Integrating prediction and portfolio optimization

by

Andrew David Sidney Butler

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Mechanical and Industrial Engineering
University of Toronto

Integrating prediction and portfolio optimization

Andrew David Sidney Butler
Doctor of Philosophy

Department of Mechanical and Industrial Engineering
University of Toronto
2022

Abstract

The fundamental objective of portfolio optimization is the determination of next period's optimal asset allocation under conditions of uncertainty. In most settings the portfolio optimization inputs, such as risk and return, are unobservable at decision time and must be forecasted using observable feature data. Traditionally, predictive forecast models are optimized independently of their use in the downstream portfolio optimization. While a perfect prediction model would lead to optimal decision-making, in reality, all prediction models do make some error. As such, an inefficiency exists in the traditional 'predict then optimize' paradigm: prediction models are optimized with unique prediction-based objectives and are therefore unaware as to how those predictions will be used in the downstream portfolio optimization. This thesis aims to address this shortcoming and presents an alternative integrated prediction and optimization (IPO) framework. In contrast to a traditional estimation approach, the IPO framework optimizes prediction models in order to minimize the final downstream decision objective. We first consider the integration of linear regression prediction models in a mean-variance optimization setting and derive closed-form analytical expressions for the optimal IPO coefficients. We then consider IPO estimation for more general prediction and portfolio optimization problems. We make use of recent advances in neural network architecture and provide a first-order gradient descent framework for optimizing prediction model parameters to minimize downstream portfolio objectives. Experimental results demonstrate that IPO prediction models can result in lower out-of-sample decision error and improved economic performance. The IPO parameter estimation process, however, can be computationally demanding, particularly when the number of decision variables is large. We address these computational challenges in the medium to large scale limit and present an alternative differentiable neural network architecture for constrained quadratic programming. We conclude by demonstrating that the IPO frameworks developed and studied herein are generalizable to alternative forms of prediction and optimization modelling. We present a gradient boosting algorithm for integrating general additive prediction models with downstream convex quadratic cone decision optimization. Experimental results comparing with state-of-the-art IPO methods demonstrate improved reduction in out-of-sample decision regret.

To my son, Callum.

Acknowledgement

I would first like to extend my sincerest appreciation to my supervisor and mentor Professor Roy Kwon. I am incredibly thankful for all of the advice, guidance and support that you have given me. Thank you for providing me with an unbelievable opportunity to grow, to think and to hone my research skills in such an exiting and relevant field. Your knowledge and passion for both research and education is inspiring. I would also like to thank my committee members: Professor Scott Sanner and Professor Joseph C. Paradi for their support and invaluable insights over the years.

I would like to thank my colleagues at ReSolve Asset Management for their encouragement in my academic pursuits. I would like to extend a special appreciation to Adam Butler, who 10 years ago took a chance on a kid who enjoyed building mathematical models but had zero experience in financial markets. I have learned, and continue to learn so much from you and I am thankful for the mentorship, friendship and kindness that you have given me over the years. I would also like to thank my colleague and friend Maciej Zawadzki. I deeply value our countless conversations on machine learning and optimization; many of which have inspired the contents of this thesis. I have grown so much as a person because of your generosity, mentorship and unfaltering belief in me.

I would like to thank my parents, David and Janice Butler. I would not be where I am today without your ever present support and encouragement. You are a true inspiration and I hope that one day I can provide the same foundation of love and nurturing to Callum. Most importantly I would like to thank my wife, Gina Hamilton, for her unwavering love and support throughout this journey. These past 4 years have been challenging with many long nights and missed opportunities to be together. None of this would be possible without you. You have sacrificed so much in pursuit of my dream and have done so with such patience and grace - for that I am eternally grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fundamental role of a portfolio manager is to determine the next period's optimal asset allocation under conditions of uncertainty. Determining what constitutes as an 'optimal' portfolio is primarily determined by the objectives of the investor along various dimensions of preference, such as *expected* risk and *expected* return. It is important to highlight that in most cases the portfolio characteristics, such as risk and return, are not known with certainty at decision time. The portfolio asset allocation problem is therefore a perfect example of a decision-making problem under uncertainty; the portfolio manager must make an asset allocation decision with noisy or incomplete information.

