January 2021

Integrated Optimization And Learning Methods Of Predictive And Prescriptive Analytics

Mehmet Kolcu
Wayne State University

INTEGRATED OPTIMIZATION AND LEARNING METHODS OF PREDICTIVE AND PRESCRIPTIVE ANALYTICS

by

MEHMET KOLCU

DISSERTATION

Submitted to the Graduate School,

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2021

MAJOR: INDUSTRIAL ENGINEERING

Approved By:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

to my Lovely Wife, my Mother, and my Sisters

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerest gratitude to my advisor, Dr. Alper Murat, for his constant guidance, continuous support, motivation, and immense knowledge throughout the course of my doctoral studies. His ingenious ideas, perfectionism, and enthusiasm have not only directly contributed to me academically for the work presented in this dissertation but also inspired me to work harder, aim higher, and helped me grow immeasurably personally.

I would also like to express my sincere appreciation towards the committee members, Dr. Ratna Babu Chinnam, Dr. Saravanan Venkatachalam, Dr. Yanchao Liu, and Dr. Suzan Arslanturk. Their invaluable inputs significantly improved the preparation for this dissertation and helped me a lot to embark on my new career. I would like to thank especially Dr. Chinnam and Dr. Venkatachalam whose offered graduate courses served as a great tool and shaped my ideas in this dissertation, and Dr. Liu, who is extremely curious about my work and for his thoughtful words of encouragement. Moreover, I would like to thank Dr. Murat Yildirim for his unselfish, unfailing support, expert opinion, and encouragement. I am also extremely grateful to Dr. Arslanturk for her guidance throughout my graduate studies and her generosity to support my work financially.

I am thankful to my friends: Suleyman, Melike, Deniz, and Nur Banu for their continuous support, valuable comments, advice, and fun times we spent together, and also thankful to Mahmut Tutam, who is my first companion in graduate studies.

In addition, I also wish to thank Dr. Jayant Trewn, who contributes my teaching skills and backs me up in the courses we serve together, and who became a more friend to me rather than a teaching supervisor.

My sincere thanks also go to the Industrial and Systems Engineering department and Engineering Dean's Office, which are generously funded my doctoral studies at Wayne State University, the scholarship granted by The Ministry of National Education of the Republic of Turkey, and my friends Murat Sahin, Naim Sari, and Ibrahim Bacaksiz, who co-signed with me to get this scholarship.

Finally, I would like to express gratitude to my family for their unconditional support during my life and to my lovely wife, Elif, who is the pillar of my life, made me a better person, and unconditionally loves me during my good and bad times. I also wish to express my appreciation to many people who have contributed to my whole studies and education. I will not be able to conclude this section without thanking, especially Dr. Metin Dagdeviren, Dr. Izzettin Temiz, Dr. Murat Arikan, Dr. Cevriye Gencer, who are the motivational source of my academic life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Background

The amount of collected data is rapidly increasing, and the revolution of data collection has created different kinds of data analytics. Modern data processing platforms can load collected big data sets and prepare them for analysis within seconds these days. In the meantime, data-driven businesses analyze more data and dive deeper into analytics than ever before. Descriptive analytics, predictive analytics, and prescriptive analytics are the three dominant types of analytics, transforming the data exploration and examination into insights to make better and smarter decisions. Each of these analytics process the data to answer different questions.

Descriptive analytics: Descriptive analytics is the most basic and commonly used form of data analytics that looks at data statistically to answer "What has happened?". This analytics aims to learn from the past by aggregating and mining the data and using simple maths and statistical tools.

Predictive analytics: Predictive analytics focuses on predicting and understanding "What could happen?". It analyzes the past data patterns and trends by leveraging forecasting algorithms, statistical tools, and machine learning models. Then, it attempts to forecast the best possible future outcomes and their likelihood.

Prescriptive analytics: Prescriptive analytics recommends the best possible courses of action based on the best possible future scenarios. To achieve the best outcomes, prescriptive analytics uses simulation and optimization algorithms to answer "What should be done?".

In summary, prescriptive analytics is the next step of predictive analytics that seeks solutions to transform data into a good decision rather than a good prediction. We can talk about a good prediction as long as it provides good decisions. However, all the predictive methods measure the quality of predictions based on the degree of closeness between actual and predicted responses, such as mean squared error, maximum likelihood, or cross-entropy. The one point overlooked is the quality of these predictions must be measured based on the decisions that these predictions lead us in a business environment.

Although there has been increasing research interest in combining predictive and prescriptive analytics to improve the quality of decisions, it is still an open challenge to address this issue and leverage the available data to better future actions. To this end, this dissertation focuses on developing new frameworks using machine learning and mathematical optimization techniques under different circumstances. The common point of all the developed frameworks is that they evaluate the prediction performance based on decision quality.

1.2 Motivation

Despite all the machine learning area developments, traditional stochastic and robust optimization techniques are still being employed to overcome uncertainty in the parameters of decision models. Point estimate-based solutions like "first predict, then optimize" do not provide a good decision since the effect of these predictions

are not considered in the prescription stage. In today's data-rich world, learning algorithms' performance is measured based on the closeness between predicted and actual responses. It is a need to evaluate the performance of learning algorithms based on the decision qualities and integrate predictive and prescriptive analytics frameworks to improve the accuracy of the predictions most relevant for the optimal decisions.

1.3 Research Objectives

In this dissertation, the overall goal is to develop integrated frameworks for performing predictive and prescriptive analytics concurrently to realize the best prescriptive performance under uncertainty; simultaneously, these frameworks should be applicable to all prescriptive tasks involving uncertainty. These frameworks should be scalable and capable of handling integrated predictive and prescriptive tasks with a reasonable computational effort and enabling users to apply decomposition algorithms for large-scale problems. Further, these frameworks should be able to accommodate prediction tasks ranging from simple regression to more complex black-box neural network models. Accordingly, this dissertation has the following research objectives:

1. Developing an integrated analytical framework for regression-based prediction and mathematical programming-based prescription tasks as a bilevel program. While the lower-level problem prescribes decisions based on the predicted out-

come for a specific observation, the upper-level evaluates the quality of the decision regarding true parameters. The upper-level problem can be considered as a prescriptive error, and the goal is to minimize this prescriptive error.

(a) Offering different approaches to control the "the prescription generalization error" associated with out-of-sample observation in order to achieve the same performance in external data (test) compared to internal data (train).

(b) For large dimensional problems, developing a scalable method to solve the resulting bilevel formulation and solving the generated sub-problems in parallel-working slave nodes, then sending the results to the master node for evaluation and preparation for the next iteration if the criteria are not satisfied.

(c) Comparing the results with traditional methods such as stochastic optimization, point-estimate-based optimization, and recently developed competing methods from the literature in order to show the performance of integrated methodology.

2. Developing an integrated learning framework for neural network-based prediction and optimization tasks as a nested neural network. While the predictive neural network promotes decisions based on predicted outcomes, the prescriptive neural network evaluates the quality of predicted decisions with respect to true values. Extending this integrated learning framework for fully or partially

known prescription tasks.

(a) For large dimensional problems, developing a scalable method to train the nested neural network, training each prescriptive scenario with respect to the predictive task, then updating predictive weights by aggregating the scenario-based derivatives until convergence or a maximum number of iteration is reached.

(b) Proposing a weight initialization process for nested neural networks to improve convergence by starting a zero-constrained violation in the prescription task or starting with a suitable prescription representation.

(c) Demonstrating the effectiveness of proposed nested neural network architectures and comparing results with recently developed methodology.

1.4 Dissertation Organization

Now that we address the problem in predictive and prescriptive areas, the rest of this dissertation is framed as follow. In Chapter 2, We build an integrated optimization framework for both predictive and prescriptive tasks in the light of our objectives where the prescriptive model is fully known (objective function and constraints). We show the power of integrated optimization with two examples, and we offer ways of controlling the generalization error for the sake of consistent test results. We also introduce a decomposition algorithm to solve the resulting integrated optimization framework. In Chapter 3, we build different integrated learning frame-

works for the predictive and prescriptive tasks where the prescriptive task is known fully or partially to decision-makers. We also propose a weight initialization process and decomposition technique for large-scale problems. In Chapter 4, we discuss what has been achieved so far based on our motivation and objectives. We offer some concluding remarks for this dissertation and define future work extensions.

1.5 Notations

There are different variables and parameters in prediction and prescription tasks. In real-world examples, prediction tasks mainly have their own components, feature data, response data, and the parameters connecting these two via a function. As for prescription tasks, we generally have cost vector parameters in the objective function,right-hand and left-hand side parameters in the constraints, and decision variables. Our interest in this dissertation is to fit a function with responses and feature data and feed prescriptive model parameters by this fitted function. For the sake of uniformity, we use the same notation for the rest of this dissertation. We indicate feature data as X, responses as Y, the parameter of the predictive algorithm as β, and decision variables as Z. Responses indicated by Y serve as a bridge connecting predictive and prescriptive tasks.

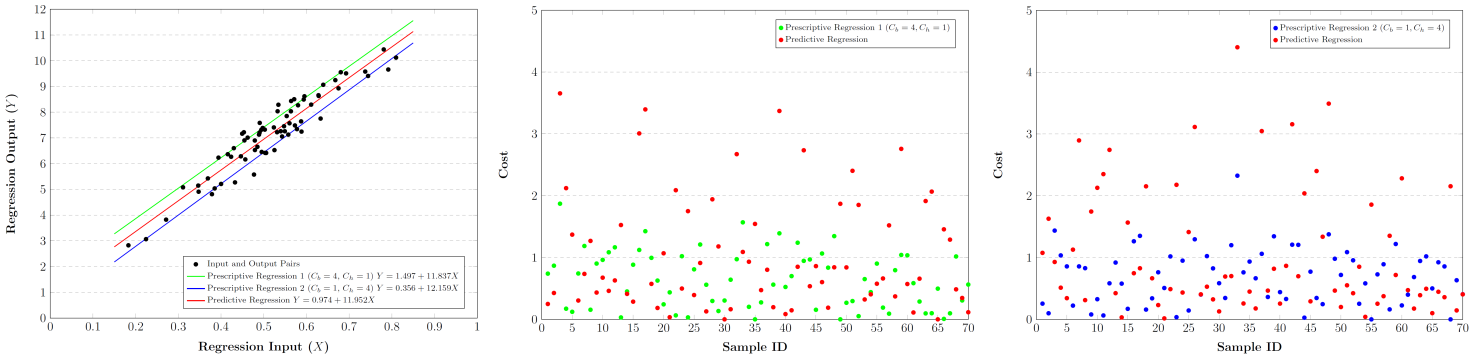
CHAPTER 2 INTEGRATED OPTIMIZATION OF PREDICTIVE AND PRESCRIPTIVE TASKS

2.1 Introduction

Today we are living in a world called the information age. The exponential growth of data availability, ease of accessibility in computational power, and more efficient optimization techniques have paved the way for massive developments in the field of predictive analytic. Particularly when organizations have realized the benefits of predictive methods in improving their efficiency and gaining an advantage over their competitors, these predictive techniques become more powerful. There is a mutual relationship between predictive and prescriptive analytics. We cannot deny the role of optimization techniques while obtaining predictive models because most of the predictive models are trained over the minimization of a loss or maximization of a gain function. On the other hand, prescriptive models containing uncertainty need the estimated inputs from predictive analytics to handle uncertainty in optimization parameters. Although these statistical (machine) learning methods have been provided well-aimed predictions for uncertain parameters in many different fields of science, scenario-based stochastic optimization introduced by Dantzig [9] and similar works in [8, 19, 15, 20] are widely preferred techniques in order to tackle uncertainty in decision problems. One of the reasons why these stochastic or robust optimization methods or similar solutions provided by Ben-Tal et al. [4] and Bertsimas et al. [6] do perform better than point estimate-based optimization is that training process of statistical learning methods does not take into account optimal

actions because traditional learning algorithms measure prediction quality based on the degree of closeness between true and predicted values. This gap between prescriptive and predictive analytics leads point estimate-based decisions to a failure in the prescription phase.

In order to visualize our motivation, imagine a newsvendor problem with a cost function, $Cost = C_h(Z - Y)^+ + C_b(Y - Z)^+$, containing holding cost (C_h) and backordering cost (C_b) with demand parameter (Y), and let's assume the historical data $D = \{(\tilde{X}^n, \tilde{Y}^n)\}_{n \in N} = (\tilde{X}, \tilde{Y})$ for the demand Y and explanatory features X is given. In the company of feature data, a predictive regression model can be built for future demand as $\hat{Y} = \psi(\tilde{X}, \hat{\beta})$ where $\hat{\beta} = \underset{\beta}{\operatorname{argmin}}(\tilde{Y} - \psi(\tilde{X}, \beta))^T(\tilde{Y} - \psi(\tilde{X}, \beta))$ in order to minimize cost function. However, the training criteria of this predictive regression model will be based on the closeness between true and predicted responses via a loss function which is mean squared error in our case, the predictive regression model will not capture the effect of holding and backordering costs in the cost function, and future predictions will most probably fail in the prescription stage. In Figure 1a, one predictive regression and two different prescriptive regression models (considering holding and backordering costs) are presented based on two different scenario. When backordering cost is greater than holding cost, using prescriptive regression 1 gives a lower cost on the average since it keeps predictions higher in order to avoid shortage cost as seen in Figure 1b. Similarly, when holding cost is greater than backordering cost, using prescriptive regression 2 gives a lower cost on the average since it keeps predictions lower in order to avoid holding cost as seen in Figure 1c. The



(a) Regression Models **(b)** Predictive Reg. vs Prescriptive Reg 1 **(c)** Predictive Reg. vs Prescriptive Reg 2

Figure 1: Comparison of Predictive and Prescriptive Regression Models Based on Cost Function

question is how to obtain such a predictive model directly caring the characteristics of the prescriptive model.

In this chapter, our purpose is to build a framework for a predictive regression model caring for the characteristics of the prescriptive model and providing the best decisions. We will call our framework as "Integrated Predictive and Prescriptive Optimization" (IPPO). While IPPO is seeking the most accurate predictive regression model, the desired predictive regression model will tackle uncertainty by providing the best actions in the prescription stage.

2.2 Related Work

There have been recent developments in supervised machine learning algorithms (classification and regression tasks), but no matter what kind of learning algorithm is employed, the common and final task is generally to train these algorithms in order

to find the closest predictions to true values.

The latest works in the literature show that scientists realize that training machine learning models solely based on a prediction criterion like mean squared error, mean absolute error, or log-likelihood loss is not enough when these noisy predictions will be parameters of prescription tasks. Bengio [5] is one of the first works considering this issue and building an integrated framework for both prediction and prescription tasks. He emphasized the importance of the evaluation criteria of predictive tasks. His neural network model was not designed to minimize prediction error but instead to maximize a financial task where those noisy predictions are used as input.

Another integrated framework developed by Kao et al.[14] trains parameters of a regression model based on an unconstrained optimization problem with a quadratic cost function. It is a hybrid algorithm between ordinary least square and empirical optimization. Tulabandhula and Rudin [23] minimizes a weighted combination of prediction error and operational cost, but they ignore that the operational cost must be assessed based on the true parameters instead of the predicted parameters. Bertsimas and Kallus [7] add a new dimension to this field by introducing the conditional stochastic optimization term. This simple but efficient idea leverages non-parametric machine learning methods in order to assign weights into train data points for a given test data point, then calculates optimal decisions over stochastic optimization based on calculated weights. However, this methodology is using machine learning

tools outside of the prescription problem. A different approach developed by Ban and Rudin [2] considers the decision variables as a function of auxiliary (feature) data, but this method may fail if the connection between optimal decisions and uncertain parameters is weak in a constrained optimization problem or may end up with indefeasible decisions. Oroojlooyjadid et al.[17] and Zhang and Gao [25] built an extension of the work provided by Ban and Rudin [2] since both of them approach to the solution from the same perspective, but they solve the problem via neural network to capture the non-linearity between auxiliary (feature) data and optimal decisions.

One more neural network-based integrated task is proposed by Donti et al. [11], and primarily they focus on quadratic stochastic optimization problems since they are tuning neural network parameters by differentiating the optimization solution to a stochastic programming problem.

One of the latest works in integrating predictive and prescriptive tasks is developed by Elmachtoub and Grigas [12] by introducing a new loss function called as SPO+. This framework aims to find the parameter of linear regression model inside of decision problem via SPO+ loss function. This method aims to minimize the difference between objective value provided by true parameters and objective value where decisions provided by predicted parameters are assessed.

2.3 Integrated Predictive and Prescriptive Optimization

Given a decision problem (DP) with parameter uncertainty, our goal is to estimate the predictive relationship between responses (uncertain parameters of the decision problem) and a set of input features such that the prescriptive modeling of the decision problem using the predicted responses results in the best decisions. We consider decision problems that can be formulated as an optimization model. Further, the statistical relationship between uncertain parameters (Y) and input features (X) can be approximated through a parametric regression model, i.e., $Y = \psi(X, \beta) + \epsilon$, where β is a vector of k parameters and ϵ is an error term and $\psi(\cdot)$ is some function describing the relationship between Y and X. Without loss of generality, given the estimates $\hat{Y} = (\hat{Y}_o, \hat{Y}_c)$ of uncertain parameters $Y = (Y_o, Y_c)$, we define the deterministic decision problem (DP) as follows:

$$\min_{Z} \; f(Z; \hat{Y}_o) \tag{2.1a}$$

$$\text{s.t. } h(Z; \hat{Y}_c) = 0 \tag{2.1b}$$

In the above formulation, Z denotes the decision variables, and $\hat{Y} = (\hat{Y}_o, \hat{Y}_c)$ denotes the estimates of the uncertain parameters in the objective and constraints set, respectively. Let $(\hat{Z})$ denotes the optimal solution of DP given $\hat{Y} = (\hat{Y}_o, \hat{Y}_c)$, i.e. $\hat{Z} = \underset{Z}{\operatorname{argmin}}\{f(Z; \hat{Y}_o)$ satisfying constraint 2.1b$\}$. The uncertain parameters in DP are estimated through a parametric regression model. Let's assume given the historical data $D = \{(\tilde{X}^n, \tilde{Y}^n)\}_{n \in N} = (\tilde{X}, \tilde{Y})$ for the response Y and explanatory features X, the

prediction problem (PP) using parametric regression is expressed as:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}(\tilde{Y} - \psi(\tilde{X}, \beta))^T(\tilde{Y} - \psi(\tilde{X}, \beta)) \tag{2.2}$$

In classical approach, the predictive (PP) and prescriptive (DP) tasks which are often treated independently and often in a sequence, i.e., first predict $\hat{Y} = \psi(\tilde{X}, \hat{\beta})$ (using PP) and then prescribe (using DP). This process is illustrated in Figure 2.

