authors show how GNNs can be formulated using mixed-integer optimization, even when the input graph includes decision-variable edges.

The increasing adoption of OCL in practice is facilitated by the growing availability of software supporting constraint learning, such as OptiCL (Maragno and Wiberg 2021), OMLT (Ceccon et al. 2022), and JANOS (Bergman et al. 2022). Established optimization solvers like Gurobi and SCIP now also offer built-in options to design constraints and objective functions learned from data.

7.2 Limitations and future research

The field of OCL is still young and necessitates further research to promote its adoption in practice. The following are potential research topics that could be focused on in the coming years.

First, OCL needs innovative methods to improve robustness to the uncertainty of learning from data. Current approaches, such as using convex hulls, help avoid poor performance of embedded ML models. However, these methods are not ML model-specific and can be overly conservative. A promising direction is to develop ML models that include confidence intervals in their predictions, which can then be integrated to define robustness.

Second, the computational complexity of embedding ML models as constraints or objectives in optimization models requires further investigation. This thesis looks into ML models that are MIO-representable to maintain tractable optimization complexity.

However, the same predictive models can be embedded in various ways. For example, decision trees can be represented using one binary variable for each leaf as shown in Chapter 2 or one binary variable for each node (Kudła and Pawlak 2018). In neural networks, efficiency can be improved using cutting planes techniques (Anderson et al. 2019, Tsay et al. 2021) or by tightening bounds (Grimstad and Andersson 2019, Fischetti and Jo 2018). On a similar note, more research is needed to explore whether other types of ML models can also be efficiently formulated. Another factor influencing the computational complexity is the number of variables that serve as features in the ML models. ML models trained on a large number of features can significantly increase the computational burden of the optimization model. A potential solution to this problem is *dimensionality reduction*, which involves creating new features by combining existing ones to reduce the overall number of features in the ML model. A key aspect of feature engineering within an OCL framework is ensuring that these features can be efficiently represented as constraints in the optimization model. Future research could focus on designing features that can be seamlessly integrated into an MIO formulation.

Third, ensuring that variables are causally related and that no unobserved confounders are excluded from the decision variables is crucial in prescriptive analytics. To date, research has assumed the absence of unobserved confounders, which is often a simplification. Future research could incorporate causal inference methods to relax this assumption and better address the presence of unobserved confounders. These three points could be addressed simultaneously by focusing on the selection of predictive models used in constraint learning. Until now, the ML models employed in OCL were already available in the literature, well-known among practitioners, and primarily designed for predictive performance. However, I propose that the OR and ML communities should work on designing new predictive models specifically for OCL applications, as these models require additional characteristics such as robustness to data uncertainty, efficiency in embedding and optimization, explainability, and causal validity.

Fourth, this thesis assumes that the data needed to train the predictive model is already available. However, when data is unavailable but can be generated, a key research question arises: which data points should be evaluated/simulated? Generating the right data points is essential when the process is costly and/or time-consuming. For instance, in the WFP example, collecting data on people's preferences for different diets is a complex task. It involves designing various food baskets and gathering opinions from numerous individuals to obtain a meaningful palatability score. In some cases, data can be collected in multiple stages. This data collection process can be strategically designed to enhance the performance of the ML model and its effectiveness once integrated into the optimization model. Future research could examine the performance of the existing design of experiment methods and explore new approaches specifically tailored to the OCL framework.

Fifth, OCL can be applied to address highly complex combinatorial optimization problems, such as the simultaneous facility location and vehicle routing problem (VRP). In this scenario, the cost associated with the routing problem can be approximated using the output of a predictive model. The training dataset for this model would consist of feasible solutions to the facility location problem, with each label representing the routing cost obtained by solving the VRP for fixed facility locations. Determining which data points to generate is the critical challenge. More generally, OCL offers an alternative approach to tackling complex problems that involve both strategic (long-term) and operational (short-term) decision variables. By approximating either the long-term or short-term effects, OCL can reduce the number of decision variables and simplify the overall complexity of the optimization problem. Similarly, OCL can be beneficial in simulation-based optimization, where evaluating the quality of a solution through simulation can be resource-intensive. Integrating the OCL framework with simulation can significantly reduce the number of simulations needed to obtain good solutions.

Finally, I argue that each decision science field could evaluate whether OCL drives advancements. This thesis demonstrates how two distinct fields—radiotherapy optimization and explainable AI—can both benefit from OCL. The OCL framework offers a promising approach in areas where personalization is crucial, such as healthcare treatments or pricing, and where functions are difficult to explicitly define, yet data is available or can be generated, such as hyperparameter tuning. On a higher level, OCL has the potential to democratize optimization for decision-making, enabling practitioners without mathematical expertise to formulate optimization models. Optimization democratization has always been a challenge in OR due to the high entry barriers and the significant costs associated with designing optimization models. Recent advancements in large language models (LLMs) have sparked interest among researchers and companies in using these models to formulate mathematical models from problem descriptions (AhmadiTeshnizi et al. 2023, Tsouros et al. 2023). However, LLMs face significant challenges in generating accurate and efficient optimization models, often introducing subtle errors that non-experts find difficult to detect (Wasserkrug et al. 2024). On the other hand, the OCL framework offers a data-centric approach where each objective function and constraint identified by the domain expert can be learned using available or to-be-generated data. This approach shifts the focus from the highly technical and complex design of mathematical models to the more intuitive and familiar process of generating representative data and training accurate predictive models. Additionally, OCL presents an innovative method for teaching mathematical optimization in schools, starting with data and predictive models and progressing toward optimization modeling. This approach highlights the close and mutually reinforcing relationship between predictive and prescriptive analytics which has been a recurrent topic of this thesis.

Bibliography

Abbasi, B., Babaei, T., Hosseinifard, Z., Smith-Miles, K., and Dehghani, M. (2020). Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management. *Computers & Operations Research*, 119:104941.

Ahiska, S. and King, R. E. (2010). Inventory optimization in a one product recoverable manufacturing system. *International Journal of Production Economics*, 124(1):11–19.

AhmadiTeshnizi, A., Gao, W., and Udell, M. (2023). Optimus: Optimization modeling using mip solvers and large language models. *arXiv preprint arXiv:2310.06116*.

Ajdari, A., Liao, Z., Mohan, R., Wei, X., and Bortfeld, T. (2022). Personalized mid-course fdg-pet based adaptive treatment planning for non-small cell lung cancer using machine learning and optimization. *Physics in Medicine & Biology*, 67(18):185015.

Altshuler, Y., Shmueli, E., Zyskind, G., Lederman, O., Oliver, N., and Pentland, A. (2014). Campaign optimization through behavioral modeling and mobile network analysis. *IEEE Transactions on Computational Social Systems*, 1(2):121–134.

Amos, B., Xu, L., and Kolter, J. Z. (2017). Input convex neural networks. *Proceedings of the 34th International Conference on Machine Learning*, 70:146–155.

Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., and Vielma, J. P. (2020). Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1-2):3–39.

Anderson, R., Huchette, J., Tjandraatmadja, C., and Vielma, J. P. (2019). Strong mixed-integer programming formulations for trained neural networks. *arXiv preprint arXiv:1811.08359*.

Artelt, A., Vaquet, V., Velioglu, R., Hinder, F., Brinkrolf, J., Schilling, M., and Hammer, B. (2021). Evaluating robustness of counterfactual explanations. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–09.

Athey, S. and Imbens, G. (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360.

Bagloee, S. A., Asadi, M., Sarvi, M., and Patriksson, M. (2018). A hybrid machine-learning and optimization method to solve bi-level problems. *Expert Systems with Applications*, 95:142–152.

Baldomero-Naranjo, M., Martínez-Merino, L. I., and Rodríguez-Chía, A. M. (2020). Tightening big Ms in integer programming formulations for support vector machines with ramp loss. *European Journal of Operational Research*, 286(1):84–100.