This thesis takes a quantitative approach to portfolio management in which optimal asset allocation decisions are determined by solving a mathematical optimization program. For example, Markowitz [96], a pioneer of Modern Portfolio Theory, proposed that investors' preferences for return and risk can be characterized by a linear tradeoff between the expected return and variance of the portfolio. Therefore, given the estimates of return and risk, an optimal portfolio can then be determined by solving an optimization problem, known as mean-variance optimization (MVO).

MVO and many other portfolio optimization problems fall into a special class of mathematical programming known as *convex* optimization and can be solved to optimality with 100% reliability. However, the portfolio optimization program requires as input estimates of asset return and risk, which are themselves not known with certainty and thus susceptible to estimation error. In today's data rich world, much of the recent effort in quantitative portfolio management focuses on the design and refinement of predictive forecasting models.

Conventional wisdom suggests that the quality of the portfolio is determined, in large part, by the accuracy of the forecasted inputs. Traditionally, prediction models for estimating portfolio risk

and return are optimized with unique *prediction-based* objectives and constraints, and are therefore unaware of how those predictions will ultimately be used in the context of their final *decision-based* optimization. Indeed, a traditional 'predict, then optimize' framework would first fit the prediction model (for example by maximum likelihood or least-squares) and then 'plug-in' those estimates to the corresponding decision-based optimization program. While it is true that a perfect prediction model would lead to optimal decision-making, in reality, all prediction models do make some error, and thus an inefficiency exists in the 'predict, then optimize' paradigm.

In this thesis, we explore the fundamental question: 'do optimal predictions result in optimal portfolio decision-making?' Here, 'optimal predictions' are forecasts generated by a prediction model that is optimized for forecast accuracy; independent of its downstream impact on portfolio decision-making. In this thesis we present and study an alternative 'integrated prediction and optimization' (IPO) framework. The key distinction is that under the integrated setting, the prediction model parameters are estimated in order to minimize the downstream portfolio optimization objective. As such, prediction models are parameterized such that they induce 'optimal' decisions, not 'optimal' predictions.

With the widespread adoption of machine-learning and data science in operations research, there has been a growing body of literature on data-driven optimization and the relative merits of decoupled versus integrated predictive decision-making (see for example [4, 45, 13, 47, 48, 67, 95, 94, 121]). The preliminary findings of the aforementioned work advocate for an IPO approach. Indeed, IPO models typically exhibit lower model complexity and improved out-of-sample performance in comparison to the traditional 'predict, then optimize' approach. This thesis makes several important theoretical and practical contributions to the integrated prediction and optimization body of literature. We focus on portfolio optimization problems with an emphasis on systematic and data-driven asset allocation frameworks. Our overarching objective is to study the integrated prediction and portfolio optimization problems and to contribute to both the theoretical and practical advancement of the field. To that end, we provide an overview of each chapter and summarize the primary contributions therein.

1.1 Thesis overview

1.1.1 Chapter 2: Introduction to portfolio optimization

In Chapter 2 we provide a brief review of convex optimization theory. We present the Lagrangian and Lagrange dual problems and provide the well-known Karush-Kuhn-Tucker (KKT) optimality conditions; both of which are fundamental to the content of subsequent chapters. We then discuss portfolio optimization as an asset allocation decision policy. We present the Markowitz mean-variance optimization framework, discuss its natural extensions, and present alternative risk-based portfolio optimization frameworks. Where necessary we provide convex reformulations and provide the KKT optimality conditions particular to each portfolio optimization problem.