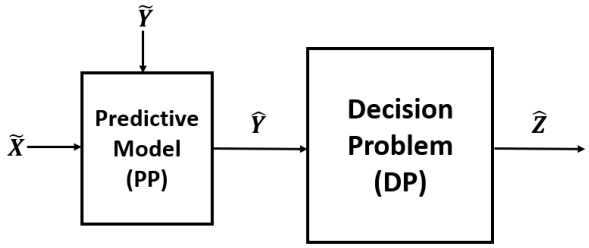


Figure 2: Independent Framework

We herein develop an integrated framework joining the predictive (PP) and prescriptive (DP). Three modules of the integration framework are as follows:

1. Given a set of independent features, a predictive regression model generating responses which are input to the optimization model as part of the input parameter set (Module 1),

2. An optimization model prescribing decisions based on input parameters (Module 2),

3. Another optimization model evaluating the quality of prescribed decisions with

respect to ground truth in the response space and updating the parameters of the predictive model (Module 3).

Figure 3 illustrates these three modules. The sequential predictive and prescriptive tasks (modules 1 and 2) are concurrently optimized through the module 3. While module 1 is a prediction model, modules 2 and 3 are decision optimization problems with their respective decisions influencing one another. The embedding structure of modules 1 and 2 within module 3 is similar to those of bilevel optimization problems. Hence, we model the integration framework as a nested optimization model. In the next section, we model the integrated prediction and prescription problem as a bilevel optimization model.

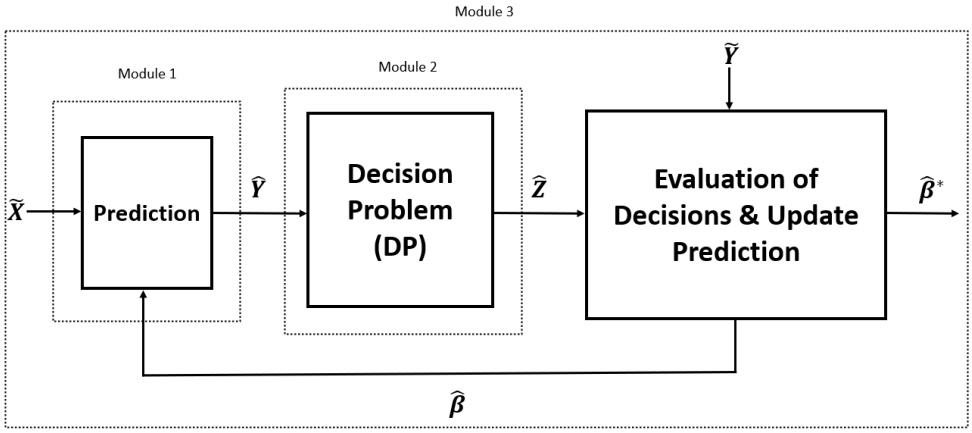


Figure 3: Integrated Framework

2.3.1 Bilevel Models and Solving Techniques

Bilevel problems are nested optimization problems where an upper-level optimization problem is constrained by a lower-level optimization problem [3]. A common application of the bilevel problems is a static leader-follower game in economics

[21], where the upper-level decision-maker (leader) has complete knowledge of the lower-level problem (follower). Decision variables of the upper-level serve as parameters of the lower-level.

We model the integrated framework for the prediction and prescription tasks as a bilevel optimization problem. The upper-level problem jointly determines the parameters of the regression model (β) and prescription decisions. At the lower-level, we make decisions with the help of predicted parameters, and we evaluate these decisions with respect to true parameters by fixing those prediction based "here-and-now" decisions ($\hat{Z}_f$) at the upper-level problem in constraint 2.3c, so that we integrate all proposed steps in one framework formulated as in (2.3a)-(2.3e).

$$\min_{Z_f, Z_s, \beta} \; F(Z_f, Z_s) \tag{2.3a}$$

$$\text{s.t.} \; G_i(Z_f, Z_s; \tilde{Y}) \leq 0 \qquad \forall i \in I \tag{2.3b}$$

$$Z_f = \hat{Z}_f \tag{2.3c}$$

$$\min_{\hat{Z}_f, \hat{Z}_s} \; F(\hat{Z}_f, \hat{Z}_s) \tag{2.3d}$$

$$\text{s.t.} \; G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y} = \psi(\tilde{X}, \beta)) \leq 0 \qquad \forall i \in I \tag{2.3e}$$

The most popular solution technique for bilevel optimization problems is to transform the bilevel problem into a single-level problem by replacing the objective function of the lower-level problem with Karush–Kuhn–Tucker (KKT) conditions [3]. The KKT conditions appear as dual and complementary slackness constraints. Duality con-

straint appears in 2.4e where ∇ represents the derivative of the Lagrangian function (Γ) with respect to the lower-level problem decisions variables. Because of the complementary slackness constraint, KKT conditions require convexity, so this approach is limited to convex lower-level problems. Complementary slackness constraint in 2.4f converts model into a non-linear problem; thus, these constraints are replaced with logic constraints by defining new binary variables and sufficiently enough an M parameter. With this final touch, the bilevel model turns into a mixed-integer problem, and traditional solvers can solve it as in form (2.4a)-(2.4g).

$$\min_{\substack{Z_f, Z_s, \beta \\ \hat{Z}_f, \hat{Z}_s, \pi_i}} \quad F(Z_f, Z_s) \tag{2.4a}$$

$$\text{s.t.} \quad G_i(Z_f, Z_s; \tilde{Y}) \leq 0 \qquad \forall i \in I \tag{2.4b}$$

$$Z_f = \hat{Z}_f \tag{2.4c}$$

$$G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y}) \leq 0 \qquad \forall i \in I \tag{2.4d}$$

$$\nabla_{\hat{Z}_f, \hat{Z}_s} \Gamma(\hat{Z}_f, \hat{Z}_s, \pi_i) = 0 \tag{2.4e}$$

$$G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y})\pi_i = 0 \qquad \forall i \in I \tag{2.4f}$$

$$\pi_i \geq 0 \qquad \forall i \in I \tag{2.4g}$$

where

$$\Gamma(\hat{Z}_f, \hat{Z}_s, \pi_i) = F(\hat{Z}_f, \hat{Z}_s) + \sum_{i \in I} \pi_i G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y})$$

$$\hat{Y} = \psi(\tilde{X}, \beta)$$

2.3.2 Controlling Generalization Error in IPPO

Our model is developed based on finding the best decision in the train data set. In order to ensure the quality of the prediction model in the external data set, we propose three different ways to control generalization error within this framework by regularizing predictive model parameters. The first method is to rewrite objective function as weighted average of predictive error and prescriptive error as shown in formulation (2.5a)-(2.5e) where $0 \leq \lambda_1 \leq 1$. When $\lambda_1 = 1$, we solve a pure bilevel optimization model without generalization error. When $\lambda_1 = 0$, we ignore the prescription part and optimize directly predictive algorithm solely, and that leads us to point estimate-based prescriptions.

$$\min_{Z_f, Z_s, \beta} \ \lambda_1 F(Z_f, Z_s) + (1 - \lambda_1) L(\tilde{Y}, \hat{Y} = \psi(\tilde{X}, \beta)) \tag{2.5a}$$

$$\text{s.t.} \ \ G_i(Z_f, Z_s; \tilde{Y}) \leq 0 \qquad \forall i \in I \tag{2.5b}$$

$$Z_f = \hat{Z}_f \tag{2.5c}$$

$$\min_{\hat{Z}_f, \hat{Z}_s} \ \ F(\hat{Z}_f, \hat{Z}_s) \tag{2.5d}$$

$$\text{s.t.} \ \ G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y} = \psi(\tilde{X}, \beta)) \leq 0 \qquad \forall i \in I \tag{2.5e}$$

The second method uses predictive error term again, but in a way that a constraint can restrict it. However, this restriction cannot be less than the loss value which is provided by the predictive model solely because constraining loss less than the optimal (L^*) makes the optimization model indefeasible. This method is shown in

formulation (2.6a)-(2.6f) with a restriction parameter $\lambda_2 \geq 1$.

$$\min_{Z_f, Z_s, \beta} \; F(Z_f, Z_s) \tag{2.6a}$$

$$\text{s.t. } G_i(Z_f, Z_s; \tilde{Y}) \leq 0 \qquad \forall i \in I \tag{2.6b}$$

$$L(\tilde{Y}, \hat{Y} = \psi(\tilde{X}, \beta)) \leq \lambda_2 L^* \tag{2.6c}$$

$$Z_f = \hat{Z}_f \tag{2.6d}$$

$$\min_{\hat{Z}_f, \hat{Z}_s} \; F(\hat{Z}_f, \hat{Z}_s) \tag{2.6e}$$

$$\text{s.t. } \; G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y} = \psi(\tilde{X}, \beta)) \leq 0 \qquad \forall i \in I \tag{2.6f}$$

The last method is to shrink predictive model parameters by penalizing with a penalty coefficient $\lambda_3 \geq 0$ as Tibshirani [22] introduced, this model is formulated in (2.7a)-(2.7e).

$$\min_{Z_f, Z_s, \beta} \; F(Z_f, Z_s) + \lambda_3 \beta^T \beta \tag{2.7a}$$

$$\text{s.t. } G_i(Z_f, Z_s; \tilde{Y}) \leq 0 \qquad \forall i \in I \tag{2.7b}$$

$$Z_f = \hat{Z}_f \tag{2.7c}$$

$$\min_{\hat{Z}_f, \hat{Z}_s} \; F(\hat{Z}_f, \hat{Z}_s) \tag{2.7d}$$

$$\text{s.t. } \; G_i(\hat{Z}_f, \hat{Z}_s; \hat{Y} = \psi(\tilde{X}, \beta)) \leq 0 \qquad \forall i \in I \tag{2.7e}$$

2.3.3 Proposed Decomposition Method for IPPO

Bilevel optimization problems are NP-hard, and it is not easy to solve. However, our proposed predictive and prescriptive integrated methodology has a unique feature. All the defined variables belong to their own scenario except the regression parameters. Regression parameters are common for all scenarios. After converting bilevel to a single-level problem by applying KKT conditions, our model becomes a two-stage mixed-integer program whose first stage variables are regression parameters. Here, we create copies of regression parameters across all scenarios and make the problem fully scenario-based decomposable, but we need to include a non-anticipativity or implementability constraint to ensure all regression parameters are equal to each other for all scenarios. Progressive hedging algorithm (PHA) proposed by Rockafellar and Wets [18] can be used as decomposition techniques for our two-stage mixed-integer problem. In our framework, we will provide the best candidate solution as initial regression parameters for depicted Figure 3, and PHA solves all scenario problems independently, then we wıll update regression parameters ıteratıvely. These steps repeat until convergence is satisfied.

$$\min_{Z^f, Z^s} \; c^f Z^f + \sum_{n \in N} c^s Z_n^s \tag{2.8a}$$

$$\text{s.t.} \quad aZ^f \geq b \tag{2.8b}$$

$$t_n Z^f + w_n Z_n^s \geq r_n \qquad \forall n \in N \tag{2.8c}$$

For better understanding, let's consider the formulation in (2.8a)-(2.8c), Z^f indicates first stage variable, and Z_n^s indicates second stage variable for n^{th} scenario.

$$\min_{Z^f, Z^s, Z^d} \sum_{n \in N} (c^f Z_n^d + c^s Z_n^s) \tag{2.9a}$$

$$\text{s.t.} \quad aZ_n^d \geq b \qquad \forall n \in N \tag{2.9b}$$

$$t_n Z_n^d + w_n Z_n^s \geq r_n \qquad \forall n \in N \tag{2.9c}$$

$$Z_n^d - Z^f = 0 \qquad \forall n \in N \tag{2.9d}$$

In the formulation (2.9a)-(2.9d), first stage variable is duplicated, and Z_n^d variables are created for each scenario, but they are linked via non-anticipativity constraint 2.9d. By relaxing constraint 2.9d, all scenarios can be easily solved in a parallel.

PHA iterates and converges to a common solution taking into account all the scenarios belonging to the original problem. We show the details and steps for basic PHA in Algorithm 1. Let $\rho > 0$ be penalty factor, δ be stopping criteria, and ω be dual prices for non-anticipativity constraint 2.9d.

Algorithm 1: The Progressive Hedging Algorithm

1 ***Initialization***

2 $k = 0$

3 $Z_n^{d,k} = \underset{Z^d, Z^s}{\operatorname{argmin}}(c^f Z_n^d + c^s Z_n^s)$ *s.t. 2.9b-2.9c* $\forall n \in N$

4 $\overline{Z}^k = \dfrac{\sum_{n \in N} Z_n^{d,k}}{|N|}$

5 $\omega_n^k = \rho(Z_n^{d,k} - \overline{Z}^k)$ $\forall n \in N$

6 ***Iteration Update***

7 $k = k + 1$

8 ***Decomposition***

9 $Z_n^{d,k} = \underset{Z^d, Z^s}{\operatorname{argmin}}(c^f Z_n^d + c^s Z_n^s + \omega_n^{k-1} Z_n^d + \frac{\rho}{2}(Z_n^d - \overline{Z}^{k-1})^2$ *s.t. 2.9b-2.9c* $\forall n \in N$

10 $\overline{Z}^k = \dfrac{\sum_{n \in N} Z_n^{d,k}}{|N|}$

11 $\omega_n^k = \omega_n^{k-1} + \rho(Z_n^{d,k} - \overline{Z}^k)$ $\forall n \in N$

12 ***Convergence Check***

13 *If all scenario solutions $Z_n^{d,k}$ are equal with at most δ deviation, stop. Else, go to step 6.*

2.4 Experimental Study

In this part, we discuss why and how we select the predictive and prescriptive models, and then we will introduce the parameters and variables of these two tasks. Next, we explain the data creation process step by step, and we will show the formulation of the integrated predictive and prescriptive task.

2.4.1 Prescriptive and Predictive Model Selection

We validate the performance of integrated predictive and prescriptive methodology and compare it with various well-known and recently developed methods. We perform numerical experiments on two different prescriptive models. The first one is a well-known newsvendor problem used by Ban and Rudin [2], but we extend it from a single product to a multi-product newsvendor problem ($d_p = 12$ products), and we use different costs for each scenario (production, holding, and backordering) instead of fix costs in order to increase the complexity of the problem. Extensive form of classical newsvendor problem with multi-product is expressed as formulated in (2.10a)-(2.10d).

$$\min_{Q,U,O} \quad \frac{1}{|J|}\sum_{j\in J}\left[\frac{1}{|N|}\sum_{n\in N}(c_{n,j}Q_j + b_{n,j}U_{n,j} + h_{n,j}O_{n,j})\right] \tag{2.10a}$$

$$\text{s.t.} \quad U_{n,j} \geq \tilde{Y}_{n,j} - Q_j \qquad \forall j \in J, \forall n \in N \tag{2.10b}$$

$$O_{n,j} \geq Q_j - \tilde{Y}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.10c}$$

$$Q_j, U_{n,j}, O_{n,j} \geq 0 \qquad \forall j \in J, \forall n \in N \tag{2.10d}$$

Decision Variables for Newsvendor Problem

Q_j Amount of regular order done in advance for product j

$U_{n,j}$ Amount of shortage for product j in scenario n

$O_{n,j}$ Amount of surplus for product j in scenario n

Parameters for Newsvendor Problem

$c_{n,j}$ Cost of order for product j in scenario n

$b_{n,j}$ Cost of backordering for product j in scenario n

$h_{n,j}$ Cost of hold for product j in scenario n

$\tilde{Y}_{n,j}$ Amount of observed demand for product j in scenario n

The second prescriptive model is two-stage shipment planning problem leveraged by Bertsimas and Kallus [7] where there is a network between $d_w = 4$ warehouses and $d_l = 12$ locations. The goal is to produce and hold a product at a cost in warehouses to satisfy the future demand of locations. Then the product is shipped, when needed, from warehouses to locations with transportation costs. In case the current total supply in warehouses does not satisfy the demand of locations, the last-minute production takes place at a higher cost. The extensive form of two-stage shipment problem is formulated as follow in (2.11a)-(2.11d).

$$\min_{Z,T,S} \quad \sum_{i\in I} p_1 Z_i + \frac{1}{|N|}\sum_{n\in N}\left[\sum_{i\in I} p_2 T_{n,i} + \sum_{i\in I}\sum_{j\in J} c_{n,i,j} S_{n,i,j}\right] \tag{2.11a}$$

$$\text{s.t.} \quad \sum_{i\in I} S_{n,i,j} \geq \tilde{Y}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.11b}$$

$$\sum_{j\in J} S_{n,i,j} \leq Z_i + T_{n,i} \qquad \forall i \in I, \forall n \in N \tag{2.11c}$$

$$Z_i, \quad T_{n,i}, \quad S_{n,i,j} \geq 0 \qquad \forall i \in I, \forall j \in J, \forall n \in N \tag{2.11d}$$