Balestriero, R., Pesenti, J., and LeCun, Y. (2021). Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*.

Baumann, M., Krause, M., Overgaard, J., Debus, J., Bentzen, S. M., Daartz, J., Richter, C., Zips, D., and Bortfeld, T. (2016). Radiation oncology in the era of precision medicine. *Nature Reviews. Cancer*, 16:234–249.

Beldiceanu, N. and Simonis, H. (2016). Modelseeker: Extracting global constraint models from positive examples. pages 77–95. Springer.

Belenguer, L. (2022). AI bias: exploring discriminatory algorithmic decision-making models and the application of possible machine-centric solutions adapted from the pharmaceutical industry. *AI and Ethics*, 2:771–787.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization*. Princeton University Press.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Bergman, D., Huang, T., Brooks, P., Lodi, A., and Raghunathan, A. U. (2022). JANOS: an integrated predictive and prescriptive modeling framework. *INFORMS Journal on Computing*, 34(2):807–816.

Bertsimas, D., Borenstein, A., Mingardi, L., Nohadani, O., Orfanoudaki, A., Stellato, B., Wiberg, H., Sarin, P., Varelmann, D. J., Estrada, V., Macaya, C., and Gil, I. J. (2021). Personalized prescription of ACEI/ARBs for hypertensive COVID-19 patients. *Health Care Management Science*, 24(2):339–355.

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106:1039–1082.

Bertsimas, D., Dunn, J., Pawlowski, C., and Zhuo, Y. D. (2019). Robust classification. *INFORMS Journal on Optimization*, 1(1):2–34.

Bertsimas, D., Dunning, I., and Lubin, M. (2016a). Reformulation versus cutting-planes for robust optimization: A computational study. *Computational Management Science*, 13:195–217.

Bertsimas, D. and Kallus, N. (2020). From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044.

Bertsimas, D. and Margaritis, G. (2023). Global optimization: A machine learning approach. *arXiv preprint arXiv:2311.01742*.

Bertsimas, D., McCord, C., and Sturt, B. (2023). Dynamic optimization with side information. *European Journal of Operational Research*, 304(2):634–651.

Bertsimas, D., O'Hair, A., Relyea, S., and Silberholz, J. (2016b). An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531.

Bertsimas, D., Silberholz, J., and Trikalinos, T. (2018). Optimal healthcare decision making under multiple mathematical models: application in prostate cancer screening. *Health Care Management Science*, 21(1):105–118.

Bertsimas, D. and Dunn, J. (2018). *Machine Learning under a Modern Optimization Lens*. Dynamic Ideas, Belmont.

Bessiere, C., Coletta, R., Freuder, E. C., and O'Sullivan, B. (2004). Leveraging the learning power of examples in automated constraint acquisition. *International Conference on Principles and Practice of Constraint Programming*, pages 123–137.

Bienstock, D. and Özbay, N. (2008). Computing robust basestock levels. *Discrete Optimization*, 5(2):389–414.

Biggs, M., Hariss, R., and Perakis, G. (2021). Optimizing objective functions determined from random forests. *SSRN Electronic Journal*, pages 1–46. Available at SSRN: http://www.ssrn.com/abstract=2986630.

Biggs, M., Hariss, R., and Perakis, G. (2023). Constrained optimization of objective functions determined from random forests. *Production and Operations Management*, 32(2):397–415.

Birge, J. R. and Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer Science & Business Media.

Black, E., Wang, Z., Fredrikson, M., and Datta, A. (2021). Consistent counterfactuals for deep models. *arXiv preprint arXiv:2110.03109*.

Bonfietti, A., Lombardi, M., and Milano, M. (2015). Embedding decision trees and random forests in constraint programming. *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 74–90.

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

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Routledge.

Bui, N., Nguyen, D., and Nguyen, V. A. (2022). Counterfactual plans under distributional ambiguity. *International Conference on Learning Representations*.

Cacciola, M., Frangioni, A., and Lodi, A. (2023). Structured pruning of neural networks for constraints learning. *arXiv preprint arXiv:2307.07457*.

Cancer Therapy Evaluation Program (2006). Common terminology criteria for adverse events v3.0.

Cao, W., Wang, X., Ming, Z., and Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing*, 275:278–287.

Carrizosa, E., Molero-Río, C., and Romero Morales, D. (2021). Mathematical optimization in classification and regression trees. *Top*, 29(1):5–33.

Carrizosa, E., Ramírez-Ayerbe, J., and Romero Morales, D. (2024). Generating collective counterfactual explanations in score-based classification via mathematical optimization. *Expert Systems with Applications*, 238:121954.

Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C. D., and Misener, R. (2022). Omlt: Optimization & machine learning toolkit. *Journal of Machine Learning Research*, 23(349):1–8.

Chaikh, A., Docquière, N., Bondiau, P.-Y., and Balosso, J. (2016). Impact of dose calculation models on radiotherapy outcomes and quality adjusted life years for lung cancer treatment: do we need to measure radiotherapy outcomes to tune the radiobiological parameters of a normal tissue complication probability model? *Translational Lung Cancer Research*, 5(6):673–680.

Chatzivasileiadis, S. (2020). From decision trees and neural networks to milp: Power system optimization considering dynamic stability constraints. *2020 European Control Conference (ECC)*, pages 594–594.

Chen, S., Zhou, S., Yin, F.-F., Marks, L. B., and Das, S. K. (2007). Investigation of the support vector machine algorithm to predict lung radiation-induced pneumonitis. *Medical Physics*, 34(10):3808–3814.

Chen, X., Wang, Y., and Zhou, Y. (2020a). Dynamic assortment optimization with changing contextual information. *Journal of Machine Learning Research*, 21(216):1–44.

Chen, Y., Shi, Y., and Zhang, B. (2020b). Input convex neural networks for optimal voltage regulation. *arXiv preprint arXiv:2002.08684*.

Chew, V. (1966). Confidence, prediction, and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*, 61(315):605–617.

Chi, H.-M., Ersoy, O. K., Moskowitz, H., and Ward, J. (2007). Modeling and optimizing a vendor managed replenishment system using machine learning and genetic algorithms. *European Journal of Operational Research*, 180(1):174–193.

Chiang, M., Low, S. H., Calderbank, A. R., and Doyle, J. C. (2007). Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312.

Cho, M. K. (2021). Rising to the challenge of bias in health care ai. *Nature Medicine*, 27:2079–2081.

Conn, A., Scheinberg, K., and Vicente, L. (2009). Introduction to derivative-free optimization. *MPS-SIAM Series on Optimization*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cozad, A., Sahinidis, N. V., and Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227.

CPLEX, IBM ILOG (2009). V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157.

Cremer, J. L., Konstantelos, I., Tindemans, S. H., and Strbac, G. (2018). Data-driven power system operation: Exploring the balance between cost and risk. *IEEE Transactions on Power Systems*, 34(1):791–801.

Crombecq, K., Laermans, E., and Dhaene, T. (2011). Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *European Journal of Operational Research*, 214(3):683–696.

Dandl, S., Molnar, C., Binder, M., and Bischl, B. (2020). Multi-objective counterfactual explanations. *Parallel Problem Solving from Nature – PPSN XVI*, pages 448–469.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

Das, S. K., Chen, S., Deasy, J. O., Zhou, S., Yin, F.-F., and Marks, L. B. (2008). Combining multiple models to generate consensus: Application to radiation-induced pneumonitis prediction. *Medical Physics*, 35(11):5098–5109.

Das, S. K., Zhou, S., Zhang, J., Yin, F.-F., Dewhirst, M. W., and Marks, L. B. (2007). Predicting lung radiotherapy-induced pneumonitis using a model combining parametric lyman probit with nonparametric decision trees. *International Journal of Radiation Oncology∗Biology∗Physics*, 68(4):1212–1221.

Davidovich, O., Bercea, G.-T., and Wasserkrug, S. (2022). The good, the bad, and the outliers: A testing framework for decision optimization model learning. page 2811–2821.