1.1.2 Chapter 3: Integrated prediction and optimization

We begin Chapter 3 with a review of existing literature on integrated prediction and optimization. We then describe the traditional 'predict, then optimize' approach and present a general form of the integrated prediction and optimization (IPO) alternative. We cast the IPO problem as a bi-level stochastic optimization program. We describe the properties of the IPO program and discuss challenges in solving the more difficult integrated problem. We conclude with a motivating example that highlights the benefit of the IPO approach in comparison to the decoupled alternative.

1.1.3 Chapter 4: Differentiable optimization layers

In Chapter 4 we introduce the concept of a differentiable optimization layer (DOL). In short, DOLs, embed iterative optimization algorithms as specialized layers in a larger neural network system and are fundamental to the innovations described in Chapters 5, 6 and 7. We discuss how to reformulate the IPO program as an end-to-end trainable neural network and optimize prediction model parameters by backpropagation. We present two approaches for differentiating the solution to a convex optimization program, namely *unrolled differentiation* and *implicit differentiation*. We conclude by providing the DOL architecture for several portfolio optimization programs.

1.1.4 Chapter 5: Integrating prediction in mean-variance optimization

In Chapter 5 we present the IPO framework for integrating linear regression prediction models in a mean-variance portfolio optimization setting. The theoretical contribution of this chapter is the development of closed-form analytical expressions for the optimal IPO regression coefficients under

particular portfolio constraint assumptions. Specifically, we demonstrate that for the case where the decision program is either unconstrained or contains only linear equality constraints then the integrated program can be recast as an unconstrained quadratic program. We provide the necessary conditions for convexity and provide analytical solutions for the optimal IPO regression coefficients. Moreover, we provide conditions for which the regression coefficient is an unbiased estimator and derive the analytical expression for the variance. We further demonstrate that the IPO coefficients explicitly minimize the tracking error to the unconstrained ex-post optimal MVO portfolio and provide the equivalent minimum tracking error optimization formulation.

Under more general constraint assumptions, we make use of recent advances in neural network architecture for efficient optimization of batch quadratic programs and provide a first-order gradient based method for performing the integrated optimization. To our knowledge, this is the first rigorous study of integrating prediction in a mean-variance portfolio optimization setting. Out-of-sample results demonstrate that the IPO model can provide lower realized cost and improved economic performance in comparison to a traditional 'predict then optimize' approach. The contributions and findings from this research are presented in Butler and Kwon [28]; currently under review by the Journal of Quantitative Finance.

1.1.5 Chapter 6: Integrated covariance estimation for risk-based portfolio optimization

In Chapter 6 we present the IPO framework for covariance estimation with downstream risk-based portfolio optimizations. We consider three risk-based portfolio optimizations programs: minimum-variance, maximum-diversification and equal-risk-contribution portfolios, and provide the corresponding IPO formulations. Experimental results focus on the IPO framework for minimum-variance portfolios. Out-of-sample results demonstrate that the integrated approach can provide consistent and significantly lower realized portfolio variance in comparison to the traditional 'predict, then optimize' alternative. The contributions from this research are published as Butler and Kwon [27] in the Journal of Risk.

1.1.6 Chapter 7: Differentiable quadratic programming layers: an ADMM approach

In Chapter 7 we address the computational challenges for medium to large scale IPO problems and propose an alternative differentiable neural network architecture for batch constrained quadratic pro-

grams. Our differentiable quadratic programming layer is built on top of the alternating direction method of multipliers (ADMM) algorithm and applies a custom fixed-point implicit differentiation algorithm that is computationally efficient. We perform several numerical simulations and compare the computational efficiency and performance accuracy of our ADMM layer with state-of-the-art implementations. We demonstrate that for medium-scale problems, our ADMM layer implementation is approximately an order of magnitude faster and provides solutions that are equally as optimal. We conclude with an application of the ADMM layer to medium scale portfolio optimization problems.

This chapter provides an efficient and practical algorithm for solving the IPO optimization problem and is relevant to asset managers who construct portfolios from a relatively large pool of assets. The contributions from this research are presented in Butler and Kwon [26]; currently under review by the Journal of Computational Optimization and Applications. Our differentiable ADMM layer and all algorithmic implementations are made available as an open-source R package, available here https://github.com/butl3ra/lqp.