Decision Variables for Two-Stage Shipment Problem

Z_i Amount of production done in advance at warehouse i

$T_{n,i}$ Amount of production done last minute at warehouse i in scenario n

$S_{n,i,j}$ Amount of shipment from warehouse i to location j in scenario n

Parameters for Two-Stage Shipment Problem

p_1 Cost of production done in advance at warehouse

p_2 Cost of production done last minute at warehouses

$c_{n,i,j}$ Cost of shipment from warehouse i to location j in scenario n

$\tilde{Y}_{n,j}$ Amount of observed demand at location j in scenario n

As for the predictive model, since we embed the predictive model inside of the prescriptive model, we choose the linear regression model as in 2.12 to maintain the linearity of the prescriptive model. However, other predictive methodologies still can be applied to capture non-linearity outside of this integrated framework as preprocess, and dimensionality can be reduced between feature variables and responses, especially in high dimensional data as built in deep learning. These converted fea-

ture variables can be embedded inside of prescriptive model via linear regression again as described above.

$$\hat{Y}_{n,j} = \beta_{j,0} + \sum_{p \in P} \beta_{j,p} \tilde{X}_{n,p} \quad \forall j \in J \quad \forall n \in N \tag{2.12}$$

2.4.2 Data Generation

In both experiments, we randomly generate feature variables of predictive model based on a $d_x = 4$ dimensional multivariate normal distribution with size of $n = 2000$ observations, $\tilde{X} \in \mathbb{R}^{n \times d_x}$, i.e., $\tilde{X} \sim N(\mu, \Sigma)$, where $\mu = [1, 0, 0, 0]$ and $\Sigma = [[0, 0, 0, 0], [0, 1, 0.5, -0.5], [0, 0.5, 1, -0.5], [0, -0.5, 0.5, 1]]$. Then, we choose the true parameters of our predictive model, linear regression in our case, as $\beta \in \mathbb{R}^{d_x \times d_l}$ matrix including slopes and intercepts. Next, we calculated observed response according to the model $\tilde{Y} = \tilde{X}\beta + \varepsilon$, where ε is independently generated noise term and follows normal distribution, i.e., $\varepsilon \sim N(0, \sigma)$. Here the standard deviation of added noise controls the correlation between feature values and responses (responses represent the demand in both prescriptive problems). To see the behavior of our method and other methods, we have employed 10 different noise standard deviations, thus we create 10 different feature and response pairs with different correlations. we measure these correlations based on R-Square value of a linear regression model. As for shipment cost, we randomly simulate its matrix as from warehouse i to location j based on uniform distribution $c_{n,i,j} \sim U(0, 30)$ for each scenario. In newsvendor problem, we create order, backordering, and holding costs again based on uniform

distribution $c_{n,j} \sim U(0, 300)$, $b_{n,j} \sim U(0, 3000)$, and $h_{n,j} \sim U(0, 150)$ for each scenario and product, respectively. Out of created $n = 2000$ observations, we randomly choose train, validation, and test sets with size of $70, 15, 15$, respectively. This splitting process is repeated by 30 times, and all results are reported based on the average cost of these 30 replications in both problems.

2.4.3 IPPO Formulations

We modify newsvendor and two-stage shipment problems here for the integration process. First, we introduce β variable to make predictions for demand via linear regression. Then we also introduce counterpart decision variables of original variables in newsvendor and shipment models because these counterpart decision variables will be made based on predicted demands. In our lower-level, we make decisions based on the output of the predictive algorithm, linear regression, and submit these decisions to the upper-level, so that we can evaluate the quality of these decisions based on true parameters in 2.13d and 2.14d.

$$\min_{Q,U,O,\beta} \quad \frac{1}{|J|}\sum_{j\in J}\left[\frac{1}{|N|}\sum_{n\in N}(c_{n,j}Q_{n,j}+b_{n,j}U_{n,j}+h_{n,j}O_{n,j})\right] \tag{2.13a}$$

$$\text{s.t.} \quad U_{n,j} \geq \tilde{Y}_{n,j} - Q_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.13b}$$

$$O_{n,j} \geq Q_{n,j} - \tilde{Y}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.13c}$$

$$Q_{n,j} = \hat{Q}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.13d}$$

$$Q_{n,j}, U_{n,j}, O_{n,j} \geq 0 \qquad \forall j \in J, \forall n \in N \tag{2.13e}$$

$$\min_{\hat{Q},\hat{U},\hat{O}} \frac{1}{|J|}\sum_{j\in J}\left[\frac{1}{|N|}\sum_{n\in N}(c_{n,j}\hat{Q}_{n,j}+b_{n,j}\hat{U}_{n,j}+h_{n,j}\hat{O}_{n,j})\right] \tag{2.13f}$$

$$\text{s.t.} \quad \hat{U}_{n,j} \geq \hat{Y}_{n,j} - \hat{Q}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.13g}$$

$$\hat{O}_{n,j} \geq \hat{Q}_{n,j} - \hat{Y}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{2.13h}$$

$$\hat{Q}_{n,j}, \hat{U}_{n,j}, \hat{O}_{n,j} \geq 0 \qquad \forall j \in J, \forall n \in N \tag{2.13i}$$

Detailed formulation of integrated newsvendor problem and two stage shipment problem is provided below in (2.13a)-(2.13i) and (2.14a)-(2.14i), respectively. If controlling generalization error is needed, one of the recommendations formulated in Section 2.3.2 can be included in these models.

In both formulations, there will be a trade-off between the lower-level and the upper-level problems, such that the lower-level problem minimizes its own cost based on β variables provided by the upper-level problem, and the upper-level problem minimizes its own objective value based on prescriptions provided by the lower-level problem. To be able to solve the bilevel model, we need to add optimality conditions for the lower-level problem based on KKT conditions. We introduce dual variables

for lower-level constraints and write these conditions, but KKT conditions bring non-linearity because of complementary slackness, so new binary variables can be defined, and SOS constraints and the big M method can be used. After introducing KKT conditions, the predictive task integrated two-stage shipment problem becomes a mix integer problem with single-level, and this formulation is given in (2.4a)-(2.4g).

$$\min_{Z,T,S\beta} \frac{1}{|N|} \sum_{n\in N} (\sum_{i\in I} p_1 Z_{n,i} + \sum_{i\in I} p_2 T_{n,i} + \sum_{i\in I}\sum_{j\in J} c_{n,i,j} S_{n,i,j}) \tag{2.14a}$$

$$\text{s.t.} \quad \sum_{i\in I} S_{n,i,j} \geq \tilde{Y}_{n,j} \qquad \forall n \in N, \forall j \in J \tag{2.14b}$$

$$\sum_{j\in J} S_{n,i,j} \leq Z_{n,i} + T_{n,i} \qquad \forall n \in N, \forall i \in I \tag{2.14c}$$

$$Z_{n,i} = \hat{Z}_{n,i} \qquad \forall n \in N, \forall i \in I \tag{2.14d}$$

$$Z_{n,i}, \quad T_{n,i}, \quad S_{n,i,j} \geq 0, \quad \beta \quad free \qquad \forall n \in N, \forall i \in I, \forall j \in J \tag{2.14e}$$

$$\min_{\hat{Z},\hat{T},\hat{S}} \frac{1}{|N|} \sum_{n\in N} (\sum_{i\in I} p_1 \hat{Z}_{n,i} + \sum_{i\in I} p_2 \hat{T}_{n,i} + \sum_{i\in I}\sum_{j\in J} c_{n,i,j} \hat{S}_{n,i,j}) \tag{2.14f}$$

$$\text{s.t.} \quad \sum_{i\in I} \hat{S}_{n,i,j} \geq \hat{Y}_{n,j} \qquad \forall n \in N, \forall j \in J \tag{2.14g}$$

$$\sum_{j\in J} \hat{S}_{n,i,j} \leq \hat{Z}_{n,i} + \hat{T}_{n,i} \qquad \forall n \in N, \forall i \in I \tag{2.14h}$$

$$\hat{Z}_{n,i}, \quad \hat{T}_{n,i}, \quad \hat{S}_{n,i,j} \geq 0 \qquad \forall n \in N, \forall i \in I, \forall j \in J \tag{2.14i}$$

2.4.4 Convergence to Stochastic Optimization

The Equation in 2.12 defines the linear regression where the output is a weighted combination of inputs plus an intercept. Linear regressions are generally trained

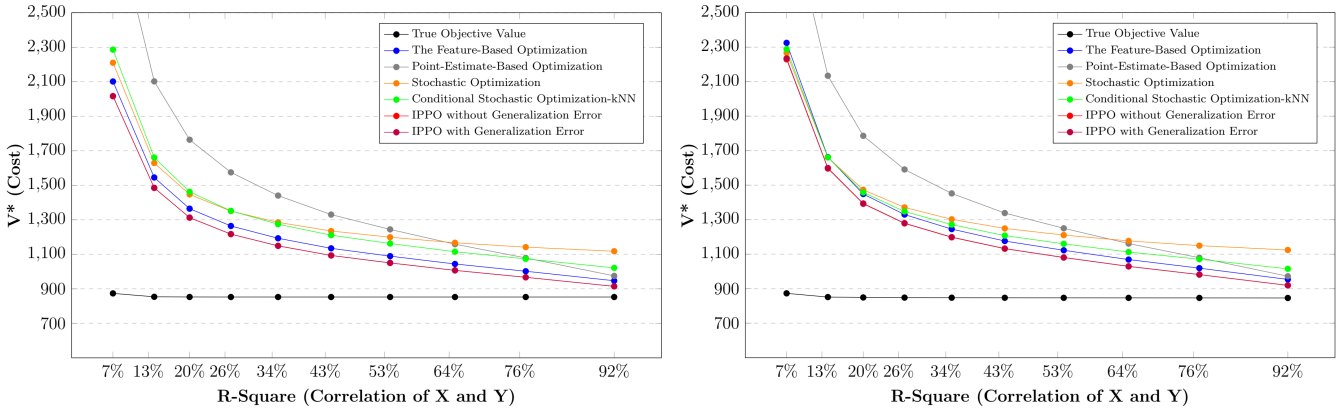
based on mean squared deviations. In our proposed integrated model, this Equation in 2.12 produces predictions for each scenario, and lower-level objective and constraints prescribes decisions based on these predictions as seen in (2.13f)-(2.13i) and (2.14f)-(2.14i) for each scenario again. Suppose the level of correlation between observed feature data $\tilde{X}$ and response $\tilde{Y}$ goes to zero. In that case, slopes of a predictive model in 2.12 or the feature variable contribution goes to zero, and predictions will be all equal to each other thanks to intercepts. The same prediction for all scenarios will prescribe the same decisions across all scenarios as seen in forwarded decisions from lower-level to upper-level in (2.13d) and (2.14d). This assumption converts our integrated methodology to a single-level problem and becomes scenario formulation of a two-stage stochastic problem as shown in (2.9a)-(2.9d).

2.5 Computational Results

This section discusses the performance of our integrated methodology and other methods under different circumstances. All results for these experiments are obtained from Gurobi python API [13]. We compared results of various well-known methods like Point-Estimate-Based Optimization, Stochastic Optimization, and recent methods like Conditional Stochastic Optimization (kNN), and The Feature-Based Optimization by Bertsimas and Kallus [7], and by Ban and Rudin [2], respectively. We investigate the behaviors of these methods under different correlations between observed features $\tilde{X}$ and responses $\tilde{Y}$, and evaluate the performance of validation data set to see if generalization error is needed.

2.5.1 Newsvendor Problem

First and common feature of all methods, as we see in Figure 4, solution quality improves when we increase the correlation between features (side info) and responses, and this is expected because the more information is provided, the better results are obtained. However, improvement rates are different in each method. Second, kNN from Bertsimas and Kallus [7] gives better result compared to stochastic optimization (k neighbors value is optimized over validation data set). It gives a better solution because it leverages specific neighbors in train data. Instead of minimizing expected cost over the whole train data set, it eliminates irrelevant scenarios and seeks optimal solutions by minimizing expected cost around neighbors.



(a) Train Data Set

(b) Test Data Set

Figure 4: Comparison of Different Methods for Newsvendor Problem

We observe that value of k neighbors increases when correlation level between observed $\tilde{X}$ and $\tilde{Y}$ decreases. Since this correlation decreases, kNN tends to minimize expected cost over more train data as we reported in Table 1 and 2. We did not

include result of Elmachtoub and Grigas [12] in Figure 4 and 6 because Elmachtoub and Grigas [12] build their method when there is an uncertainty in cost vector of objective function.

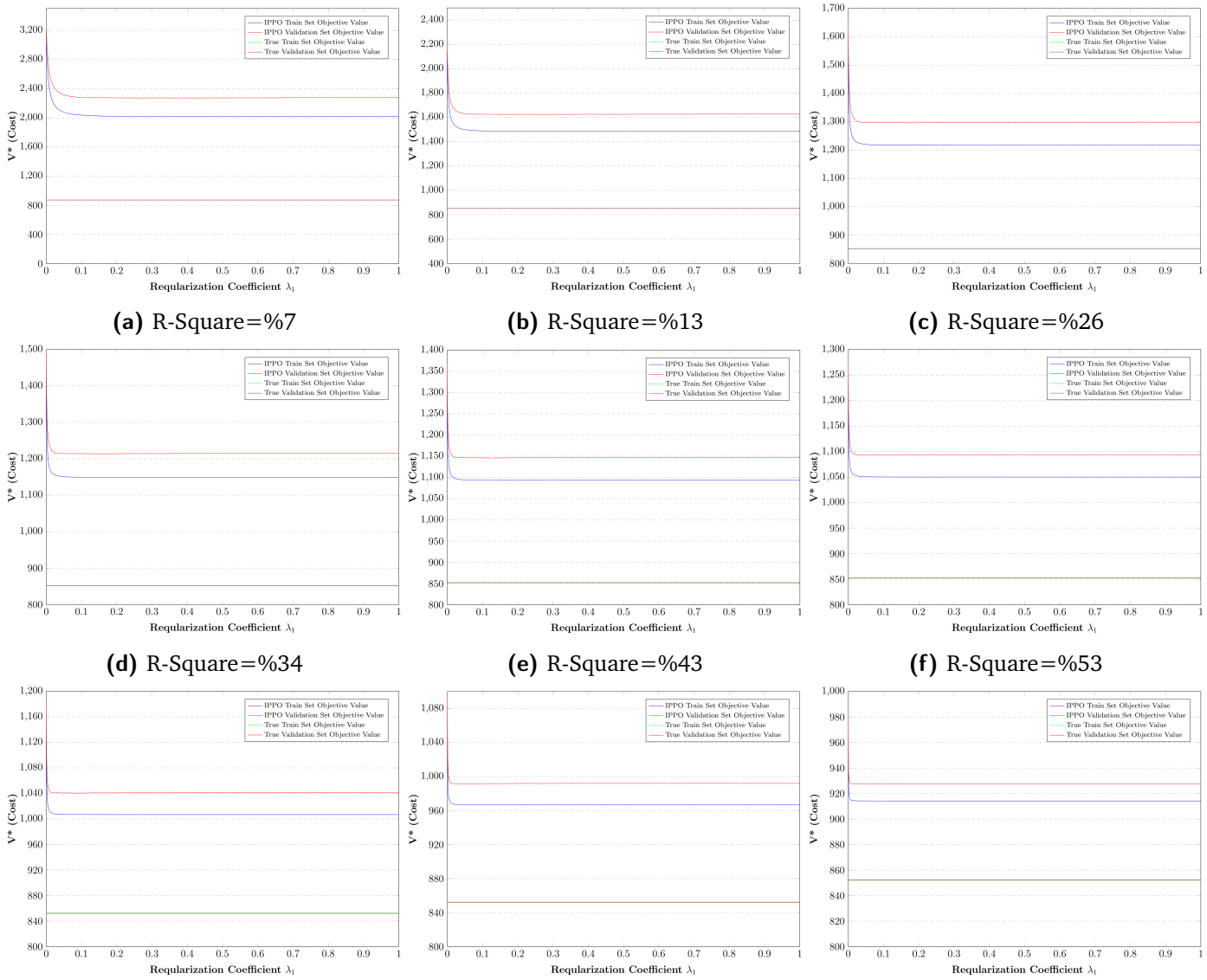
Correlation between X and Y	True Objective Values				IPPO				Performance	Optimal Regularization Parameter	Optimal Neighbors for kNN
	Mean	Max	Min	S.Dev.	Mean	Max	Min	S.Dev.			
%7	873.5	924.8	815.0	27.2	2016.3	2118.5	1897.9	58.3	%7	1	41
%13	853.7	892.9	813.6	21.0	1484.9	1549.1	1410.3	35.0	%10	1	41
%20	852.7	889.5	814.5	19.6	1312.5	1369.6	1252.7	28.3	%11	1	31
%26	852.5	889.2	815.3	19.0	1216.8	1270.1	1164.7	25.0	%13	1	28
%34	852.5	889.4	816.1	18.6	1148.9	1199.4	1102.0	22.9	%15	1	28
%43	852.5	889.6	816.8	18.3	1093.3	1141.6	1050.7	21.4	%17	1	22
%53	852.5	889.8	817.3	18.2	1050.1	1096.5	1010.9	20.4	%20	1	16
%64	852.5	890.0	817.8	18.1	1006.9	1051.5	971.0	19.5	%24	1	16
%76	852.5	890.2	818.3	18.0	966.7	1009.7	933.4	18.8	%31	1	10
%92	852.5	890.4	818.9	17.9	914.2	955.0	881.2	18.2	%53	1	5

Table 1: Newsvendor Problem Train Data Set Statistical Values

Correlation between X and Y	True Objective Values				IPPO				Performance	Optimal Regularization Parameter	Optimal Neighbors for kNN
	Mean	Max	Min	S.Dev.	Mean	Max	Min	S.Dev.			
%7	873.0	972.6	688.7	59.2	2229.5	2546.0	1987.2	140.0	%7	1	41
%13	851.3	929.5	711.3	44.0	1596.6	1774.5	1431.6	80.2	%9	1	41
%20	849.1	924.9	724.4	40.5	1391.9	1527.4	1248.6	63.4	%10	1	31
%26	848.2	922.4	731.7	39.3	1278.3	1391.2	1148.2	55.0	%12	1	28
%34	847.7	920.9	737.2	38.7	1197.6	1293.9	1075.7	50.0	%14	1	28
%43	847.3	921.2	741.8	38.4	1131.6	1214.2	1016.3	46.2	%16	1	22
%53	847.0	925.1	745.5	38.3	1080.3	1154.2	970.4	43.7	%18	1	16
%64	846.7	929.1	749.1	38.4	1029.0	1103.7	925.0	41.8	%22	1	16
%76	846.4	932.7	752.4	38.6	981.3	1061.2	882.6	40.5	%28	1	10
%92	846.1	937.5	756.8	38.9	919.0	1007.0	827.2	39.5	%47	1	5

Table 2: Newsvendor Problem Test Data Set Statistical Values

Since IPPO methodology fully leverages side information and evaluates decisions regarding true parameters inside the integrated framework, it gives the best solution. We did not observe an over-fitting issue in the newsvendor problem, and there is no need to control generalization error. As we report in Figure 5, regardless of the correlation level between observed $\tilde{X}$ and $\tilde{Y}$, validation and train performance improve constantly until $\lambda_1 = 1$ where no generalization is needed. That is why IPPO with and without generalization error in Figure 4 overlaps.