De Angelis, V., Felici, G., and Impelluso, P. (2003). Integrating simulation and optimisation in health care centre management. *European Journal of Operational Research*, 150(1):101–114.

de Mast, J., Steiner, S. H., Nuijten, W. P. M., and Kapitan, D. (2022). Analytical problem solving based on causal, correlational and deductive models. *The American Statistician*, pages 1–11.

De Raedt, L., Passerini, A., and Teso, S. (2018). Learning constraints from examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Demirovic, E., Stuckey, P. J., Bailey, J., Chan, J., Leckie, C., Ramamohanarao, K., and Guns, T. (2019). Predict+ optimise with ranking objectives: Exhaustively learning linear functions. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1078–1085.

den Hertog, D. and Stehouwer, H. (2002). Optimizing color picture tubes by high-cost nonlinear programming. *European Journal of Operational Research*, 140(2):197–211.

Deng, Y., Liu, J., and Sen, S. (2018). Coalescing data and decision sciences for analytics. pages 20–49. INFORMS.

Dolányi, M., Bruninx, K., Toubeau, J.-F., and Delarue, E. (2024). Capturing electricity market dynamics in strategic market participation using neural network constrained optimization. *IEEE Transactions on Power Systems*, 39(1):533–545.

Dominguez-Olmedo, R., Karimi, A.-H., and Schölkopf, B. (2021). On the adversarial robustness of causal algorithmic recourse. *arXiv preprint arXiv:2112.11313*.

Donti, P., Amos, B., and Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. *Advances in Neural Information Processing Systems*, pages 5484–5494.

Drucker, H., Surges, C. J., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems*, 1:155–161.

Drzymala, R., Mohan, R., Brewster, L., Chu, J., Goitein, M., Harms, W., and Urie, M. (1991). Dose-volume histograms. *International Journal of Radiation Oncology*Biology*Physics*, 21(1):71–78.

Dua, D. and Graff, C. (2017). UCI machine learning repository. Available at: `http://archive.ics.uci.edu`.

Dumouchelle, J., Julien, E., Kurtz, J., and Khalil, E. B. (2024). Neur2bilo: Neural bilevel optimization. *arXiv preprint arXiv:2402.02552*.

Dutta, S., Long, J., Mishra, S., Tilli, C., and Magazzeni, D. (2022). Robust counterfactual explanations for tree-based ensembles. *Proceedings of the 39th International Conference on Machine Learning*, 162:5742–5756.

D'andreagiovanni, F. and Gleixner, A. M. (2016). Towards an accurate solution of wireless network design problems. *International Symposium on Combinatorial Optimization*, pages 135–147.

Ebert, T., Belz, J., and Nelles, O. (2014). Interpolation and extrapolation: Comparison of definitions and survey of algorithms for convex and concave hulls. *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 310–314.

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

Fahmi, I. and Cremaschi, S. (2012). Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models. *Computers & Chemical Engineering*, 46:105–123.

Fajemisin, A. O., Maragno, D., and den Hertog, D. (2024). Optimization with constraint learning: A framework and survey. *European Journal of Operational Research*, 314(1):1–14.

Fargeas, A., Acosta, O., Arrango, J. D. O., Ferhat, A., Costet, N., Albera, L., Azria, D., Fenoglietto, P., Créhange, G., Beckendorf, V., Hatt, M., Kachenoura, A., and de Crevoisier, R. (2018). Independent component analysis for rectal bleeding prediction following prostate cancer radiotherapy. *Radiotherapy and Oncology*, 126(2):263–269.

Fargeas, A., Albera, L., Kachenoura, A., Dréan, G., Ospina, J.-D., Coloigner, J., Lafond, C., Delobel, J.-B., Crevoisier, R. D., and Acosta, O. (2015). On feature extraction and classification in prostate cancer radiotherapy using tensor decompositions. *Medical Engineering & Physics*, 37(1):126–131.

Ferrario, A. and Loi, M. (2022). The robustness of counterfactual explanations over time. *IEEE Access*, 10:82736–82750.

Fischetti, M. and Fraccaro, M. (2019). Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. *Computers & Operations Research*, 106:289–297.

Fischetti, M. and Jo, J. (2018). Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309.

Forel, A., Parmentier, A., and Vidal, T. (2022). Robust counterfactual explanations for random forests. *arXiv preprint arXiv:2205.14116*.

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering Design via Surrogate Modelling: a Practical Guide*. Wiley.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Gabryś, H. S., Buettner, F., Sterzing, F., Hauswald, H., and Bangert, M. (2018). Design and selection of machine learning methods using radiomics and dosiomics for normal tissue complication probability modeling of xerostomia. *Frontiers in Oncology*, 8:35.

Galassi, A., Lombardi, M., Mello, P., and Milano, M. (2018). Model agnostic solution of CSPs via deep learning: A preliminary study. *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 254–262.

Gambella, C., Ghaddar, B., and Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828.

GAMS Development Corporation (2021). Disjunctive programming. Available at: `https://www.gams.com/latest/docs/UG_EMP_DisjunctiveProgramming.html`.

Garg, A., Jalali, M., Kekatos, V., and Gatsis, N. (2018). Kernel-based learning for smart inverter control. *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 875–879.

Garud, S. S., Karimi, I. A., and Kraft, M. (2017). Design of computer experiments: A review. *Computers & Chemical Engineering*, 106:71–95.

Gay, H. A. and Niemierko, A. (2007). A free program for calculating eud-based ntcp and tcp in external beam radiotherapy. *Physica Medica*, 23(3):115–125.

Gilan, S. S. and Dilkina, B. (2015). Sustainable building design: A challenge at the intersection of machine learning and design optimization. *AAAI Workshop: Computational Sustainability*, pages 101–106.

Goldfarb, D. and Iyengar, G. (2003). Robust portfolio selection problems. *Mathematics of Operations Research*, 28(1):1–38.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

Gorissen, B. L., Yanıkoğlu, İ., and den Hertog, D. (2015). A practical guide to robust optimization. *Omega*, 53:124–137.

Grave, É., Joulin, A., Cissé, M., Grangier, D., and Jégou, H. (2017). Efficient softmax approximation for GPUs. *Proceedings of the 34th International Conference on Machine Learning*, 70:1302–1310.

Grigoroudis, E. and Phillis, Y. A. (2013). Modeling healthcare system-of-systems: A mathematical programming approach. *IEEE Systems Journal*, 7(4):571–580.

Grimstad, B. and Andersson, H. (2019). Relu networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, 131:106580.

Grossmann, I. E. (2009). Generalized disjunctive programming. *Encyclopedia of Optimization*, pages 1180–1183.

Guidotti, R. (2022). Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55.

Gulliford, S. L., Webb, S., Rowbottom, C. G., Corne, D. W., and Dearnaley, D. P. (2004). Use of artificial neural networks to predict biological outcomes for patients receiving radical radiotherapy of the prostate. *Radiotherapy and Oncology*, 71(1):3–12.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual. Available at: `https://www.gurobi.com`.

Gutierrez-Martinez, V. J., Cañizares, C. A., Fuerte-Esquivel, C. R., Pizano-Martinez, A., and Gu, X. (2010). Neural-network security-boundary constrained optimal power flow. *IEEE Transactions on Power Systems*, 26(1):63–72.

Halilbašić, L., Thams, F., Venzke, A., Chatzivasileiadis, S., and Pinson, P. (2018). Data-driven security-constrained AC-OPF for operations and markets. *2018 Power Systems Computation Conference (PSCC)*, pages 1–7.

Han, B., Shang, C., and Huang, D. (2021). Multiple kernel learning-aided robust optimization: Learning algorithm, computational tractability, and usage in multi-stage decision-making. *European Journal of Operational Research*, 292(3):1004–1018.