1.1.7 Chapter 8: Gradient boosting for convex cone predict and optimize problems

In Chapter 8 we demonstrate that the IPO frameworks developed and studied herein are readily generalizable to a much larger class of prediction model and decision optimization problems. Specifically, we present *dboost*, a gradient boosting algorithm for training prediction model ensembles to minimize downstream decision regret. We consider 'additive' prediction models and focus on the case where each function unit is a regression tree. The *dboost* framework supports convex quadratic cone programming and gradient boosting is performed by implicit differentiation of a custom fixed-point mapping. Experimental results demonstrate that training prediction models with *dboost* can further reduce decision regret in comparison to existing state-of-the-art solutions. The contributions from this research are presented in Butler and Kwon [26]; currently under review by Operations Research Letters. The *dboost* framework is made available as an open-source R package, available here: https://github.com/butl3ra/dboost.

1.2 Primary Contributions

- **Chapter 5: Integrating prediction in mean-variance portfolio optimization:**
  - We present the IPO framework for integrating linear regression prediction models in a

mean-variance optimization (MVO) setting. Empirical analysis, on both synthetic and historical price data, highlight the benefits of the integrated approach.

  - We demonstrate that under special constraint sets the IPO estimation problem reduces to a convex quadratic program. We derive a globally optimal analytical solution for the IPO regression coefficients that is computationally efficient to compute.
  - We provide the analytical expressions for the bias and the variance of the optimal IPO regression coefficients and present conditions for which the IPO regression coefficients are an unbiased estimator.
  - We demonstrate that the IPO regression coefficients explicitly minimize the tracking error to the unconstrained ex-post optimal MVO portfolio and provide the equivalent minimum tracking error optimization formulation.

- **Chapter 6: Integrated covariance estimation for risk-based portfolio optimization:**
  - We present the IPO framework for covariance estimation with downstream risk-based portfolio optimizations. To our knowledge, this is the first empirical study that integrates covariance prediction modelling in a portfolio optimization setting.

- **Chapter 7: Differentiable quadratic programming layers: an ADMM approach:**
  - We present a novel and efficient ADMM differentiable neural network layer for box constrained quadratic programs. For medium to large scale IPO problems, the ADMM layer is shown to be approximately an order of magnitude faster than the state-of-the-art implementations.
  - We derive a fixed-point implicit differentiation algorithm that is customized to the ADMM iterations. The fixed-point implicit differentiation is more efficient than KKT implicit differentiation, and under certain conditions is preferred to unrolled differentiation.

- **Chapter 8: Gradient boosting for convex cone predict and optimize problems:**
  - We present *dboost*, a gradient boosting algorithm for training 'additive' prediction model ensembles to minimize downstream decision regret. Previous work considers gradient boosting for integrated prediction and optimization problems but only considers a small subset of optimization problems with linear inequality constraints. In contrast, *dboost* is a general-purpose implementation that is capable of supporting any optimization problem

that can be cast as a convex quadratic cone program; and thus supports linear, quadratic and second-order cone programming with general convex cone constraints.

– We present a fixed-point implicit differentiation algorithm for computing the gradient of the decision loss with respect to all of the cone program variables.

1.3 Notation

We denote the real coordinate space of dimension n as $\mathbb{R}^n$. Vectors are denoted as bold lowercase letters. Matrices are denoted as bold uppercase letters. The superscript $\mathbf{z}^T$ and $\mathbf{V}^T$ indicates the transpose of vector $\mathbf{z}$ and matrix $\mathbf{V}$, respectively. We denote a discrete dataset of size m as $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$. The i^{th} element of dataset, D, is given by the pair $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x}$ and $\mathbf{y}^{(i)} \in \mathbb{R}^{d_y}$. The subscript, j, is used to reference a specific element j of a vector or a matrix. For example, the j^{th} element of the vector $\mathbf{x}^{(i)}$ is given as ${\mathbf{x}_{\text{j}}}^{(i)}$, whereas $\mathbf{V}_{jk}$ denotes the j^{th} row and k^{th} column of matrix $\mathbf{V}$. Element-wise multiplication between two vectors or matrices is denoted by the Hadamard operator: '$\odot$'. We denote 'diag' as the diagonal operator, which takes as input a vector and creates a diagonal matrix with the vector values along the main diagonal. Lastly, we denote the expectation operator as '$\mathbb{E}$' whereas '$\mathbb{E}_D$' denotes the sample expectation with respect to a discrete dataset D.