(a) R-Square=%7 **(b)** R-Square=%13 **(c)** R-Square=%26

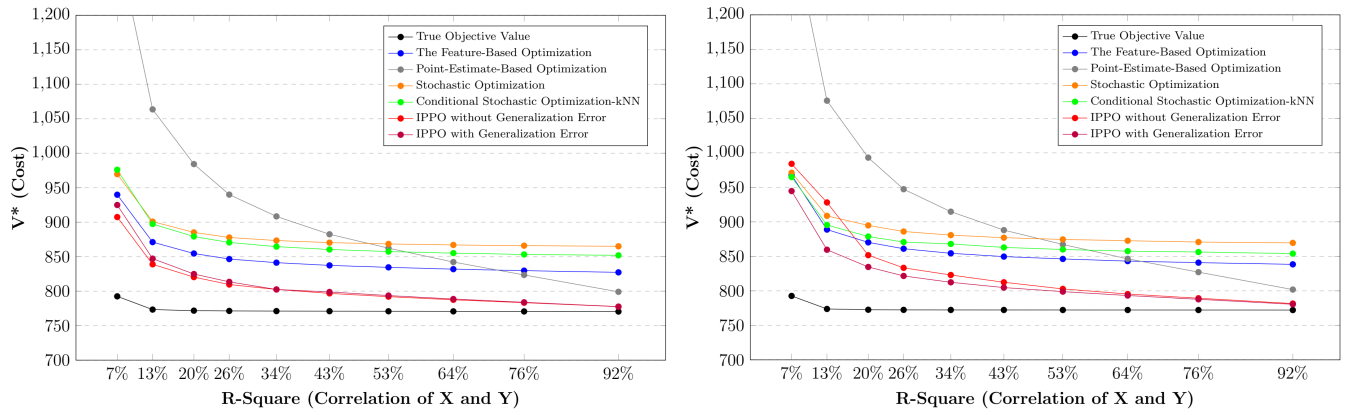
(d) R-Square=%34 **(e)** R-Square=%43 **(f)** R-Square=%53

(g) R-Square=%64 **(h)** R-Square=%76 **(i)** R-Square=%92

Figure 5: Newsvendor Problem Train and Validation Performance Over Regularization Parameter λ_1 For Different R-Square Values Between X and Y

2.5.2 Two-Stage Shipment Problem

We observe similar behaviors here as in the newsvendor problem. The main difference in the two-stage shipment problem is IPPO overfits in especially low level X and Y correlations since IPPO methodology fully leverages side information and searches the best solution for train data set. As we see the effect of regularization coefficient (λ_1) in Figure 7, train performance constantly improves, but validation performance becomes worse after some point.



(a) Train Data Set

(b) Test Data Set

Figure 6: Comparison of Different Methods for Shipment Problem

In figure 6, we report train and test performance with and without controlling generalization error. Although train performance is better than other methods without controlling generalization error, a better solution is achieved via optimizing the regularization coefficient (λ_1).

Some detailed statistical measures are given in Table 3 for train set and in Table 4 for test set. The performance column is calculated in Equation 2.15 based on the second-best method, which is Feature-Based Optimization. These results are based on 30 replications, and the mean column is presented in Figure 6.

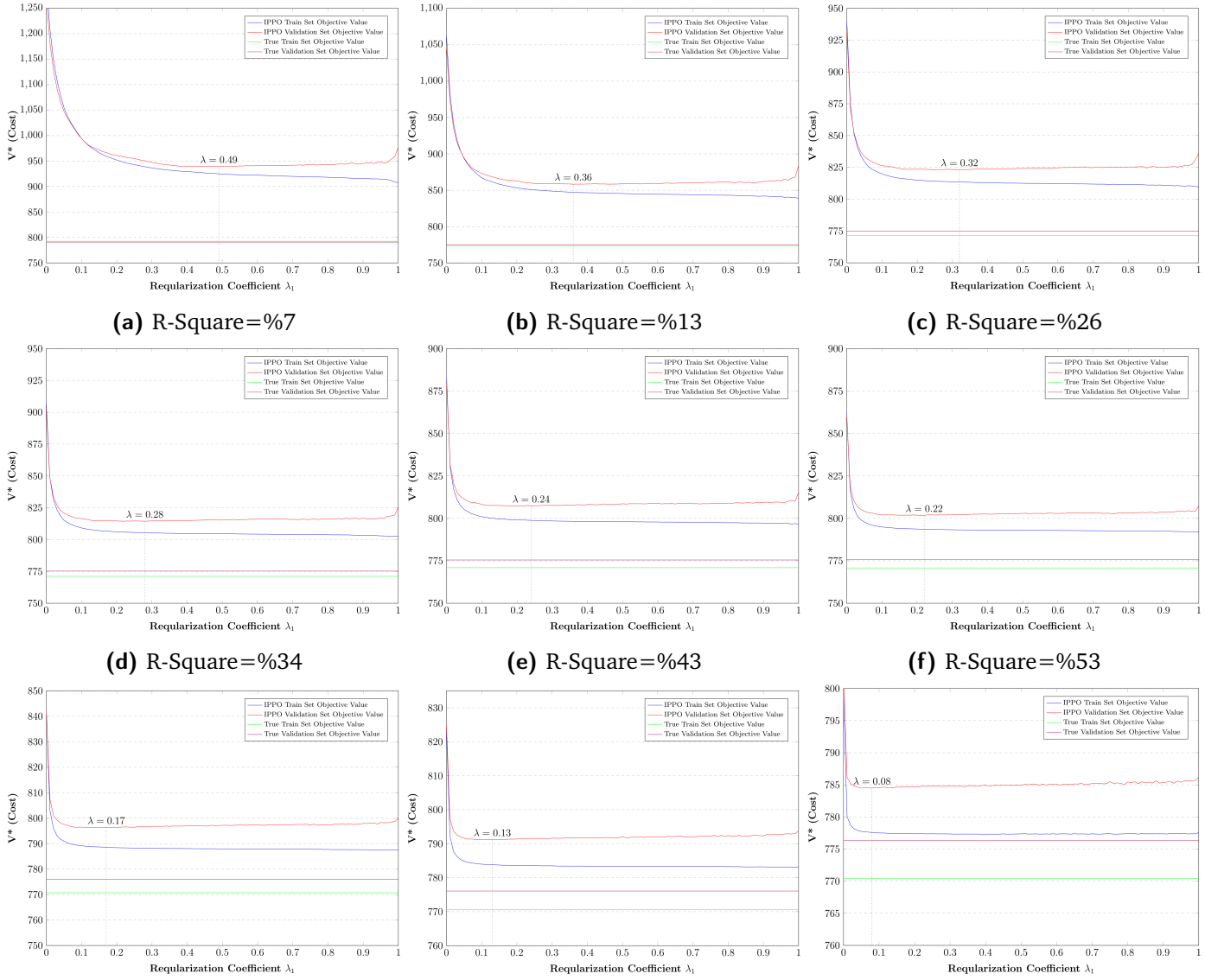
$$Performance = \left[\frac{The\ Feature\ Based - True\ Objective}{IPPO - True\ Objective} - 1\right] 100 \qquad (2.15)$$

Correlation between X and Y	True Objective Values				IPPO				Performance	Optimal Regularization Parameter	Optimal Neighbors for kNN
	Mean	Max	Min	S.Dev.	Mean	Max	Min	S.Dev.			
%7	792.5	855.7	738.0	22.2	925.0	988.4	864.1	28.4	%11	0.49	50
%13	773.3	820.2	728.2	18.6	847.2	889.9	798.1	21.2	%32	0.36	26
%20	771.7	812.5	730.8	17.5	824.9	862.0	781.6	18.7	%56	0.38	28
%26	771.3	808.7	732.7	17.0	813.5	847.9	773.0	17.9	%78	0.32	28
%34	771.1	806.1	734.0	16.7	805.4	838.0	766.8	17.3	%105	0.28	17
%43	771.0	804.0	735.0	16.5	798.9	830.0	761.6	16.9	%138	0.24	15
%53	770.8	802.4	735.8	16.4	793.6	823.7	757.7	16.7	%179	0.22	15
%64	770.7	801.0	736.7	16.3	788.6	818.7	753.7	16.5	%242	0.17	15
%76	770.6	801.7	737.4	16.2	783.8	814.9	750.1	16.3	%347	0.13	15
%92	770.5	802.7	737.2	16.1	777.6	809.8	743.6	16.1	%695	0.08	13

Table 3: Shipment Problem Train Data Set Statistical Values between True Objective and IPPO Objective

Correlation between X and Y	True Objective Values				IPPO				Performance	Optimal Regularization Parameter	Optimal Neighbors for kNN
	Mean	Max	Min	S.Dev.	Mean	Max	Min	S.Dev.			
%7	792.6	903.0	691.6	51.3	944.6	1089.4	821.4	67.6	%15	0.49	50
%13	773.8	857.3	690.3	40.0	859.6	932.8	766.6	46.8	%34	0.36	26
%20	772.6	845.3	695.1	36.3	834.6	901.8	751.0	40.0	%57	0.38	28
%26	772.4	838.6	697.9	34.5	821.6	886.0	742.4	37.2	%80	0.32	28
%34	772.4	834.8	700.3	33.4	812.4	874.8	736.5	35.4	%105	0.28	17
%43	772.3	833.2	702.3	32.5	804.8	865.6	731.1	34.0	%138	0.24	15
%53	772.3	831.9	703.6	31.9	798.9	858.5	728.0	33.0	%178	0.22	15
%64	772.3	830.6	704.1	31.4	793.2	851.4	724.3	32.1	%238	0.17	15
%76	772.2	829.4	704.6	30.9	787.7	844.8	720.9	31.4	%343	0.13	15
%92	772.2	827.8	705.3	30.4	780.6	836.1	714.4	30.6	%691	0.08	13

Table 4: Shipment Problem Test Data Set Statistical Values between True Objective and IPPO Objective



(a) R-Square=%7 **(b)** R-Square=%13 **(c)** R-Square=%26

(d) R-Square=%34 **(e)** R-Square=%43 **(f)** R-Square=%53

(g) R-Square=%64 **(h)** R-Square=%76 **(i)** R-Square=%92

Figure 7: Shipment Problem Train and Validation Performance Over Regularization Parameter λ_1 For Different R-Square Values Between X and Y

2.6 Conclusion

This work provides an integrated framework that fully utilizes feature data to predict responses to take the best actions in a prescriptive task. This methodology can be employed in any prescriptive task whose parameters are uncertain with feature data as long as the prescriptive model is convex since KKT optimality conditions require duality, hence limited to convexity. However, it still can be approximated to integer programming. In order to optimize predictive and prescriptive tasks simultaneously, the predictive task should appear in a linear form, so this limits the complexity of the prediction task. However, alternative methods can be used to capture non-linearity outside of the proposed framework. Our purpose is to train the predictive model based on the not predictive error but also based on a prescriptive error that assesses decision variables provided by predicted responses with respect to true responses. Bilevel models are NP-hard problems, and it is not easy to solve, but our framework is a special case, where all decision variables are based on scenarios except the parameters of the predictive model. Thus, this formulation can be easily decomposed by decoupling the predictive model parameters and solved by PHA step by step. Also, we demonstrate the behavior of our and other frameworks under different correlations of feature and response data. Finally, we provide our results and compare them to traditional and recently introduced methods, and we perform well with and even without controlling generalization error.

CHAPTER 3 INTEGRATED LEARNING OF PREDICTIVE AND PRESCRIPTIVE TASKS

3.1 Introduction

In recent years, a growing number of businesses in the same industry or service area create a competitive environment, and this competition leads parties to search for new and feasible strategies. These strategies aim to lower costs and improve business outcomes by making proper decisions to survive. Typical decision problems rely on solving constrained optimization problems. However, most real-world problems contain uncertain parameters. At this point, traditional optimization methods minimizing expected cost step in to handle uncertainty. Alternatively, in the presence of feature data pairs for uncertain parameters, a decision-maker can make prescriptions by solving the optimization problem with the estimated parameters.

In a data-rich world, machine learning algorithms are more involving in improving the quality of decisions. Machine learning is considered a branch of artificial intelligence and a method of data analysis. Machine learning algorithms are explicitly programmed to build algorithms and neural network models based on sample data to learn from data. Although machine learning may seem like a very recent research area, the history and the theory of machine learning date back to the 1950s. One of the approaches used in machine learning algorithms is a neural network that translates the input into the desired output using a computational learning system based on a sequence of network functions.

The recent developments and computing technology behind neural networks provided practical solutions in predictive analytics, from regression tasks to classification tasks. Classic neural networks use feed-forward network architectures such as the multi-layer perceptron inspired by studies of biological networks. In the general framework of feed-forward network architectures, data travel over hidden layers and transform to another shape in neurons with the help of different activation functions (changes based on needs). The output layer contains a loss function that minimizes the error between predicted and actual responses to make predicted and actual values similar. With this, neural networks establish a non-linear mapping between input variables and a set of output variables. As for the learning process, the loss function calculates the error and forwards it to previous layers to update weights via the back-propagation algorithm. This update process resumes until no further improvement is achieved. Recently proposed methods on successful applications show the power of the learning capability of neural networks. Therefore, this chapter aims to develop an integrated neural network framework that combines predictive and prescriptive tasks in a machine learning environment to provide high-quality decisions.

We propose different neural network frameworks for integrating predictive and prescriptive tasks. Our first proposed method is interested in incorporating a predictive task and a prescriptive task where objective function and constraints of the prescriptive task are well defined and known to decision-makers. We explicitly minimize the objective function, satisfy all the constraints within the neural network model, and prescribe the best decisions based on the fed predictions from the predic-

tive task. Our other proposed framework is interested in integrating two predictive models. As in most of the common predictive tasks, neural networks are trained over specific input parameters. However, in some business models, not all of the predictive algorithms have particular input values. Some may require full or partial inputs from other predictive models. For instance, there are uncontrollable factors affecting customer's demand, and customer demand's along with some other parameters effect retailer's demand. There are two tasks in these models: one model between uncontrollable factors and customer demands, and the other is between customer demands and retailer demands. Moreover, our second proposed framework is also interested in some prescriptive models involving an objective function and constraints which are unknown or not well established where decision-makers make the calls based on past experiences.

3.2 Related Work

In recent years, research interest in machine learning algorithms has grown at a significant rate. Machine learning as a branch of artificial intelligence extracts meaningful information and predicts the output of a system by processing historical data. Most of the machine learning algorithms are built on top of optimization models based on a training criterion to find the parameters of learning algorithms. The performance of these learning parameters depends on the power of the optimization models. On the other hand, real-world optimization problems mostly contain uncertain parameters that machine learning models mostly predict. In this literature

review, we focus on proposed works that handle the uncertainty in decision problems by leveraging machine learning models. These methods are that a predictive algorithm produces the uncertain parameters of a decision problem as a point estimate or set of estimates, optimize the decision problem over the point estimate, or directly integrate predictive algorithm under decision problem to learn uncertain parameters with respect to optimal decisions. In a typical machine learning model, train performance is measured based on either classifications rate or mean squared error. The proposition of that the predictive tasks should be trained based on a prescriptive criterion was recommended almost 25 years ago. Although some earlier works recommended that the right criteria of predictive learning should be derived from the decision stage, there has been practically no contribution in this field since then. Bengio [5] is one of the first works considering this gap and building an integrated framework for both prediction and prescription tasks. He emphasized the importance of the evaluation criteria of predictive tasks. His neural network model was not designed to minimize prediction error but instead to maximize a financial task where those noisy predictions are used as input. Maza et al. [10] propose a similar work to find optimal trading decisions via neural network. As an extension of Ban and Rudin's work[2], Oroojlooyjadid et al. [17], and Zhang and Gao [25] propose a neural network methodology to predict demand while minimizing holding and shortage cost to capture the non-linearity between auxiliary data and demand. However, the most common integrated approach is applying the financial training of neural networks to make trading decisions and newsvendor problems to find optimal

quantity. Another neural network-based integrated task is proposed by Donti et al. [11], and primarily they focus on quadratic stochastic optimization problems since they are tuning neural network parameters by differentiating the optimization solution to a stochastic programming problem. On the other hand, Wang [24] shows that neural networks are capable of generating optimal decisions to convex programming problems, and Lv et al. [16] also proposes a neural network frame to show how non-linear bilevel problems can be solved.