Hoffmann, A. L., den Hertog, D., Siem, A. Y., Kaanders, J. H., and Huizenga, H. (2008). Convex reformulation of biologically-based multi-criteria intensity-modulated radiation therapy optimization including fractionation effects. *Physics in Medicine & Biology*, 53(22):6345.

Hottung, A., Tanaka, S., and Tierney, K. (2020). Deep learning assisted heuristic tree search for the container pre-marshalling problem. *Computers & Operations Research*, 113:104781.

IBM (2022). User's Manual for CPLEX. Available at: `https://www.ibm.com/docs/en/icos/12.8.0.0?topic=cplex-users-manual`.

Isaksson, L. J., Pepa, M., Zaffaroni, M., Marvaso, G., Alterio, D., Volpe, S., Corrao, G., Augugliaro, M., Starzyńska, A., Leonardi, M. C., Orecchia, R., and Jereczek-Fossa, B. A. (2020). Machine learning-based models for prediction of toxicity outcomes in radiotherapy. *Frontiers in Oncology*, 10:790.

Ito, S. and Fujimaki, R. (2017). Optimization beyond prediction: Prescriptive price optimization. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1833–1841.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New Phytologist*, 11(2):37–50.

Jalali, M., Kekatos, V., Gatsis, N., and Deka, D. (2019). Designing reactive power control rules for smart inverters using support vector machines. *IEEE Transactions on Smart Grid*, 11(2):1759–1770.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

Jiang, W., Lakshminarayanan, P., Hui, X., Han, P., Cheng, Z., Bowers, M., Shpitser, I., Siddiqui, S., Taylor, R. H., Quon, H., and McNutt, T. (2019). Machine learning methods uncover radiomorphologic dose patterns in salivary glands that predict xerostomia in patients with head and neck cancer. *Advances in Radiation Oncology*, 4(2):401–412.

Jiménez-Cordero, A., Morales, J. M., and Pineda, S. (2021). A novel embedded min-max approach for feature selection in nonlinear support vector machine classification. *European Journal of Operational Research*, 293(1):24–35.

Källman, P., Agren, A., and Brahme, A. (1991). Tumour and normal tissue responses to fractionated non-uniform dose delivery. *International Journal of Radiation Biology*, 62(2):249–262.

Kanamori, K., Takagi, T., Kobayashi, K., and Arimura, H. (2020). DACE: distribution-aware counterfactual explanation by mixed-integer linear optimization. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 2855–2862.

Kanamori, K., Takagi, T., Kobayashi, K., Ike, Y., Uemura, K., and Arimura, H. (2021). Ordered counterfactual explanation by mixed-integer linear optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11564–11574.

Karimi, A.-H., Barthe, G., Balle, B., and Valera, I. (2020a). Model-agnostic counterfactual explanations for consequential decisions. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 108:895–905.

Karimi, A.-H., Schölkopf, B., and Valera, I. (2020b). Algorithmic recourse: from counterfactual explanations to interventions. *arXiv preprint arXiv:2002.06278*.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., and Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422.

Karmelita, M. and Pawlak, T. P. (2020). CMA-ES for one-class constraint synthesis. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 859–867.

Khalil, E., Le Bodic, P., Song, L., Nemhauser, G., and Dilkina, B. (2016). Learning to branch in mixed integer programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30:1.

Kierkels, R. G., Korevaar, E. W., Steenbakkers, R. J., Janssen, T., Van 'T Veld, A. A., Langendijk, J. A., Schilstra, C., and Van Der Schaaf, A. (2014). Direct use of multivariable normal tissue complication probability models in treatment plan optimisation for individualised head and neck cancer radiotherapy produces clinically acceptable treatment plans. *Radiotherapy and Oncology*, 112(3):430–436.

Kierkels, R. G., Wopken, K., Visser, R., Korevaar, E. W., van der Schaaf, A., Bijl, H. P., and Langendijk, J. A. (2016). Multivariable normal tissue complication probability model-based treatment plan optimization for grade 2–4 dysphagia and tube feeding dependence in head and neck radiotherapy. *Radiotherapy and Oncology*, 121(3):374–380.

Kleijnen, J. P. (2015). Design and analysis of simulation experiments. *International Workshop on Simulation*, pages 3–22.

Koch, T., Berthold, T., Pedersen, J., and Vanaret, C. (2022). Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, 10:100031.

Kolb, S. M. (2016). Learning constraints and optimization criteria. *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, pages 403–409.

Kotary, J., Fioretto, F., Hentenryck, P. V., and Wilder, B. (2021). End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Krafft, S. P., Rao, A., Stingo, F., Briere, T. M., Court, L. E., Liao, Z., and Martel, M. K. (2018). The utility of quantitative ct radiomics features for improved prediction of radiation pneumonitis. *Medical Physics*, 45(11):5317–5324.

Kudła, P. and Pawlak, T. P. (2018). One-class synthesis of constraints for Mixed-Integer Linear Programming with C4.5 decision trees. *Applied Soft Computing Journal*, 68:1–12.

Kuhn, M. and Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press.

Kulesza, A. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286.

Kumar, M. et al. (2019a). Acquiring integer programs from data. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 1130–1136.

Kumar, M., Kolb, S., De Raedt, L., and Teso, S. (2021). Learning mixed-integer linear programs from contextual examples. *arXiv preprint arXiv:2107.07136*.

Kumar, M., Teso, S., De Causmaecker, P., and De Raedt, L. (2019b). Automating personnel rostering by learning constraints using tensors. *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 697–704.

Kutcher, C. and Burman, G. (1989). Calculation of complication probability factors for non-uniform normal tissue irradiation: the effective volume method. *International Journal of Radiation Oncology - Biology - Physics*, 16:1623–30.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., and Detyniecki, M. (2017). Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*.

Lee, S., Ybarra, N., Jeyaseelan, K., Faria, S., Kopek, N., Brisebois, P., Bradley, J. D., Robinson, C., Seuntjens, J., and Naqa, I. E. (2015). Bayesian network ensemble as a multivariate strategy to predict radiation pneumonitis risk. *Medical Physics*, 42(5):2421–2430.

Liao, Z., Lee, J. J., Komaki, R., Gomez, D. R., O'Reilly, M. S., Fossella, F. V., Blumenschein, G. R., Heymach, J. V., Vaporciyan, A. A., Swisher, S. G., Allen, P. K., Choi, N. C., DeLaney, T. F., Hahn, S. M., Cox, J. D., Lu, C. S., and Mohan, R. (2018). Bayesian adaptive randomization trial of passive scattering proton therapy and intensity-modulated photon radiotherapy for locally advanced non–small-cell lung cancer. *Journal of Clinical Oncology*, 36(18):1813–1822. PMID: 29293386.

Liu, X. and Li, J. (2016). Patient specific characteristics are an important factor that determines the risk of acute grade ≥ 2 rectal toxicity in patients treated for prostate cancer with IMRT and daily image guidance based on implanted gold markers. *OMICS Journal of Radiology*, 5(3).

Lombardi, M. and Milano, M. (2018). Boosting combinatorial problem modeling with machine learning. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5472–5478.

Lombardi, M., Milano, M., and Bartolini, A. (2017). Empirical decision model learning. *Artificial Intelligence*, 244:343–367.

Luna, J. M., Chao, H.-H., Diffenderfer, E. S., Valdes, G., Chinniah, C., Ma, G., Cengel, K. A., Solberg, T. D., Berman, A. T., and Simone, C. B. (2019). Predicting radiation pneumonitis in locally advanced stage II–III non-small cell lung cancer using machine learning. *Radiotherapy and Oncology*, 133:106–112.

Lyman, J. T. (1985). Complication probability as assessed from dose-volume histograms. *Radiation Research Supplement*, 8:S13–9.

Mahajan, D., Tan, C., and Sharma, A. (2019). Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*.