3.3 Integrated Predictive and Prescriptive Learning

In recent years, several applications have been employed to solve different classes of constrained optimization problems. It is shown that these artificial neural networks can solve such optimization problems efficiently. In this section, we briefly overview the artificial neural networks (ANN) and their elements. Then, we illustrate how an optimization problem with inequality constraint can be solved within an artificial neural network environment. Last, we present the integrated predictive and prescriptive algorithm to find optimal decisions by mapping feature data to the prescriptive model.

3.3.1 Theoretical Background of Artificial Neural Network

An artificial neural network is a branch of machine learning that can be used either for regression or classification purposes by aiming to find an optimal map between given an input set of data and an output. Artificial neural networks have

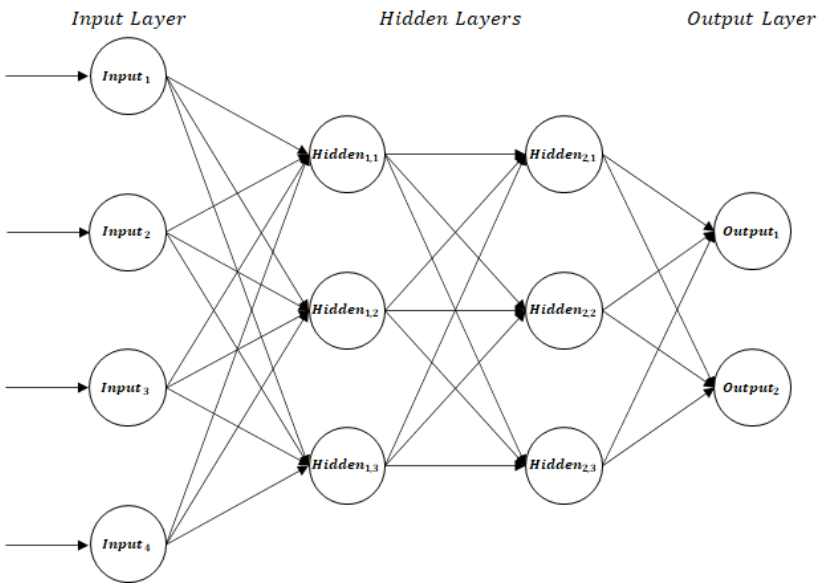


Figure 8: Artificial Neural Network Structure

broad application areas such as solving abstract business problems, sales forecasting, speech and image recognition, customer research, data validation, healthcare, time-series predictions, and more. However, since these artificial neural networks can be thought of as "black box" devices that accept inputs and produce outputs, humans suffer from understanding these models. They require careful hyperparameter tuning with a complicated training process.

We categorize the structure of a typical artificial neural network into three layers, those are input, hidden, and output layers shown in Figure 8. The input layer receives the data from external systems and delivers it to hidden layers. The hidden layer processes the coming information independently from external systems and passes the processed information to the output layer. Finally, the last layer of an artificial neural network, the output layer, produces given outputs for the program. From

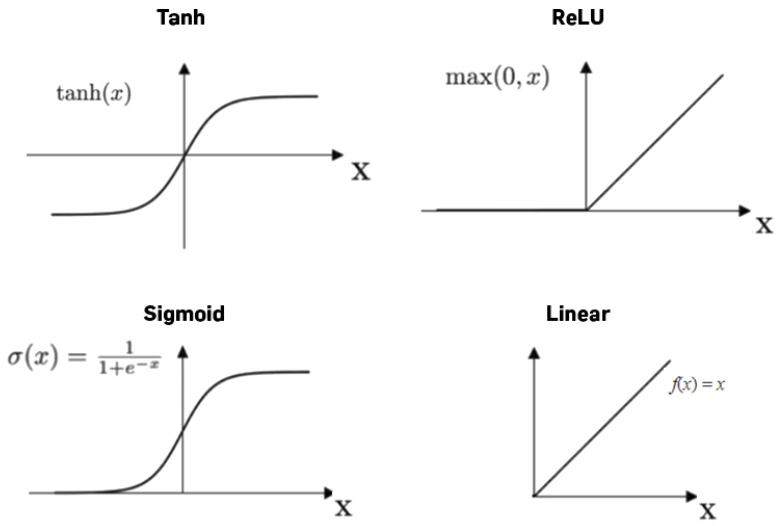


Figure 9: Common Activation Functions

the input layer to the output layer, information travels and transforms via neurons by simply calculating the weighted sum of inputs and weights, adding the bias, and executing an activation function. Activation functions transform the information in a linear or non-linear manner by using a mathematical operation. Most commonly used activation functions are shown in Figure 9.

Back-propagation is the most commonly used method for training an artificial neural network. The back-propagation algorithm constitutes the essence of the training process of an artificial neural network. Based on the error rate obtained in the previous iteration, this method calculates the gradient of a loss function with respect to all the weights in the network and trains the weights of an artificial neural network until there is no weight improvement in terms of loss.

3.3.2 Neural Network for Solving Constrained Linear Programming

In this part, we present the construction of how an artificial neural network can solve an optimization problem with inequality constraints. We begin with the basics of linear programming and its general form. The Formulation from 3.1a to 3.1c defines the standard form of a linear optimization problem. The standard form of linear programming indicates that all inequality constraints are converted to equality constraints by adding slack or subtracting surplus variable sets. The general form of the problem to be solved within an artificial neural network is

$$\min_{Z} \quad \sum_{j \in J} c_j Z_j \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{j \in J} a_{i,j} Z_j - b_i = 0 \qquad \forall i \in I \tag{3.1b}$$

$$Z_j \geq 0 \qquad \forall j \in J \tag{3.1c}$$

We rewrite the standard form of optimization model as presented in Equation 3.2. The input layer has a unit vector as feature data with the size of a number of variables in the optimization model drafted in Figure 10. Feature data is multiplied by a square weight matrix in the first layer where all the optimization problem variables laying on-diagonal entries of this weight matrix and off-diagonal entries of weight matrix are equalized to zero with a non-trainable feature.

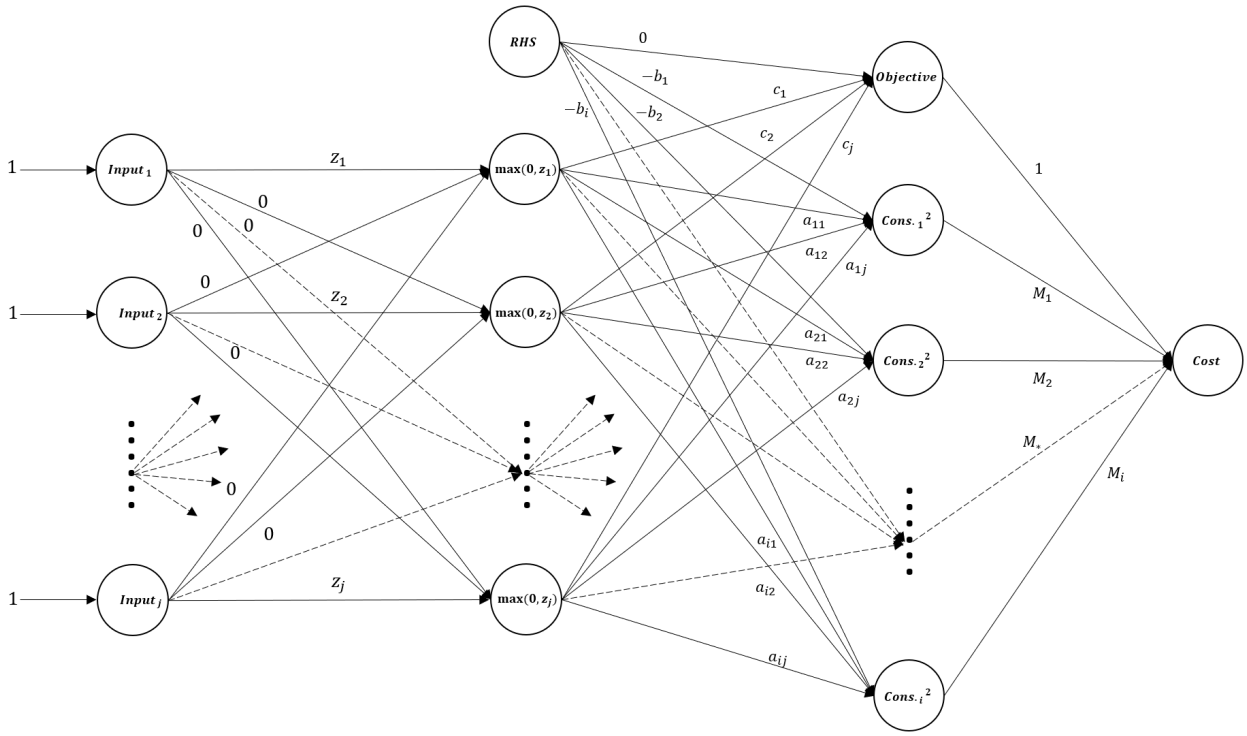


Figure 10: Neural Network Structure for Linear Programming

We use the ReLU activation function in the first layer to satisfy the non-negativity constraint if there is one. The second layer of artificial neural networks consists of neurons representing objective function and constraints. We fix the weights of the second layer of the neural network with the cost vector of the optimization problem and coefficients of constraints. Neurons representing constraints have square activation function and output of constraints multiplied by sufficiently large penalty parameter sets. These parameters might be equal to each other or different for each constraint if the constraints' importance is different. Finally, the cost function defined in Equation 3.2 as summation of the objective function and penalized quadratic loss function of constraints is minimized over decision variables of the optimization prob-

lem. With proper initialization and penalty parameter sets, neural networks converge to optimal decisions.

$$Cost = \min_{Z} \sum_{j \in J} c_j Z_j + \sum_{i \in I} \sum_{j \in J} M_i (a_{i,j} Z_j - b_i)^2 \tag{3.2}$$

In order to show the performance of the neural network solution, we use a toy example in Equations from 3.3a to 3.3c which is the linear optimization model of the famous Dakota problem with given cost, constraint coefficients, and right-hand-side values. We also show how variables converge to true solutions, how constraint violation goes to zero, and how objective value improves in Figure 11.

$$\max_{Z} \quad \sum_{j \in J} c_j^T Z_j \tag{3.3a}$$

$$\text{s.t.} \quad \sum_{j \in J} a_{i,j} Z_j \leq b_i \qquad \forall i \in I \tag{3.3b}$$

$$Z_j \geq 0 \qquad \forall j \in J \tag{3.3c}$$

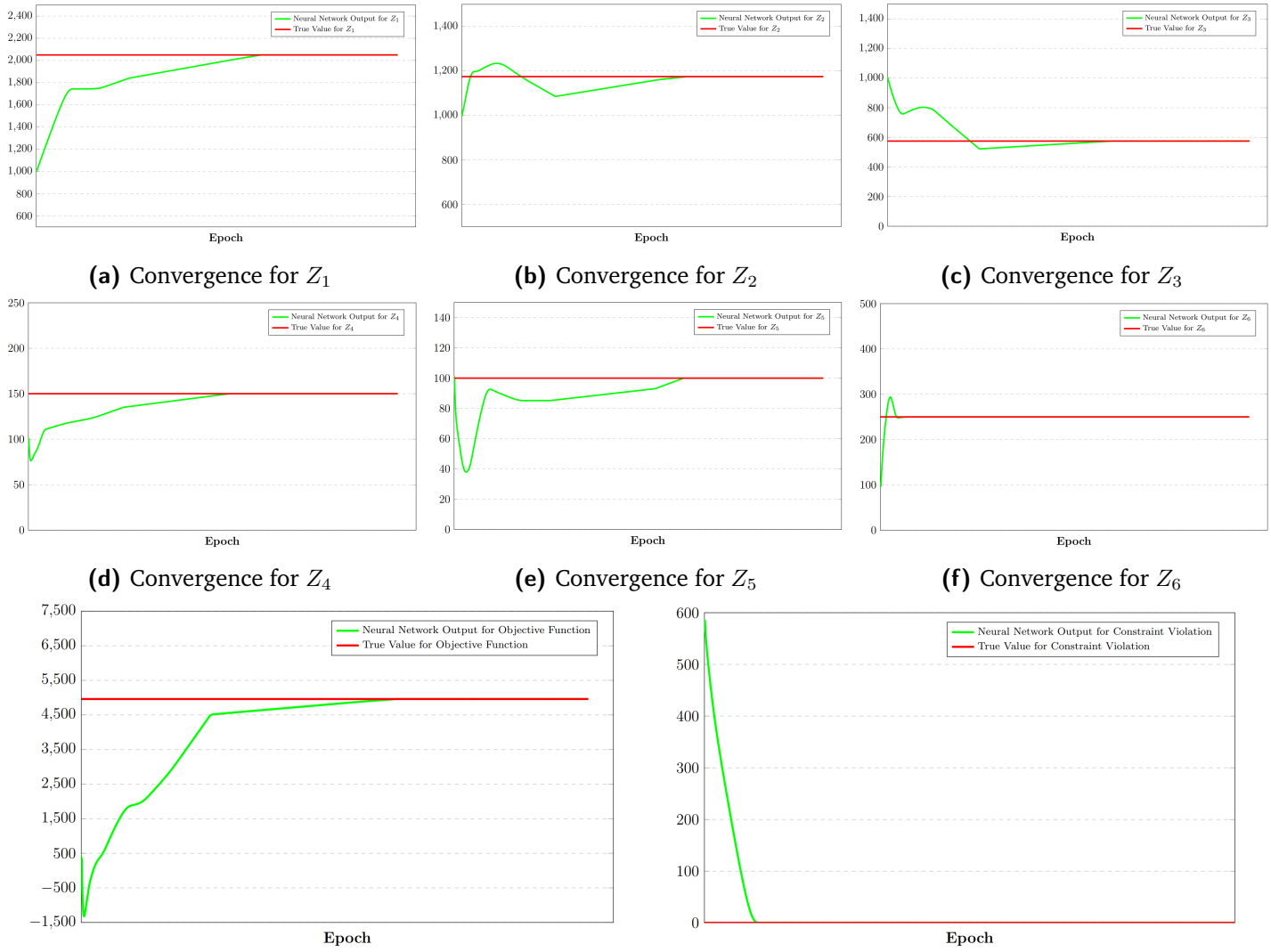
Initially, the first three optimization variables are set to 1000, and the rest are set to 100 before starting the training process. This initialization process depends on the user, but the initial variables should be chosen in a logical way. The training result of this sample Dakota problem is shown in Figure 11 and Table 5. We clearly see that artificial neural network converges to true variables and objective value without violating any constraints.

$$\mathbf{c} = \begin{bmatrix} -2.0 \\ -4.0 \\ -5.2 \\ 60.0 \\ 40.0 \\ 15.0 \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 8.0 & 6.0 & 1.0 \\ 0 & -1 & 0 & 4.0 & 2.0 & 1.5 \\ 0 & 0 & -1 & 2.0 & 1.5 & 0.5 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 3500 \\ 1500 \\ 1000 \\ 0 \\ 0 \\ 0 \\ 150 \\ 100 \\ 250 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \end{bmatrix}$$

As we can see that neural network initially focuses on improving the constraint violation in Figure 11h because of the given large penalty term, and in a fast way, it converges to zero violation in early iterations compared to convergence iteration of decision variables, then it starts to improve objective function value by keeping constraint violation at the same level.

Decision Variable	**True Optimal Decision Variables**	**Neural Network Initial Decision Variables**	**Neural Network Optimal Decision Variables**
Z_1	2050	1000	2050.189
Z_2	1175	1000	1175.082
Z_3	575	1000	575.039
Z_4	150	100	150.022
Z_5	100	100	100.017
Z_6	250	100	250.010

Table 5: Neural Network and True Solution Comparison of Dakota Optimization Problem



(a) Convergence for Z_1 **(b)** Convergence for Z_2 **(c)** Convergence for Z_3

(d) Convergence for Z_4 **(e)** Convergence for Z_5 **(f)** Convergence for Z_6

(g) Convergence for Objective Function **(h)** Convergence for Constraint Violation

Figure 11: Neural Network Convergence of Dakota Optimization Problem

3.3.3 Prescriptive Model Mapped by Feature Data

In this section, we present the details of our integrated predictive and prescriptive learning algorithm. In practice, a decision-maker can use a sample average approximation approach using historical observations to solve an optimization problem with uncertain parameters in the absence of feature data and when true distribution is un-

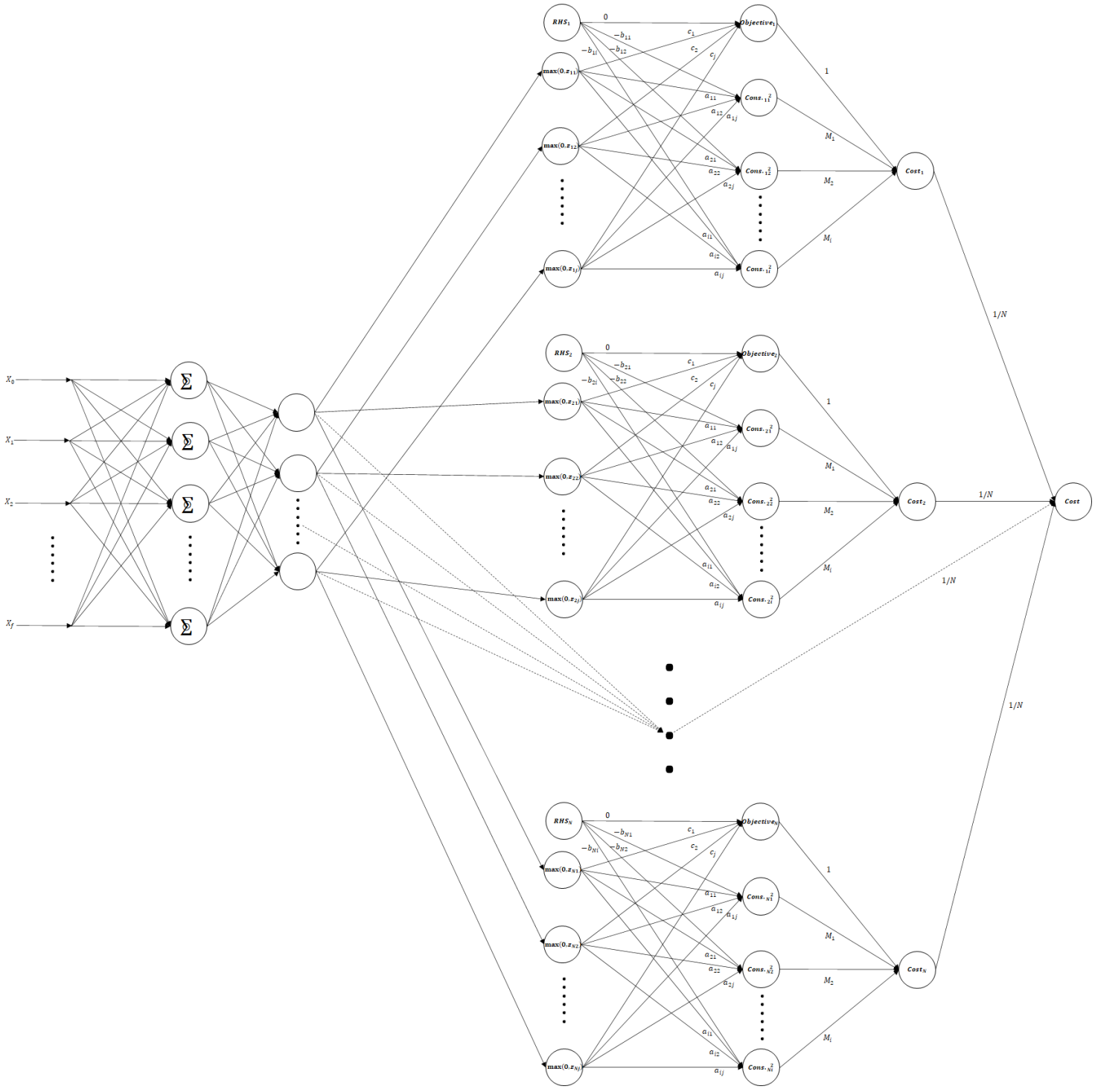
known to the decision-maker. With the rapid growth of interest in machine learning, all parties started to collect the data available to them in a structured or unstructured manner to support their decision systems. One of the approaches that tackle the uncertainty in the optimization model in the availability of feature data is the feature-based optimization proposed by Ban and Rudin [2].

This approach rewrites the "here-and-now" decisions as a linear combination of the feature data. We showed the performance and power of this approach in Chapter 2, and this method was the most competitive one to our proposed integrated predictive and prescriptive optimization method. Furthermore, Oroojlooyjadid et al. [17] and Zhang and Gao [25] extended this method in Deep Learning Neural Network to capture non-linearity between feature data and decision variables more excellently or effectively.

These two neural network methods, which are an extension of feature-based optimization proposed by Ban and Rudin [2], focus on newsvendor problems with the only objective function and no mention of constraints. At this point, we generalize these two extended methods to tackle constraints too.

We form the standard formulation of an optimization problem. We replace this constrained optimization problem with an unconstrained problem by converting equality constraints into quadratic loss functions and adding them to objective with a sufficiently large penalty term. In Figure 12, we present the integrated predictive and prescriptive learning structure, first artificial neural network, which produces

the "here-and-now" decision variables using proper linear or non-linear activation functions with an optimized number of hidden layers, then feed the decision to the prescriptive stage to evaluate the quality for each scenario.



(a) Predictive Network **(b)** Prescriptive Network

Figure 12: Neural Network Structure for Integrated Predictive and Prescriptive Learning

3.3.4 Preprocess and Setup of Prescriptive Neural Network Model

As we see in Figure 12, we have two different sets of weights. The first stage weights help predict "here-and-now" (Z^f) decisions and forward them to respective scenarios in the second stage. The second stage fixes the provided "here-and-now" decisions and optimizes the scenarios via recourse decisions (Z^s) by minimizing the cost. In this part, we describe how we obtain the initial weights for predictive and prescriptive neural networks.

The key idea for a good weight initialization is to train the first neural network separately and optimize the second neural network using an optimization solver. We begin our initialization process by setting our epoch (iteration) number to zero in Step 2. In Step 3, we optimize prescriptive part to find true "here-and-now" decision variables (Z^{f*}) using problem formulation from Equation 3.4a to Equation 3.4c with respect to true parameters. Then, we train an accurate predictive neural network function that maps feature data to true "here-and-now" decisions using the formulation given in Step 4 by minimizing loss function in Equation 3.5. Weights of this optimized neural network will be set to initial weights of the predictive stage in the integrated predictive and prescriptive neural network in Step 7. Next, we calculate the predicted "here-and-now" decisions using this trained neural network in Step 5.

$$\min_{Z^f, Z^s} \sum_{n \in N} c^f Z_n^f + c^s Z_n^s \tag{3.4a}$$

$$\text{s.t.} \quad aZ_n^f \geq b \qquad \forall n \in N \tag{3.4b}$$

$$t_n Z_n^f + w_n Z_n^s \geq r_n \qquad \forall n \in N \tag{3.4c}$$

$$Z_n^f - \hat{Z}_n^f = 0 \qquad \forall n \in N \tag{3.4d}$$

We fix these predicted "here-and-now" decisions in constraint 3.4d to find the predicted recourse decision variables of the prescription stage in Step 6, and optimized predicted recourse decisions will be set to initial weights of the prescriptive stage in the integrated predictive and prescriptive neural network in Step 8. We ensure that the integrated predictive and prescriptive neural network stars an approximate solution and with a zero constraint violation by applying this weight initialization process. Our experiments show that setting initial predictive and prescriptive weights with this initialization process yields a better convergence than randomizing the predictive and prescriptive weights.

$$\min_{\beta} \frac{1}{|N|} \sum_{n \in N} L(\upsilon(\tilde{X}, \beta), Z^{f*}) \tag{3.5}$$

3.3.5 Proposed Decomposition Method

The proposed integrated neural network methodology in Figure 12 jointly trains predictive and prescriptive stages. The predictive stage has only one neural network

that processes input features to create "here-and-now" decision variables, and this neural network forwards the decision variables to the prescriptive stage. However, the prescriptive neural network stage is a combination of multiple neural networks, and each neural network represents a different scenario for the optimization part. Each neural network consists of one input, one hidden layer, and one output layer. While the input layer takes care of non-negativity for eligible variables, the hidden layer controls the objective function and constraint violations. The output layer combines objective function and constraint violations as a sum to minimize. The number of neurons in the input layer has a number of total decision variables. The number of neurons in the hidden layer equals the number of constraints plus one term, which calculates the objective function of the optimization problem. Although hidden and output layer weights come from optimization parameters and are set as non-trainable, increasing the scenario size adds more complexity to this methodology since each added scenario will bring more decision variables and constraints.

As seen in Figure 12, the predictive neural network is shared by all scenarios in the prescriptive neural network. This property gives us the opportunity to solve all scenario problems in parallel and to decompose the prescriptive task. The general idea behind the decomposition algorithm is to solve each scenario in the prescriptive stage and calculate the scenario cost by a given predictive neural network in Step 12, then find the derivatives for the given scenario with respect to predictive and prescriptive neural network weights in Step 13 and 14, respectively. While, prescriptive neural network weights are updated one by one for each scenario after calculating

respective derivative in Step 15, predictive neural network weights are updated via summation of derivatives and a learning rate η after computing all derivatives for each scenario in Step 17. This process continues until reaching max epoch number within a loop from Step 10 to 18. The details and steps of the decomposition algorithm are given in Algorithm 2. Initialization steps prepare the integrated neural network for best performance as described in Section 3.3.4. With a proper weight initialization, the neural network accelerates the convergence and prescribes a better solution set.

Algorithm 2: Decomposition Algorithm for Integrated Learning

Initialization

Set epoch e=0

$Z_n^{f,e*} = \underset{Z^f, Z^s}{\operatorname{argmin}}\, c^f Z_n^f + c_n^s Z_n^s$ *s.t. Constraints 3.4a-3.4c* $\forall n \in N$

$\hat{\beta}_e = \underset{\beta}{\operatorname{argmin}} \frac{1}{|N|} \sum_{n \in N} L(\upsilon(\tilde{X}_n, \beta), Z_n^{f,e*})$

$\hat{Z}_n^{f,e} = \upsilon(\tilde{X}_n, \hat{\beta}_e)$

$\hat{Z}_n^{s,e} = \underset{Z^f, Z^s}{\operatorname{argmin}}\, c^f Z_n^f + c_n^s Z_n^s$ *s.t. Constraints 3.4a-3.4d* $\forall n \in N$

Initialize predictive weights with $\hat{\beta}_e$

Initialize prescriptive weights with $\hat{Z}_n^{s,e}$

Decomposition

for *epoch* $e \in E$*;*

  for *scenario* $n \in N$*;*

    $Cost_n = \phi(\tilde{X}_n, \beta_{e-1}, \hat{Z}_n^{s,e-1})$

    $\Delta_n^{\beta_e} = \frac{\partial Cost_n}{\partial \beta}$

    $\Delta_n^{Z_n^{s,e}} = \frac{\partial Cost_n}{\partial Z_n^s}$

    $\hat{Z}_n^{s,e} = \hat{Z}_n^{s,e-1} - \eta \Delta_n^{Z_n^{s,e}}$

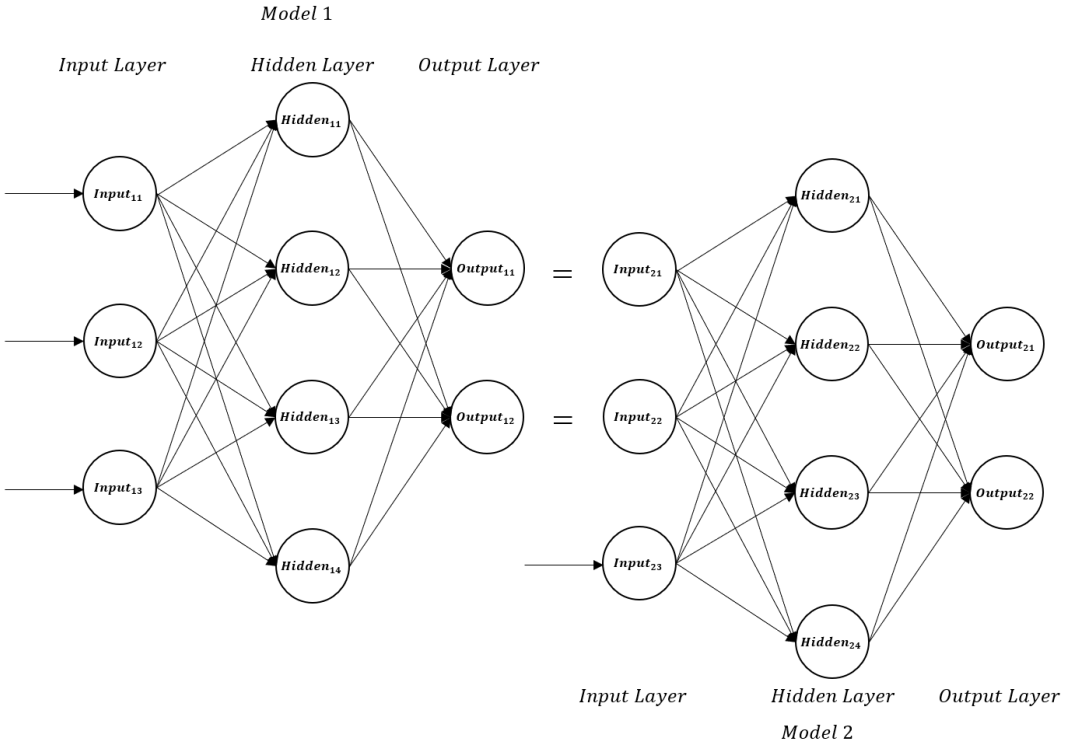
  end for

  $\beta_e = \beta_{e-1} - \eta \sum_{n \in N} \Delta_n^{\beta_e}$

end for

3.4 Integrated Learning of Nested Neural Networks

The central concept in this part is to build a method to better capture the non-linearity between uncertain parameters of the optimization problem and feature data. So far, we have assumed that the objective function and constraints are well-known to decision-makers. However, there are cases where the elements of the optimization problem are not known to decision-makers, or there might be cases where some of the feature data of the predictive algorithm should be also predicted from a different predictive algorithm. At this point, we offer a nested neural network structure where a primary predictive neural network generates inputs for the second network. We discuss four different integration types between two predictive models or between a predictive and a black-box prescriptive model considered as a second predictive model. As we see in Figure 13, there are two predictive model representation, some inputs for the second neural network comes from the output of the first neural network.



(a) Predictive Task **(b)** Prescriptive Task

Figure 13: Sequential Neural Networks

3.4.1 NN1 Model Separate Learning

We start with the simplest and most common case of network integration. Train both networks separately with respect to their own loss functions, and save them for future use. Training criteria can be the same or different, but it is shown in Equations 3.6 and 3.7. Whenever a new observation occurs, first predict using the first trained model, then feed output as input to the second network to make the final prescription.

$$\min_{\psi} \frac{1}{|N|} \sum_{n \in N} L(\hat{Y} = \psi(X), Y) \tag{3.6}$$

$$\min_{\upsilon} \frac{1}{|N|} \sum_{n \in N} L(\hat{Z} = \upsilon(Y), Z) \tag{3.7}$$

3.4.2 NN2 Model Direct Learning

The second proposed network integration is based on direct training from end to end. Since the output of the first network and input of the second network is the same, we can establish an indirect function between the input of the first network and the output of the second network. The foundation of this methodology is the same as what Ban and Rudin [2] propose in the optimization environment, and what we build in Section 3.3. The training function of this end-to-end training methodology is given in Equation 3.8.

$$\min_{\phi} \frac{1}{|N|} \sum_{n \in N} L(\hat{Z} = \phi(X), Z) \tag{3.8}$$

3.4.3 NN3 Model "First Learn then Integrate" Learning

In this framework, first, we train the second network and save the optimal weights. Then, we add this network to the end portion of the first network by fixing the opti-

mal weights of the trained network so that we can train the whole network together as shown in Figure 14. The second network acts as a custom loss function for the first one. We evaluate the prediction quality of the first network based on the effect in the second one. Moreover, we can add a second loss function to the end based on the output of the connection layer with a penalty coefficient where $0 \leq \lambda \leq 1$. Since this hidden layer represents the first network's output, the additional loss ensures the quality of the prediction model in the external data set and controls the generalization error. The training function of this end-to-end training methodology is given in Equation 3.10.

$$\upsilon^* = \underset{\upsilon}{\operatorname{argmin}} \frac{1}{|N|} \sum_{n \in N} L(\hat{Z} = \upsilon(Y), Z) \tag{3.9}$$

$$\min_{\psi,\phi} \frac{1}{|N|} \sum_{n \in N} (\lambda) L(\hat{Z} = \phi(X, \upsilon^*), Z) + (1 - \lambda) L(\hat{Y} = \psi(X), Y) \tag{3.10}$$

3.4.4 NN4 Model Weighted Joint Loss Learning

The last methodology is an extension of the second proposed method. We train the integrated network from end to end as performed in NN2 Model, but additionally, we control the output of the first network by a loss function at the end of the integrated network. We train both networks at the same time based on a weighted loss combination of both networks as seen in Figure 14. This framework is a neural network approximation version of the bilevel optimization methodology that we

propose in Chapter 2. The penalty coefficient where $0 \leq \lambda \leq 1$ ensures the balance between predictive and prescriptive loss. When $\lambda = 1$, this method converges to NN2 Model-Direct Learning defined in Equation 3.8, and when $\lambda = 0$, we ignore the prescription part and train directly predictive task solely, and that leads us to Equation 3.6.

$$\min_{\phi,\psi} \frac{1}{|N|} \sum_{n \in N} (\lambda) L(\hat{Z} = \phi(X), Z) + (1 - \lambda) L(\hat{Y} = \psi(X), Y) \tag{3.11}$$

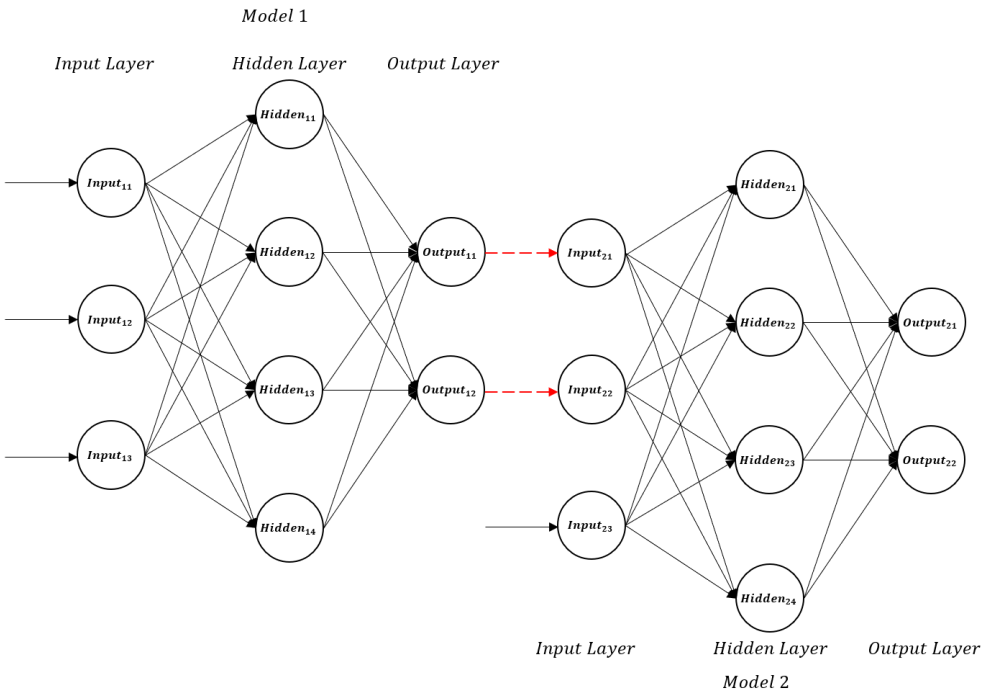


Figure 14: Nested Neural Networks

3.5 Experiment for Integrated Predictive and Prescriptive Learning

This section provides an experimental study to show the proposed integrated predictive and prescriptive learning model results.