Maragno, D., Röber, T. E., and Birbil, I. (2022). Counterfactual explanations using optimization with constraint learning. *OPT 2022: Optimization for Machine Learning* (*NeurIPS 2022 Workshop*).

Maragno, D. and Wiberg, H. (2021). Opticl: Mixed-integer optimization with constraint learning. Available at: `https://github.com/hwiberg/OptiCL/`.

Maragno, D., Wiberg, H., Bertsimas, D., Birbil, I., den Hertog, D., and Fajemisin, A. O. (2023). Mixed-integer optimization with constraint learning. *Operations Research*. Available at: `https://doi.org/10.1287/opre.2021.0707`.

Marks, L. B., Bentzen, S. M., Deasy, J. O., Kong, F.-M., Bradley, J. D., Vogelius, I. S., El Naqa, I., Hubbs, J. L., Lebesque, J. V., Timmerman, R. D., Martel, M. K., and Jackson, A. (2010). Radiation dose-volume effects in the lung. *International Journal of Radiation Oncology* Biology* Physics*, 76(3):S70–S76.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58(8):1246–1266.

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc.

Miller, T. (2018). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Mišić, V. V. (2020). Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624.

Molnar, C. (2020). *Interpretable Machine Learning*. Lulu.com. "Available at: `https://christophm.github.io/interpretable-ml-book/`".

MOSEK (2019). *MOSEK Optimizer API for Python 9.3.7*. Available at: `https://docs.mosek.com/latest/pythonapi/index.html`.

Mothilal, R. K., Sharma, A., and Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617.

Mukerjee, R. and Wu, C.-F. (2006). *A Modern Theory of Factorial Design*. Springer.

Mutapcic, A. and Boyd, S. (2009). Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods & Software*, 24(3):381–406.

Nagata, Y. and Chu, K. H. (2003). Optimization of a fermentation medium using neural networks and genetic algorithms. *Biotechnology Letters*, 25(21):1837–1842.

Nahum, A. E. and Uzan, J. (2012). (Radio)biological optimization of external-beam radiotherapy. *Computational and Mathematical Methods in Medicine*, 2012.

Nakatsugawa, M., Cheng, Z., Kiess, A., Choflet, A., Bowers, M., Utsunomiya, K., Sugiyama, S., Wong, J., Quon, H., and McNutt, T. (2019). The needs and benefits of continuous model updates on the accuracy of RT-induced toxicity prediction models within a learning health system. *International Journal of Radiation Oncology∗Biology∗Physics*, 103(2):460–467.

Naqa, I. E., Bradley, J. D., Lindsay, P. E., Hope, A. J., and Deasy, J. O. (2009). Predicting radiotherapy outcomes using statistical learning techniques. *Physics in Medicine and Biology*, 54(18):S9–S30.

Nascimento, C. A. O., Giudici, R., and Guardani, R. (2000). Neural network based approach for optimization of industrial chemical processes. *Computers & Chemical Engineering*, 24(9-10):2303–2314.

National Cancer Institute (2021). Treatment clinical trials for gastric (stomach) cancer. Available at:`https://www.cancer.gov/about-cancer/treatment/clinical-trials/disease/stomach-cancer/treatment`.

Navas-Palencia, G. (2021). Optimal counterfactual explanations for scorecard modelling. *arXiv preprint arXiv:2104.08619*.

Ng, A. (2020). Neural networks and deep learning. Available at: `https://www.coursera.org/specializations/deep-learning`.

Nieuwenhuis, R. and Oliveras, A. (2006). On SAT modulo theories and optimization problems. *International conference on theory and applications of satisfiability testing*, pages 156–169.

Oh, J. H., Kerns, S., Ostrer, H., Powell, S. N., Rosenstein, B., and Deasy, J. O. (2017). Computational methods using genome-wide association studies to predict radiotherapy complications and to identify correlative molecular processes. *Scientific Reports*, 7(1).

Ospina, J. D., Zhu, J., Chira, C., Bossi, A., Delobel, J. B., Beckendorf, V., Dubray, B., Lagrange, J.-L., Correa, J. C., Simon, A., Acosta, O., and de Crevoisier, R. (2014). Random forests to predict rectal toxicity following prostate cancer radiation therapy. *International Journal of Radiation Oncology∗Biology∗Physics*, 89(5):1024–1031.

O'Sullivan, B. (2010). Automated modelling and solving in constraint programming. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence AAAI-10*, 24:1493–1497.

O'Neil, M. and Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, USA.

Padmanabhan, B. and Tuzhilin, A. (2003). On the use of optimization for data mining: Theoretical interactions and ecrm opportunities. *Management Science*, 49(10):1327–1343.

Palma, G., Monti, S., Xu, T., Scifoni, E., Yang, P., Hahn, S. M., Durante, M., Mohan, R., Liao, Z., and Cella, L. (2019). Spatial dose patterns associated with radiation pneumonitis in a

randomized trial comparing intensity-modulated photon therapy with passive scattering proton therapy for locally advanced non-small cell lung cancer. *International Journal of Radiation Oncology, Biology, Physics*, 104(5):1124–1132.

Papalexopoulos, T., Alcorn, J., Bertsimas, D., Goff, R., Stewart, D., and Trichakis, N. (2023). Reshaping national organ allocation policy. *Operations Research*, 72(4):1475–1486.

Parmentier, A. and Vidal, T. (2021). Optimal counterfactual explanations in tree ensembles. *Proceedings of the 38th International Conference on Machine Learning*, 139:8422–8431.

Paulus, A., Rolínek, M., Musil, V., Amos, B., and Martius, G. (2021). Comboptnet: Fit the right np-hard problem by learning integer programming constraints. *International Conference on Machine Learning*, pages 8443–8453.

Pawelczyk, M., Bielawski, S., van den Heuvel, J., Richter, T., and Kasneci, G. (2021). Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783*.

Pawelczyk, M., Datta, T., van-den Heuvel, J., Kasneci, G., and Lakkaraju, H. (2022). Probabilistically robust recourse: Navigating the trade-offs between costs and robustness in algorithmic recourse. *arXiv preprint arXiv:2203.06768*.

Pawlak, T. P. (2019). Synthesis of mathematical programming models with one-class evolutionary strategies. *Swarm and Evolutionary Computation*, 44:335–348.

Pawlak, T. P. and Krawiec, K. (2017a). Automatic synthesis of constraints from examples using mixed integer linear programming. *European Journal of Operational Research*, 261(3):1141–1157.

Pawlak, T. P. and Krawiec, K. (2017b). Synthesis of mathematical programming constraints with genetic programming. *European Conference on Genetic Programming*, pages 178–193.

Pawlak, T. P. and Krawiec, K. (2018). Synthesis of constraints for mathematical programming with one-class genetic programming. *IEEE Transactions on Evolutionary Computation*, 23(1):117–129.

Pawlak, T. P. and Litwiniuk, B. (2021). Ellipsoidal one-class constraint acquisition for quadratically constrained programming. *European Journal of Operational Research*, 293(1):36–49.

Pawlak, T. P. and O'Neill, M. (2021). Grammatical evolution for constraint synthesis for mixed-integer linear programming. *Swarm and Evolutionary Computation*, 64:100896.

Pearl, J. (2013). Structural counterfactuals: a brief introduction. *Cognitive Science*, 37(6):977–985.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pella, A., Cambria, R., Riboldi, M., Jereczek-Fossa, B. A., Fodor, C., Zerini, D., Torshabi, A. E., Cattani, F., Garibaldi, C., Pedroli, G., Baroni, G., and Orecchia, R. (2011). Use of machine learning methods for prediction of acute toxicity in organs at risk following prostate radiotherapy. *Medical Physics*, 38(6Part1):2859–2867.

Peters, K., Silva, S., Gonçalves, R., Kavelj, M., Fleuren, H., den Hertog, D., Ergun, O., and Freeman, M. (2021). The nutritious supply chain: Optimizing humanitarian food assistance. *INFORMS Journal on Optimization*, 3(2):200–226.