3.5.1 Prescriptive Model

The optimization problem we use as a showcase is the two-stage shipment planning problem which we introduce in Chapter 2 provided by Bertsimas and Kallus [7]. We create a network between $d_w = 3$ warehouses and $d_l = 6$ locations. Since the uncertain demand of locations affects warehouses' production, the response variable of the predictive neural network will be the amount of production in warehouses to satisfy uncertain demand in locations. For the two-stage shipment problem, the predictive task in Figure 12a establishes a connection between side information and "here-and-now" decision variables. The prescriptive task in Figure 12b evaluates the quality of a forwarded decision set by minimizing over cost for each iteration. Integrated predictive and prescriptive optimization model is given in Equation from 3.12a to 3.12e, where $\psi(\tilde{X}_n, \beta)_i$ represents the output of predictive neural network for each warehouse.

$$\min_{\hat{Z},T,S,\beta} \quad \frac{1}{|N|}\sum_{n\in N}\left[\sum_{i\in I} p_1\hat{Z}_{n,i} + \sum_{i\in I} p_2 T_{n,i} + \sum_{i\in I}\sum_{j\in J} c_{n,i,j}S_{n,i,j}\right] \tag{3.12a}$$

$$\text{s.t.} \quad \sum_{i\in I} S_{n,i,j} \geq \tilde{Y}_{n,j} \qquad \forall j \in J, \forall n \in N \tag{3.12b}$$

$$\sum_{j\in J} S_{n,i,j} \leq \hat{Z}_{n,i} + T_{n,i} \qquad \forall i \in I, \forall n \in N \tag{3.12c}$$

$$\hat{Z}_{n,i} = \psi(\tilde{X}_n, \beta)_i \qquad \forall i \in I, \forall n \in N \tag{3.12d}$$

$$\hat{Z}_{n,i}, \quad T_{n,i}, \quad S_{n,i,j} \geq 0 \qquad \forall i \in I, \forall j \in J, \forall n \in N \tag{3.12e}$$

Decision Variables for Two-Stage Shipment Problem

β Weight sets of predictive neural network

$\hat{Z}_{n,i}$ Amount of predicted production done in advance at warehouse i in scenario n

$T_{n,i}$ Amount of production done last minute at warehouse i in scenario n

$S_{n,i,j}$ Amount of shipment from warehouse i to location j in scenario n

Parameters for Two-Stage Shipment Problem

p_1 Cost of production done in advance at warehouse

p_2 Cost of production done last minute at warehouses

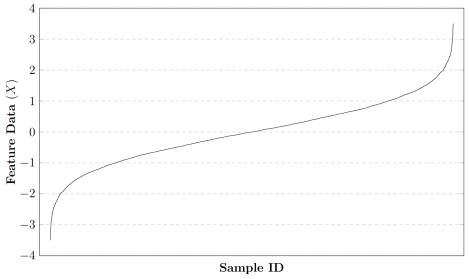
$c_{n,i,j}$ Cost of shipment from warehouse i to location j in scenario n

$\tilde{Y}_{n,j}$ Amount of observed demand at location j in scenario n

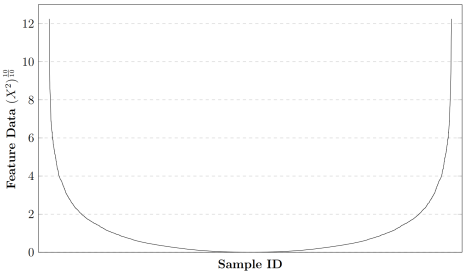
3.5.2 Data Generation

The main idea of building an integrated predictive and prescriptive algorithm in neural network environment is to better capture the relationship between feature data and uncertain parameters of prescriptive model. In order to show performance

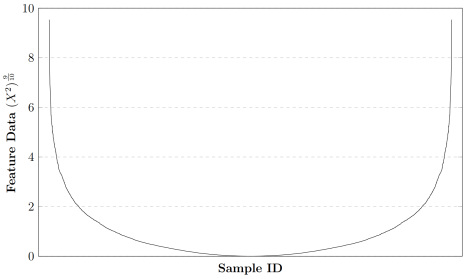
of neural network based optimization, we generate the feature variables in a non-linear way. First, we create feature variables based on a $d_x = 3$ dimensional multivariate normal distribution with size of $n = 2000$ observations, $\tilde{X} \in \mathbb{R}^{n \times d_x}$, i.e., $\tilde{X} \sim N(\mu, \Sigma)$, where $\mu = [0, 0, 0]$ and $\Sigma = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]$. In Figure 15a, sorted version of one of the feature variables is plotted and we name it as Data Set 0. We use square value of this distribution as source of our other non-linear data sets. We generate other non-linear feature variables based on $(\tilde{X}^2)^{\frac{Power}{10}}$ where $Power$ is an integer value from 2 to 10, and we name these distributions as Non-Linear Feature Data Set 1 to Non-Linear Feature Data Set 9. When $Power$ goes from 10 to 2, the degree of non-linearity decreases, and distribution curve becomes more flat as seen in Figures from 15b to 15j. Then, we choose the true parameters of our predictive side of integrated neural network model as $\beta \in \mathbb{R}^{d_x \times d_l}$ matrix for slopes. Next, we calculated demand according to the model $\tilde{Y} = \tilde{X}\beta + \varepsilon$, where ε is independently generated noise term and follows normal distribution, i.e., $\varepsilon \sim N(0, \sigma)$. As for cost parameters, we randomly simulate its matrix as from warehouse i to location j based on uniform distribution $c_{n,i,j} \sim U(0, 30)$ for each scenario. Out of created $n = 2000$ observations, we randomly choose train, validation, and test sets with size of $70, 15, 15$, respectively. This splitting process is repeated by 20 times, and we report all results based on the average cost of these replications.



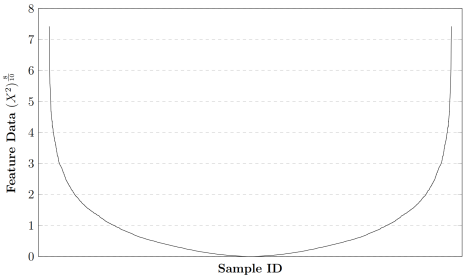
(a) Non-Linear Feature Data Set 0



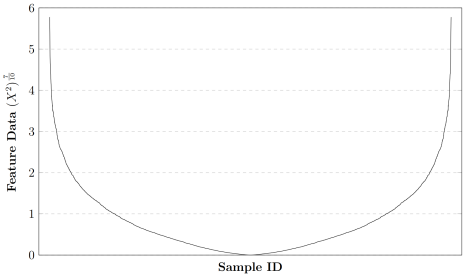
(b) Non-Linear Feature Data Set 1



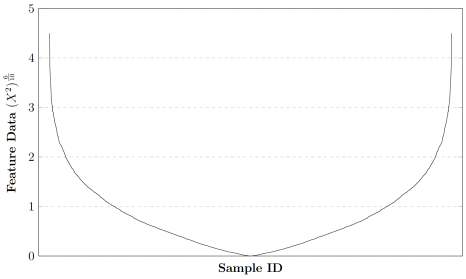
(c) Non-Linear Feature Data Set 2



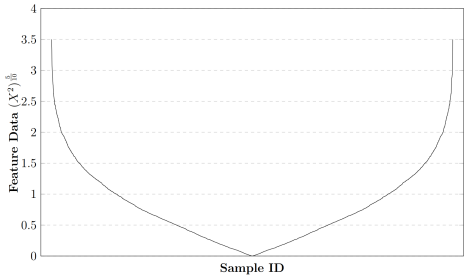
(d) Non-Linear Feature Data Set 3



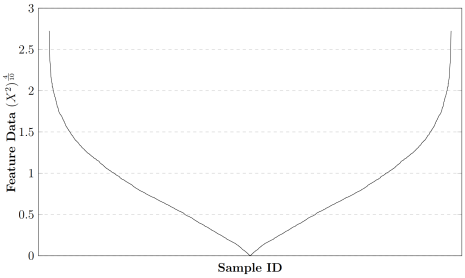
(e) Non-Linear Feature Data Set 4



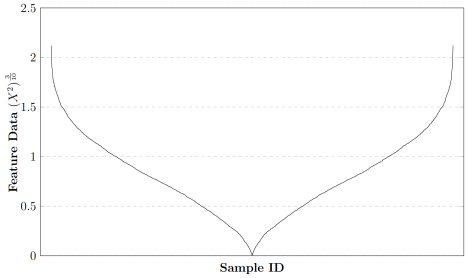
(f) Non-Linear Feature Data Set 5



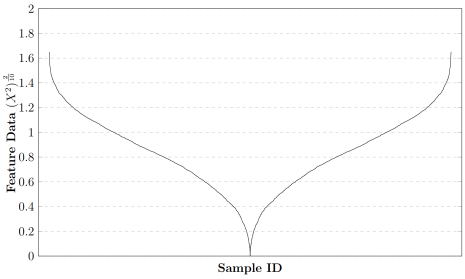
(g) Non-Linear Feature Data Set 6



(h) Non-Linear Feature Data Set 7



(i) Non-Linear Feature Data Set 8



(j) Non-Linear Feature Data Set 9

Figure 15: Distribution of Different Non-Linear Feature Data Sets

3.5.3 Weight Initialization

Initialization of weights is one of the crucial steps when designing an artificial neural network model. The simplest way to initialize network weights is to assign random numbers, but there have been some heuristic developments for better initialization. As we discuss the steps of weight initialization in Section 3.3.4, using Gurobi optimization solver via python API [13], we optimize the true decision variables subject to true parameters of the optimization model in the first place.

In the next step, we train a neural network in Tensorflow [1] to find the best representation of "here-and-now" decisions via feature data. The weights of this trained neural network initialize the weights of our predictive side of the integrated method. In the last step, we fix the predicted "here-and-now" decisions and optimize the shipment model to find optimal recourse variables. Optimal recourse variables based on predicted "here-and-now" decisions become the weights of the prescriptive side of the integrated method. We start training integrated predictive and prescriptive neural networks with an approximation solution and a zero constraint violation in this initialization process. We report the effect of different initialization for a given sample problem model in Figure 16. The different weight initialization leads us to the different prescriptions and cost values.

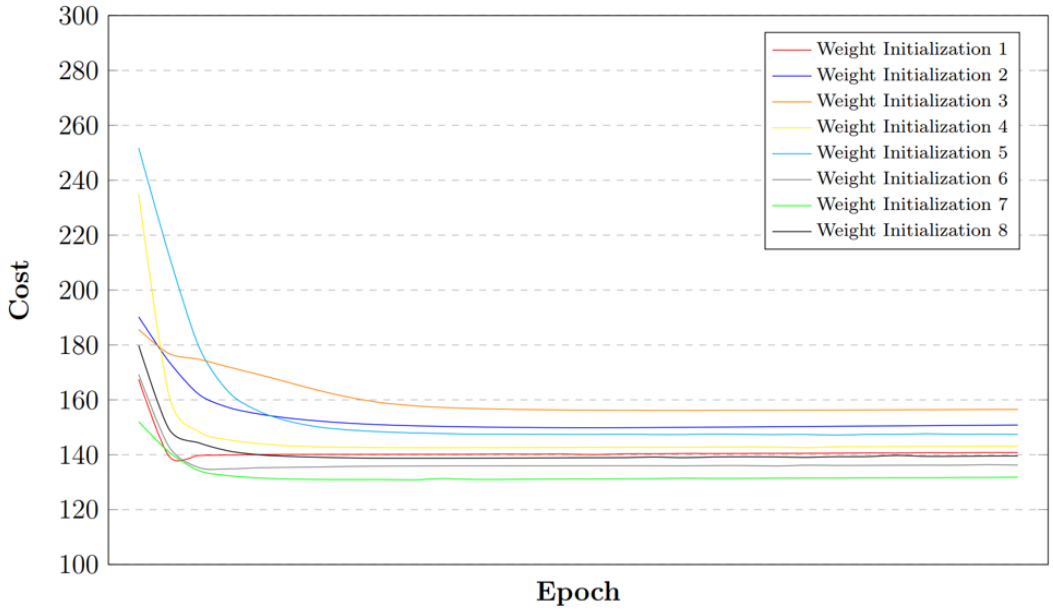


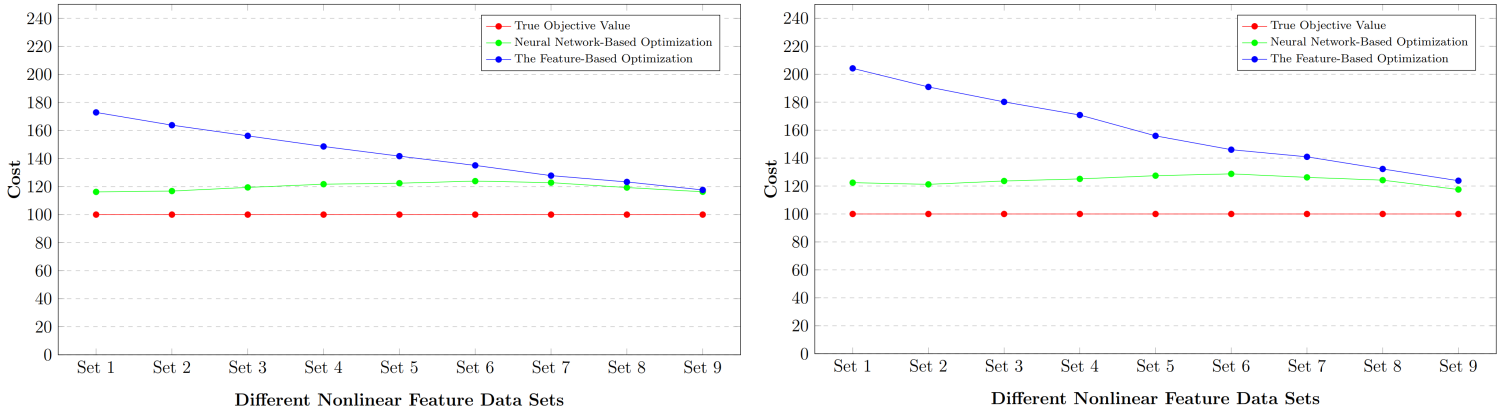
Figure 16: Effect of Different Weight Initialization in Convergence of Cost

3.5.4 Computational Results

This section discusses the performance of the proposed integrated predictive and prescriptive neural network model. All results for this experiment are obtained from Gurobi [13] python API and Tensorflow [1]. The idea of building an integrated neural network model is to better capture the non-linearity between feature data and prescription task. The Feature-Based Optimization developed by Ban and Rudin [2] proposes the linear connection between feature data and prescription task. Therefore, we basically compare the results of the feature-based optimization and neural network-based optimization to prove our claim. We show the performance of integrated neural network-based optimization and the feature-based optimization under different data sets.

As we introduce data sets in Section 3.5.2, we create different non-linear feature data sets at different non-linearity levels. We expect to see a closing gap in performance between the feature-based and neural network-based optimization when the non-linearity level goes from high to low. In Figure 17, train and test performance of two-stage shipment problem are given based on different non-linear data sets between feature values and demand. Data sets are labeled from 1 to 9, where Set 1 represents the highly no-linear data set while Set 9 represents a slightly non-linear data set. Section 3.5.2 contains detailed and more information about how we create non-linear data sets at different levels.

For better graphical purposes, we scale objective function values based on true objective values of an average of replications for each data set. The actual values for objective values of both optimization models and true objective values are reported in Table 7, and scaled results are reported in Table 6 for each data sets. As we see in Figure 17, when we go from highly to slightly non-linear, the gap of objective values decreases since feature-based optimization starts to capture better. However, neural network-based optimization has consistent objective values across all data sets. Reported results for neural network-based optimization are optimized over validation data sets; instead of choosing the best train performance, we choose the best train and validation performance to achieve the best performance in out of sample data sets.



(a) Train Data Set

(b) Test Data Set

Figure 17: Comparison of Feature-Based and Neural Network-Based Optimization for Shipment Problem

Data Set ID	Train Data True Objective Value	Neural Network Based Train Data Objective Value	Feature-Based Train Data Objective Value	Test Data True Objective Value	Neural Network Based Test Data Objective Value	Feature-Based Test Data Objective Value
Set 1	100.0	116.2	172.9	100.0	122.4	204.2
Set 2	100.0	116.8	163.8	100.0	121.2	190.9
Set 3	100.0	119.4	156.2	100.0	123.6	180.2
Set 4	100.0	121.7	148.6	100.0	125.1	170.8
Set 5	100.0	122.4	141.7	100.0	127.4	155.9
Set 6	100.0	123.9	135.1	100.0	128.7	146.0
Set 7	100.0	122.8	127.8	100.0	126.2	140.9
Set 8	100.0	119.3	123.3	100.0	124.2	132.2
Set 9	100.0	116.3	117.6	100.0	117.5	123.8

Table 6: Comparison of Neural Network and Feature Based Optimization with Scaled Objective Values

Data Set ID	Train Data True Objective Value	Neural Network Based Train Data Objective Value	Feature-Based Train Data Objective Value	Test Data True Objective Value	Neural Network Based Test Data Objective Value	Feature-Based Test Data Objective Value
Set 1	897.4	1044.2	1549.2	903.2	1107.7	1836.3
Set 2	838.3	979.5	1371.3	839.9	1016.8	1592.2
Set 3	793.6	947.6	1237.9	794.5	982.6	1420.4
Set 4	761.1	926.1	1130.4	752.5	943.1	1278.7
Set 5	744.8	911.9	1055.3	725.5	925.8	1132.5
Set 6	732.5	907.8	989.3	722.6	930.4	1054.9
Set 7	731.2	898.5	934.8	684.2	862.9	963.0
Set 8	731.9	873.0	901.9	735.4	912.2	971.9
Set 9	772.8	899.1	908.9	745.6	875.1	921.1

Table 7: Comparison of Neural Network and Feature Based Optimization with Unscaled Objective Values

3.6 Experiment for Integrated Learning of Nested Neural Networks

This section provides an experimental study to show the results of the proposed integrated learning of nested neural network models. We want to show the implementability of the proposed integrated neural network model when objective function or constraints are not known to decision-makers.

3.6.1 Prescriptive Model

The optimization problem we use as a showcase is the two-stage shipment planning problem that we introduce in Chapter 2. The response variable will be the amount of production in warehouses to satisfy uncertain demand in locations. For two-stage shipment problem, predictive task in Figure 13a will establish a connection between side information and demand. Prescriptive task in Figure 13b will establish a connection between demand and observed production amount.

We evaluate all of the proposed nested neural network models under the same conditions. The predictive task will not have a hidden layer, so there will be only input and output layer. As for the prescriptive task, there will be one hidden layer with 200 neurons. Initially, these parameters are optimized for NN1 Model-Separate Learning, then applied for integrated neural networks in order to satisfy homogeneity between proposed models. Of course, each model may give a better result for a different set of parameters. However, we intend to see the performance of the mod-

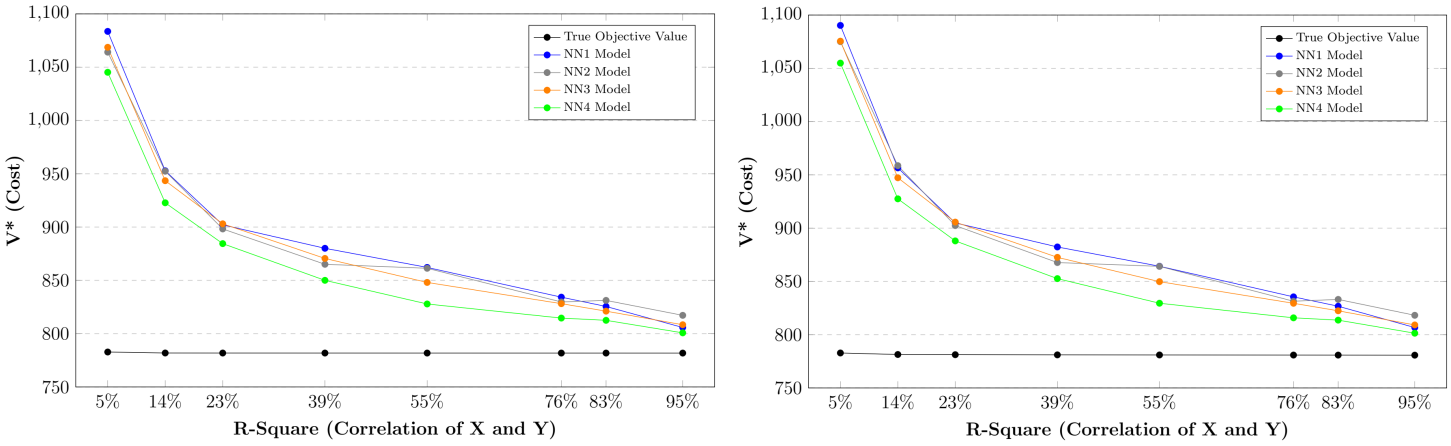
els under the same conditions instead of optimizing the parameters of each model individually.

3.6.2 Data Generation

The data generation process for the two-stage shipment problem is presented in Section 2.4.2 with a train, validation, and test sizes of 70k, 15k, and 15k, respectively. We follow the same steps to generate data presented in Section 2.4.2, but with more sizes. We select train, validation, and test sets out of created $n = 2000000$ observations.

3.6.3 Computational Results

This section discusses the performance of proposed nested neural networks under different training sizes and different levels of observed $\tilde{X}$, $\tilde{Y}$ correlations. All results for these experiments are obtained from Gurobi python API [13] and Tensorflow [1]. We compared the results of different types of neural network integration.



(a) Train Data Set

(b) Test Data Set

Figure 18: Comparison of Different Neural Network Methods for Shipment Problem

3.7 Conclusion

In real-world problems, parameter distribution of the optimization problems is known for all feature data points, and there is no need to handle uncertainty to find an exact solution. However, most of the parameters either are unknown or follow an irregular form. In the presence of feature data, most of the historical proposed methods propose to predict uncertain parameters, then prescribe the decisions. We proposed an exact and scalable algorithm in Chapter 2 integrating the predictive and prescriptive tasks to prescribe decisions efficiently, but the proposed method has limitations to linear predictive models. This chapter considers a new approach to increase the quality of decisions in the presence of feature data related to uncertain parameters in a non-linear way. The proposed algorithm works in a neural network environment to better capture the non-linearity between feature data and optimal decisions, and it is scalable. The algorithm integrates predictive and prescriptive

tasks under one neural network instead of building a predictive task and separately optimizing the prescription task. Predictive task maps feature data to decisions, and prescriptive task evaluates the quality of these decisions, then prescriptive task sends feedback to predictive task to improve the quality of decisions. We also proposed different neural network integration models where there is a need for nested predictive models or elements of an optimization problem is not known or well defined, such as constraints and objective function. In the final part of this chapter, we perform our computational studies to demonstrate the power of the proposed model using synthetic on two-stage shipment problem.

CHAPTER 4 CONCLUSION AND FUTURE RESEARCH

As we discuss in introduction sections of each chapter, our motivation mainly derives from prescription ignorance of training criteria of predictive algorithms despite wide data availability and excess accessibility in computational power. This dissertation introduces different data-driven frameworks for integrating the predictive and prescriptive tasks to prescribe optimal decisions by effectively leveraging feature data and closes the gap between predictive and prescriptive tasks. We develop two main frameworks. While the first proposed framework in Chapter 2 jointly optimizes the predictive and prescriptive tasks, the second framework proposed in Chapter 3 jointly learns the predictive and prescriptive tasks.

4.1 Summary of Contributions

In Chapter 2, we build a framework called "Integrated Predictive and Prescriptive Optimization" (IPPO) for a predictive regression model caring for the characteristics of the prescriptive model and providing the best decisions. While IPPO is seeking the most accurate predictive regression model, the desired predictive regression model will tackle uncertainty by delivering the best actions in the prescription stage. In the classical approach, the predictive and prescriptive tasks are often treated independently and often in a sequence, i.e., "first predict, then optimize". However, our proposed integrated framework joining the predictive and prescriptive combines three steps: a predictive regression model generating responses to the optimization model, the optimization model prescribing decisions based on the generated responses, and

another optimization model evaluating the quality of prescribed decisions subject to actual parameters. We model this three-step framework as a bilevel optimization problem because of involved nested optimization models. While the lower-level problem makes decisions with the help of predicted parameters, the upper-level problem jointly determines the regression parameters. It evaluates these decisions subject to actual parameters. Since the integrated method directly focuses on finding the best prescriptions for "in-sample" data, we also propose different techniques to achieve the same level of quality for "out-of-sample" data. Conversion of bilevel problem to single level increases the number of constraints and the variables; we propose a decomposition algorithm based on progressive hedging algorithm for scalability. However, the conversion of the bilevel optimization problem to a single-level problem leverages Karush–Kuhn–Tucker (KKT) conditions that require convexity; this method is limited o linear predictive and prescriptive tasks.

In Chapter 3, we develop an artificial neural network-based framework to integrate predictive and prescriptive tasks because of the linearity limitation of the proposed algorithm in Chapter 2. The two artificial neural network trains simultaneously to prescribe best decisions via feature data. The first neural network serves as a predictive task and predicts decision variables based on feature data and forward decisions for quality check. The second neural network uses predicted decision variables, the output of the first neural network. It optimizes the prescription stage based on minimization of cost, which is the sum of the objective function and quadratic term of penalized constraints. The second neural network is a stack of independent

neural networks presenting each scenario for prescriptive problems. Hence, the increase in the number of scenarios explodes the number of neurons in the prescriptive neural network. We also develop a decomposition method for scalability. The proposed decomposition algorithm solves each scenario in a prescriptive neural network by using the same predictive neural network weights, then updates these weights by aggregating derivatives from each scenario with respect to scenario cost. To obtain the best performance, we also propose a weight initialization process for predictive and prescriptive neural networks.

Our results for the example problems validate the performance of our proposed integrated predictive and prescriptive optimization and training frameworks. With customarily generated synthetic data sets, proposed methods surpass all of the "first predict, then optimize" approaches and recently developed approximate integration methods for both "in-sample" and "out of sample" data sets. We also observe how the proposed generalization error controlling approach improves results in "out of sample" data sets. Customarily generated synthetic data pairs at different levels of correlation and non-linearity graphically show us how different methods converge to each other.

4.2 Future Research

Despite our significant contributions in Chapter 2 and 3, the exploration of prescription-based predictive algorithms is still open to development, and there still exist several

opportunities. The proposed two integrated algorithms in Chapter 2 and 3 completes each other in terms of linearity and non-linearity. Although the first proposed method solves the integration of predictive and prescriptive tasks in an exact way, it is limited to the linearity of both tasks. We demonstrate how to overcome this linearity limitation in the second proposed algorithm by leveraging the artificial neural network models. However, we approximate decision variables by directly mapping them from feature data using a predictive neural network in the second proposed method. In other words, we predict decision variables rather than predicting uncertain parameters of the optimization problem. Predicting decision variables approximates the solution of the model. Therefore, this brings us to the intersection of our two contributions in this dissertation: to learn the bilevel optimization framework in a neural network environment to predict uncertain parameters of the optimization problem in a non-linear manner and prescribe decisions based on parameter prediction. However, this proposition may require more effort to tune the integrated neural network since the bilevel formulation produces four constraint types: upper-level constraints, lover level constraints, duality constraints, and complementarity constraints. Although all these constraints increase the complexity of the integrated learning framework, there is still an opportunity to decompose the problem since all scenario problems share the same predictive neural network. Another proposition to solve integrated predictive and prescriptive tasks in a bilevel optimization framework to capture non-linearity is to apply a separate artificial neural network. The separate neural network algorithm trains from feature data to uncertain parameters. The out-

put of the last hidden layer can be used as feature data in the bilevel optimization as linear regression. However, this proposition still lacks full integration of predictive and prescriptive tasks since the separate artificial neural network handles the non-linearity outside of the integration as a preprocess to integrated optimization.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] G.-Y. Ban and C. Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67:90–108, 2019.

[3] J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications (Nonconvex Optimization and Its Applications)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

[5] Y. Bengio. Using a financial training criterion rather than a prediction criterion. *International journal of neural systems*, 8 4:433–43, 1997.

[6] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[7] D. Bertsimas and N. Kallus. From predictive to prescriptive analytics. *Management Science*, 0(0):null, 2020.

[8] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Publishing Company, Incorporated, 2nd edition, 2011.

[9] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3-4):197–206, 1955.

[10] M. de la Maza, D. Yuret, et al. A critique of the standard neural network application to financial time series analysis.

[11] P. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5484–5494. Curran Associates, Inc., 2017.

[12] A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize". *ArXiv*, abs/1710.08005, 2017.

[13] L. Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[14] Y. Kao, B. V. Roy, and X. Yan. Directed regression. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 889–897. Curran Associates, Inc., 2009.

[15] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[16] Y. Lv, T. Hu, G. Wang, and Z. Wan. A neural network approach for solving nonlinear bilevel programming problem. *Computers & Mathematics with Applications*, 55(12):2823 – 2829, 2008.

[17] A. Oroojlooyjadid, L. V. Snyder, and M. Takác. Applying deep learning to the newsvendor problem. *ArXiv*, abs/1607.02177, 2016.

[18] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–

147, 1991.

[19] A. Shapiro. Monte carlo sampling methods. In *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 353 – 425. Elsevier, 2003.

[20] A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. In V. Jeyakumar and A. Rubinov, editors, *Continuous Optimization: Current Trends and Modern Applications*, pages 111–146. Springer US, Boston, MA, 2005.

[21] H. Stackelberg. The theory of the market economy. *Oxford University Press, New York, Oxford*, 1952.

[22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[23] T. Tulabandhula and C. Rudin. Machine learning with operational costs. *J. Mach. Learn. Res.*, 14:1989–2028, 2011.

[24] J. Wang. A deterministic annealing neural network for convex programming. *Neural Networks*, 7(4):629 – 641, 1994.

[25] Y. Zhang and J. Gao. Assessing the performance of deep learning algorithms for newsvendor problem. In *ICONIP*, 2017.

ABSTRACT

INTEGRATED OPTIMIZATION AND LEARNING METHODS OF PREDICTIVE AND PRESCRIPTIVE ANALYTICS

by

MEHMET KOLCU

August 2021

Advisor : Dr. Alper Murat

Major : Industrial Engineering

Degree : Doctor of Philosophy

A typical decision problem optimizes one or more objectives subject to a set of constraints on its decision variables. Most real-world decision problems contain uncertain parameters. The exponential growth of data availability, ease of accessibility in computational power, and more efficient optimization techniques have paved the way for machine learning tools to effectively predict these uncertain parameters. Traditional machine learning models measure the quality of predictions based on the closeness between true and predicted values and ignore decision problems involving uncertain parameters for which predicted values are treated as the true values. Standard approaches passing point estimates of machine learning models into decision problems as replacement of uncertain parameters lose the connection between predictive and prescriptive tasks. Recently developed methods to strengthen the bond between predictive and prescriptive tasks still rely on either "first predict, then optimize" strategy or use approximation techniques in integrating predictive and prescriptive tasks.

We develop an integrated framework for performing predictive and prescriptive analytics concurrently to realize the best prescriptive performance under uncertainty. This framework is applicable to all prescriptive tasks involving uncertainty. Further, it is scalable to handle integrated predictive and prescriptive tasks with reasonable computational effort and enables users to apply decomposition algorithms for large-scale problems. The framework also accommodates prediction tasks ranging from simple regression to more complex black-box neural network models.

The integrated optimization framework is composed of two integration approaches. The first approach integrates regression-based prediction and mathematical programming-based prescription tasks as a bilevel program. While the lower-level problem prescribes decisions based on the predicted outcome for a specific observation, the upper-level evaluates the quality of decisions with respect to true values. The upper-level problem can be considered as a prescriptive error, and the goal is to minimize this prescriptive error. In order to achieve the same performance in external data sets (test) compared to internal data sets (train), we offer different approaches to control the "prescription generalization error" associated with out-of-sample observation. We develop a decomposition algorithm for large-scale problems by leveraging a progressive hedging algorithm to solve the resulting bilevel formulation. The second approach integrates the learning of neural network-based prediction and optimization tasks as a nested neural network. While the predictive neural network promotes decisions based on predicted outcomes, the prescriptive neural network evaluates the quality of predicted decisions with respect to true values. We also propose a

weight initialization process for nested neural networks and build a decomposition algorithm for large-scale problems.

Our results for the example problems validate the performance of our proposed integrated predictive and prescriptive optimization and training frameworks. With customarily generated synthetic data sets, proposed methods surpass all of the "first predict, then optimize" approaches and recently developed approximate integration methods for both "in-sample" and "out of sample" data sets. We also observe how the proposed generalization error controlling approach improves results in "out of sample" data sets. Customarily generated synthetic data pairs at different levels of correlation and non-linearity graphically show us how different methods converge to each other.

AUTOBIOGRAPHICAL STATEMENT

Mehmet Kolcu is a registered Ph.D. student of the Department of Industrial and Systems Engineering, Wayne State University, Detroit, Michigan. He earned his Bachelor of Science degree in Industrial Engineering from Gazi University, Ankara, Turkey, in 2010. He received his Master of Engineering degree from Lehigh University, Bethlehem, Pennsylvania, in 2015. His research background and interests lie in the areas of machine learning, decision making under uncertainty, and integrated predictive and prescriptive analytics. He joined Norfolk Southern Corporation, one of the nation's premier transportation companies serving every major container port in the eastern United States, in May 2021. He started his NS career by supporting and enhancing the optimal blocking model.