Pota, M., Scalco, E., Sanguineti, G., Farneti, A., Cattaneo, G. M., Rizzo, G., and Esposito, M. (2017). Early prediction of radiotherapy-induced parotid shrinkage and toxicity based on CT radiomics and fuzzy classification. *Artificial Intelligence in Medicine*, 81:41–53.

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., Bie, T. D., and Flach, P. (2020). FACE. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*.

Prat, E. and Chatzivasileiadis, S. (2020). Learning active constraints to efficiently solve bilevel problems. *arXiv preprint arXiv:2010.06344*.

Rawal, K., Kamar, E., and Lakkaraju, H. (2020). Algorithmic recourse in the wild: Understanding the impact of data and model shifts. *arXiv preprint arXiv:2012.11788*.

Rossi, L., Bijman, R., Schillemans, W., Aluwini, S., Cavedon, C., Witte, M., Incrocci, L., and Heijmen, B. (2018). Texture analysis of 3d dose distributions for predictive modelling of toxicity rates in radiotherapy. *Radiotherapy and Oncology*, 129(3):548–553.

Russell, C. (2019). Efficient search for diverse coherent explanations. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 20–28.

Sang, L., Xu, Y., and Sun, H. (2024). Encoding carbon emission flow in energy management: A compact constraint learning approach. *IEEE Transactions on Sustainable Energy*, 15(1):123–135.

Say, B., Devriendt, J., Nordström, J., and Stuckey, P. J. (2020). Theoretical and experimental results for planning with learned binarized neural network transition models. *International Conference on Principles and Practice of Constraint Programming*, pages 917–934.

Say, B. and Sanner, S. (2018). Planning in factored state and action spaces with learned binarized neural network transition models. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*, pages 4815–4821.

Say, B., Wu, G., Zhou, Y. Q., and Sanner, S. (2017). Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence IJCAI-17*, pages 750–756.

Schede, E. A., Kolb, S., and Teso, S. (2019). Learning linear programs from data. *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1019–1026.

Schweidtmann, A. M. and Mitsos, A. (2019). Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3):925–948.

Scott, J. G., Sedor, G., Ellsworth, P., Scarborough, J. A., Ahmed, K. A., Oliver, D. E., Eschrich, S. A., Kattan, M. W., and Torres-Roca, J. F. (2021a). Pan-cancer prediction of radiotherapy benefit using genomic-adjusted radiation dose (gard): A cohort-based pooled analysis. *The Lancet Oncology*, 22(9):1221–1229.

Scott, J. G., Sedor, G., Scarborough, J. A., Kattan, M. W., Peacock, J., Grass, G. D., Mellon, E. A., Thapa, R., Schell, M., Waller, A., et al. (2021b). Personalizing radiotherapy prescription dose using genomic markers of radiosensitivity and normal tissue toxicity in nsclc. *Journal of Thoracic Oncology*, 16(3):428–438.

Semenenko, V. A., Reitz, B., Day, E., Qi, X. S., Miften, M., and Li, X. A. (2008). Evaluation of a commercial biologically based IMRT treatment planning system. *Medical Physics*, 35(12):5851–5860.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. *International Conference on Machine Learning*, pages 3076–3085.

Shang, C., Huang, X., and You, F. (2017). Data-driven robust optimization based on kernel learning. *Computers & Chemical Engineering*, 106:464–479.

Sharma, S. (2017). Activation functions in neural networks. Available at: `shorturl.at/ACOP26`.

Skiena, S. S. (2008). *The Algorithm Design Manual*. Springer Publishing Company, Incorporated, 2nd edition.

Slack, D., Hilgard, A., Lakkaraju, H., and Singh, S. (2021). Counterfactual explanations can be manipulated. *Advances in Neural Information Processing Systems*, 34:62–75.

Spyros, C. (2020). From decision trees and neural networks to MILP: power system optimization considering dynamic stability constraints. *2020 European Control Conference (ECC)*, pages 594–594.

Sroka, D. and Pawlak, T. P. (2018). One-class constraint acquisition with local search. *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, pages 363–370.

Stefanicka-Wojtas, D. and Kurpas, D. (2023). Personalised medicine—implementation to the healthcare system in europe (focus group discussions). *Journal of Personalized Medicine*, 13(3):380.

Stinstra, E. and den Hertog, D. (2008). Robust optimization using computer experiments. *European Journal of Operational Research*, 191(3):816–837.

Stoer, J. and Botkin, N. D. (2005). Minimization of convex functions on the convex hull of a point set. *Mathematical Methods of Operations Research*, 62(2):167–185.

Stoer, J., Botkin, N. D., and Pykhteev, O. A. (2007). An interior-point method for minimizing convex functions on the convex hull of a point set. *Optimization*, 56(4):515–524.

Sun, S., Cao, Z., Zhu, H., and Zhao, J. (2019). A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681.

Tasci, E., Zhuge, Y., Camphausen, K., and Krauze, A. V. (2022). *Cancers*, 14(12):2897.

Thams, F., Halilbaši, L., Pinson, P., Chatzivasileiadis, S., and Eriksson, R. (2017). Data-driven security-constrained OPF. *Proc. 10th Bulk Power Syst. Dyn. Control Symp.*, pages 1–10.

Tomatis, S., Rancati, T., Fiorino, C., Vavassori, V., Fellin, G., Cagna, E., Mauro, F. A., Girelli, G., Monti, A., Baccolini, M., Naldi, G., Bianchi, C., Menegotti, L., Pasquino, M., Stasi, M., and Valdagni, R. (2012). Late rectal bleeding after 3d-CRT for prostate cancer: development of a neural-network-based predictive model. *Physics in Medicine and Biology*, 57(5):1399–1412.

Tsay, C., Kronqvist, J., Thebelt, A., and Misener, R. (2021). Partition-based formulations for mixed-integer optimization of trained relu neural networks. *arXiv preprint arXiv:2102.04373*.

Tsouros, D., Verhaeghe, H., Kadıoğlu, S., and Guns, T. (2023). Holy grail 2.0: From natural language to constraint models. *arXiv preprint arXiv:2308.01589*.

Tucker, S. L., Li, M., Xu, T., Gomez, D., Yuan, X., Yu, J., Liu, Z., Yin, M., Guan, X., Wang, L.-E., Wei, Q., Mohan, R., Vinogradskiy, Y., Martel, M., and Liao, Z. (2013). Incorporating single-nucleotide polymorphisms into the lyman model to improve prediction of radiation pneumonitis. *International Journal of Radiation Oncology*Biology*Physics*, 85(1):251–257.

UNHCR, UNICEF, WFP, and WHO (2002). Food and nutrition needs in emergencies. Available at: `https://www.who.int/nutrition/publications/emergencies/a83743/en/`.

UNWFP, U. (2021). The WFP food basket. Available at: `https://www.wfp.org/wfp-food-basket`.

Upadhyay, S., Joshi, S., and Lakkaraju, H. (2021). Towards robust and reliable algorithmic recourse. *Advances in Neural Information Processing Systems*, 34:16926–16937.

Ustun, B., Spangher, A., and Liu, Y. (2019). Actionable recourse in linear classification. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 10–19.

Vaclavik, R., Novak, A., Sucha, P., and Hanzlek, Z. (2018). Accelerating the branch-and-price algorithm using machine learning. *European Journal of Operational Research*, 271(3):1055–1069.

Valdes, G., Solberg, T. D., Heskel, M., Ungar, L., and Simone, C. B. (2016). Using machine learning to predict radiation pneumonitis in patients with stage i non-small cell lung cancer treated with stereotactic body radiation therapy. *Physics in Medicine and Biology*, 61(16):6105–6120.

van Dijk, L. V., Brouwer, C. L., van der Schaaf, A., Burgerhof, J. G., Beukinga, R. J., Langendijk, J. A., Sijtsema, N. M., and Steenbakkers, R. J. (2017). CT image biomarkers to improve patient-specific prediction of radiation-induced xerostomia and sticky saliva. *Radiotherapy and Oncology*, 122(2):185–191.

Venzke, A. and Chatzivasileiadis, S. (2020). Verification of neural network behaviour: Formal guarantees for power system applications. *IEEE Transactions on Smart Grid*, 12(1):383–397.

Venzke, A., Qu, G., Low, S., and Chatzivasileiadis, S. (2020a). Learning optimal power flow: Worst-case guarantees for neural networks. *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7.

Venzke, A., Viola, D. T., Mermet-Guyennet, J., Misyris, G. S., and Chatzivasileiadis, S. (2020b). Neural networks for encoding dynamic security-constrained optimal power flow to mixed-integer linear programs. *arXiv preprint arXiv:2003.07939*.

Verma, S., Dickerson, J., and Hines, K. (2020). Counterfactual explanations for machine learning: a review. *arXiv preprint arXiv:2010.10596*.

Verwer, S., Zhang, Y., and Ye, Q. C. (2017). Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence*, 244:368–395.

Villarrubia, G., De Paz, J. F., Chamoso, P., and De la Prieta, F. (2018). Artificial neural networks used in optimization problems. *Neurocomputing*, 272:10–16.

Virgolin, M. and Fracaros, S. (2023). On the robustness of sparse counterfactual explanations to adverse perturbations. *Artificial Intelligence*, 316:103840.

Wachter, S., Mittelstadt, B., and Russell, C. (2018). Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard Journal of Law and Technology*, 31(2):841–887.

Wang, K., Lozano, L., Cardonha, C., and Bergman, D. (2023). Optimizing over an ensemble of trained neural networks. *INFORMS Journal on Computing*, 35(3):652–674.

Wasserkrug, S., Boussioux, L., den Hertog, D., Mirzazadeh, F., Birbil, I., Kurtz, J., and Maragno, D. (2024). From large language models and optimization to decision optimization copilot: A research manifesto. *arXiv preprint arXiv:2402.16269*.

Witte, M. G., Van Der Geer, J., Schneider, C., Lebesque, J. V., Alber, M., and Van Herk, M. (2007). IMRT optimization including random and systematic geometric errors based on the expectation of TCP and NTCP. *Medical Physics*, 34(9):3544–3555.

Wolfe, P. (1961). A duality theorem for non-linear programming. *Quarterly of Applied Mathematics*, 19(3):239–244.

Wuyckens, S., Dasnoy, D., Janssens, G., Hamaide, V., Huet, M., Loÿen, E., de Hertaing, G. R., Macq, B., Sterpin, E., Lee, J. A., Souris, K., and Deffet, S. (2023). Opentps – open-source treatment planning system for research in proton therapy. *arXiv preprint arXiv:2303.00365*.

Xavier, Á. S., Qiu, F., and Ahmed, S. (2021). Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756.

Xu, H., Caramanis, C., and Mannor, S. (2008). Robust regression and lasso. *Advances in Neural Information Processing Systems*, 21.

Xu, S., Liu, H., Wang, X., and Jiang, X. (2014). A robust error-pursuing sequential sampling approach for global metamodeling based on voronoi diagram and cross validation. *Journal of Mechanical Design*, 136(7):071009.

Yang, D., Hendifar, A., Lenz, C., Togawa, K., Lenz, F., Lurje, G., Pohl, A., Winder, T., Ning, Y., Groshen, S., and Lenz, H.-J. (2011). Survival of metastatic gastric cancer: Significance of age, sex and race/ethnicity. *Journal of Gastrointestinal Oncology*, 2(2):77–84.

Yang, S. and Bequette, B. W. (2021). Optimization-based control using input convex neural networks. *Computers & Chemical Engineering*, 144:107143.

Yao, X., Crook, J., and Andreeva, G. (2017). Enhancing two-stage modelling methodology for loss given default with support vector machines. *European Journal of Operational Research*, 263(2):679–689.

Yu, W. and Lui, R. (2006). Dual methods for nonconvex spectrum optimization of multicarrier systems. *IEEE Transactions on Communications*, 54(7):1310–1322.

Zhang, H. H., D'Souza, W. D., Shi, L., and Meyer, R. R. (2009). Modeling plan-related clinical complications using machine learning tools in a multiplan IMRT framework. *International Journal of Radiation Oncology∗Biology∗Physics*, 74(5):1617–1626.

Zhang, S., Campos, J., Feldmann, C., Walz, D., Sandfort, F., Mathea, M., Tsay, C., and Misener, R. (2023). Optimizing over trained gnns via symmetry breaking. *Advances in Neural Information Processing Systems*, 36:44898–44924.

Zhou, Z., Cheng, S., and Hua, B. (2000). Supply chain optimization of continuous process industries with sustainability considerations. *Computers & Chemical Engineering*, 24(2):1151–1158.

Zhu, Y. and Burer, S. (2024). An extended validity domain for constraint learning. *arXiv preprint arXiv:2406.10065*.

Acknowledgments

I am deeply grateful to my supervisors for guiding me through this PhD adventure. To Dick, for many inspiring meetings and for your guidance and belief in me. You have shown me the true values and virtues of being a dedicated researcher, for which I am profoundly grateful. To Ilker, your attentive listening, encouragement, and insightful contributions have enriched every idea I pursued. Thank you for nurturing my strengths, helping me overcome my weaknesses, and for always having my back.

I want to thank my co-authors and mentors Dimitris and Ali. You made my time as a visiting researcher in Boston an incredibly enriching experience. I felt welcomed and encouraged to explore new areas of research, broadening both my professional and personal horizons.

I also want to thank my co-authors and everyone who contributed to this dissertation. To Ade and Holly, thank you for your invaluable support in writing my first paper and for collaborating on the development of the optimization with constraint learning framework. This work remains one of the achievements I am most proud of. To Tabea, Rob, and Jannis, I am grateful for the opportunity to work with you on a project that has grown to be especially meaningful to me - explainable AI. Your insights, teamwork, and enthusiasm made this collaboration both impactful and fun. To Alex, for teaching me to always aim high and to do my best to achieve it. Working with you has been truly inspiring and successful. To Thomas, Segev, Gregory, Léonard, and Farzaneh, thank you for sharing your knowledge and passion for research with me. I have learned so much from each of you. To Mayukh and Roshan, thank you for being more than just co-authors - you are true friends. I cherish the countless memories we made in the office: brainstorming, laughing, eating, sleeping during a storm, coding, doing a workout, and supporting each other through it all. These moments mean so much to me, and I will treasure them always.

I want to express my gratitude to everyone who walked (part of) this PhD journey with me. To all my colleagues, thank you for your advice, feedback, insightful questions, and above all, your kind and motivating words. You made this journey both stimulating and enjoyable. To every PhD student I had the pleasure of meeting and spending time with – thank you for the daily coffee and lunch chats, the exciting birthday celebrations, the delicious dinner parties, and your unwavering support.

To Shuai, for sharing not just the office but so much more. Your smile and positive attitude were always the perfect way to start the day. To my "above average" friend Danique, thank you for the incredible memories from Boston – playing table tennis, en-

joying a beer at Lamplighter, and exploring the streets of NYC. Your friendship added joy and balance to this adventure.

Ai miei cari amici Angelo, Ciccio, e Fabio, grazie per avermi dato fiducia e supporto morale anche nei momenti più complessi. Il vostro affetto per me è stato una costante fonte di forza e ispirazione, e sono profondamente grato di avervi nella mia vita.

Un grazie speciale alla mia famiglia. Mamma, papà, Rossella, Francesco, Sara, Matilde, Alice, il vostro amore, il vostro supporto incondizionato, e la vostra fiducia nelle mie capacità sono stati il pilastro di questo percorso. Se ho raggiunto questo traguardo, è anche merito vostro. Grazie di cuore.

To Elisabeth, for revealing to me a new side of the world - the most beautiful one. You are my muse, my source of inspiration, and my greatest support. Your love and belief in me have been invaluable. To Amelia, for teaching me to see life through a different lens, for reminding me of what truly matters, and for grounding me in the moments that count.

To the ABS, for welcoming me with open arms and providing an environment that challenged me to grow academically, professionally, and personally. ABS, you rewarded me not only with a PhD thesis that I am immensely proud of but also with the greatest gift of all - a beautiful wife who has become my partner in every sense of the word.

Thank you!

Donato Maragno
Amsterdam, December 2024

Summary
Optimization with Constraint Learning

Mathematical optimization is a decision-making tool widely used in fields such as engineering, logistics, finance, and healthcare. It helps solve complex problems by efficiently allocating resources, time, and effort, thereby improving decision-making and problem-solving processes. A sub-field of mathematical optimization is mixed-integer optimization (MIO). MIO allows practitioners to model problems using linear functions with both continuous and integer variables, enabling the solution of large-scale problems through efficient algorithms and advancements in computational power. The success of MIO and mathematical optimization depends on the accuracy of the model's representation of reality, which can be challenging when dealing with unknown constraints or objective functions. This thesis presents a framework for learning unknown constraints and objective functions via machine learning (ML). It involves training MIO-representable ML models on features that also serve as decision variables, embedding these models into the optimization framework to approximate unknown functions.

For example, consider the World Food Programme's objective to optimize the food supply chain for a specific population in need. This problem can be formulated as an MIO model aimed at minimizing total costs, with constraints on material flow, fleet capacity, and supply availability. However, the model initially lacks constraints to ensure the "palatability" of the food—ensuring that recipients enjoy the food and know how to prepare it. While there is no explicit function to capture people's taste preferences, data can be collected through surveys and ML models can be trained to learn the function that is then embedded as a constraint of the optimization model. This process of learning constraints and objective functions from data is known as "constraint learning."

This thesis introduces a comprehensive Optimization with Constraint Learning (OCL) framework, focusing on MIO representable ML models to ensure computational efficiency and global optimality - critical aspects for tackling large-scale problems. The proposed framework facilitates adoption by practitioners and researchers, offering significant potential for further research to refine and expand OCL applications. The versatility of OCL is demonstrated through two distinct applications: radiotherapy (RT) optimization and explainable artificial intelligence.

In the first application, OCL is employed to personalize cancer treatment planning, offering an alternative to traditional population-based approaches by addressing radiation-

induced toxicity (RIT) based on individual patient characteristics. The risk of RIT is modeled using a predictive model trained on data from patients who received varying radiation doses and have different biomarkers and tumor specifics. This predictive model is then integrated into the mathematical optimization model as a constraint, limiting the probability of adverse side effects.

In the second application, the OCL framework is used to generate actionable counterfactual explanations (CEs) given a trained predictive model and a factual instance. The thesis explores how OCL helps to generate CEs that adhere to established criteria for *good* CEs such as validity, sparsity, proximity, and diversity. Additionally, two novel modeling approaches are proposed to address data manifold closeness and diversity, which are relevant in practical applications. The use of OCL for generating CEs is further extended to ensure recourse robustness, ensuring that the CE remains valid even with small perturbations—a fundamental consideration when the user does not have full control over the outcome, and environmental factors may influence it.

Samenvatting
Optimization with Constraint Learning

Wiskundige optimalisatie is een besluitvormingsinstrument dat veel wordt gebruikt in gebieden zoals techniek, logistiek, financiën en gezondheidszorg. Het helpt bij het oplossen van complexe problemen door middelen, tijd en inspanningen efficiënt toe te wijzen, waardoor besluitvormings- en probleemoplossingsprocessen worden verbeterd. Een deelgebied van de wiskundige optimalisatie is *mixed-integer optimalisatie* (*MIO*). MIO stelt professionals in staat om problemen te modelleren met behulp van lineaire functies met zowel continue als geheeltallige variabelen, waardoor grootschalige problemen kunnen worden opgelost door efficiënte algoritmen. Het succes van MIO en wiskundige optimalisatie hangt af van de nauwkeurigheid van de weergave van de werkelijkheid in het model, wat een uitdaging kan zijn als je te maken hebt met onbekende randvoorwaarden of doelfuncties. Dit proefschrift presenteert een raamwerk om onbekende randvoorwaarden en doelfuncties te leren via *machine learning* (*ML*). Het behelst het trainen van MIO-representeerbare ML-modellen op kenmerken die ook dienen als beslissingsvariabelen, en het integreren van deze modellen in het optimalisatieraamwerk om onbekende functies te benaderen.

Neem bijvoorbeeld het doel van het Wereldvoedselprogramma om de voedselvoorzieningsketen voor een specifieke bevolking in nood te optimaliseren. Dit probleem kan worden geformuleerd als een MIO-model dat gericht is op het minimaliseren van de totale kosten, met randvoorwaarden voor de materiaalstromen, transportcapaciteit en beschikbaarheid van voorraden. Het model mist echter in eerste instantie restricties om de "smakelijkheid" van het voedsel te waarborgen—ervan uitgaande dat de ontvangers weten hoe ze het moeten bereiden. Hoewel er geen expliciete functie is om de smaakvoorkeuren van mensen vast te leggen, kunnen gegevens worden verzameld via enquêtes en kunnen ML-modellen worden getraind om de functie te leren die vervolgens wordt ingebed als een randvoorwaarde in het optimalisatiemodel. Dit proces van het leren van randvoorwaarden en doelfuncties uit gegevens staat bekend als "constraint learning".

Dit proefschrift introduceert een uitgebreid Optimization with Constraint Learning (OCL)-raamwerk, waarbij de nadruk ligt op MIO-representeerbare ML-modellen om rekenefficiëntie en globale optimaliteit te garanderen - kritieke aspecten voor het aanpakken van grootschalige problemen. Het voorgestelde raamwerk maakt toepassing door professionals en onderzoekers mogelijk en biedt aanzienlijke potentie voor verder onderzoek om OCL-toepassingen te verfijnen en uit te breiden. De veelzijdig-

heid van OCL wordt aangetoond met behulp van twee verschillende toepassingen: radiotherapie-optimalisatie (RT) en uitlegbare kunstmatige intelligentie.

In de eerste toepassing wordt OCL gebruikt om de behandeling van kanker persoonlijker te maken, waarbij een alternatief wordt geboden voor traditionele populatiegebaseerde benaderingen door stralingsgeïnduceerde toxiciteit (RIT) op basis van individuele patiëntkenmerken te analyseren. Het risico op RIT wordt gemodelleerd met behulp van een voorspellend model dat is getraind op gegevens van patiënten die verschillende stralingsdoses hebben ontvangen en verschillende biomarkers en tumorspecifieke kenmerken hebben. Dit voorspellende model wordt vervolgens geïntegreerd in het wiskundige optimalisatiemodel als een randvoorwaarde die de kans op bijwerkingen beperkt.

In de tweede toepassing wordt het OCL-raamwerk gebruikt om om Couterfactual Explanations (CEs) te genereren gegeven een getraind voorspellend model en een feitelijke instantie. Het proefschrift onderzoekt hoe OCL helpt bij het genereren van CE's die voldoen aan de vastgestelde criteria voor goede CE's zoals geldigheid, spreiding, nabijheid en diversiteit. Bovendien worden twee nieuwe modelleeraanpakken voorgesteld voor de praktijk relevante aspecten van nabijheid en diversiteit aan te pakken. Het gebruik van OCL voor het genereren van CE's wordt verder uitgebreid om *recourse robuustheid* te garanderen, zodat de CE zelfs bij kleine verstoringen geldig blijft—een fundamentele overweging wanneer de gebruiker geen volledige controle heeft over de uitkomst en omgevingsfactoren deze kunnen beïnvloeden.