Chapter 2

Introduction to portfolio optimization

We begin this chapter with a brief review of convex optimization theory. We then discuss portfolio optimization as an asset allocation decision policy and present the Markowitz [96] mean-variance optimization (MVO) framework. We discuss natural extensions of the MVO program; namely the maximum return, maximum Sharpe ratio, and minimum variance portfolio. We conclude by presenting risk-based portfolio optimization frameworks that focus on maximizing portfolio diversification, namely: maximum diversification and equal-risk-contribution portfolios.

As stated in Chapter 1, many real-world portfolio optimization problems can be cast as convex optimization programs. In fact, all portfolio optimization problems considered in this thesis are convex optimization problems. It is therefore important to layout the foundational principles of convex optimization theory. In particular we define the system of equations that describe optimality of a convex program, namely the Karush-Kuhn-Tucker (KKT) conditions. Later, in Chapter 4, we demonstrate that the KKT system of equations defines a fixed-point from which we can compute the partial derivative(s) of the solution to the convex optimization program with respect to the program input variables; ultimately culminating in the design of a first-order algorithm for locally solving the integrated prediction and optimization problem.

2.1 Review of convex optimization

Many problems in engineering, machine learning, and finance require solving convex optimization programs. Convex optimization is a special class of mathematical optimization characterized as the minimization of a convex function over a convex set. Here, we provide some useful definitions and state (without proof) some important properties of convex optimization. We refer the reader to Boyd and Vandenberghe [20] for a comprehensive overview of convex optimization theory and its applications.

Definition 1. *A set C is **convex** if for any $\mathbf{x}_1, \mathbf{x}_2 \in C$ and $0 \leq \alpha \leq 1$, we have:*

$$\alpha\,\mathbf{x}_1 + (1-\alpha)\,\mathbf{x}_2 \in C. \tag{2.1}$$

Definition 2. *A function $f\colon \mathbb{R}^{d_x} \to \mathbb{R}$ is **convex** if the domain of f is a convex set and if for all $\mathbf{x}_1, \mathbf{x}_2$ in the domain of f, and $0 \leq \alpha \leq 1$ we have:*

$$f(\alpha\,\mathbf{x}_1 + (1-\alpha)\,\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2). \tag{2.2}$$

A **convex optimization problem** is a mathematical optimization program of the form:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & c(\mathbf{z}) \\ \text{subject to} \quad & g(\mathbf{z}) \leq 0 \\ & h(\mathbf{z}) = 0 \end{aligned} \tag{2.3}$$

We call $\mathbf{z} \in \mathbb{R}^{d_z}$ the **decision variable** and the function $c\colon \mathbb{R}^{d_z} \to \mathbb{R}$ the **objective function** or **cost function**. The functions $g\colon \mathbb{R}^{d_z} \to \mathbb{R}^{d_{\text{iq}}}$ and $h\colon \mathbb{R}^{d_z} \to \mathbb{R}^{d_{\text{eq}}}$ denote the vector-valued inequality and equality constraint functions, respectively. When the function h is affine and the functions c and g are convex, then Program (2.3) is the minimization of a convex function over a convex set and is therefore a convex optimization program.

If there are no constraints then we say that the problem is **unconstrained**. Alternatively, we define the **feasible region**, $\mathbb{S}$, as the set of points that satisfy the inequality and equality constraints:

$$\mathbb{S} = \{\mathbf{z} \in \mathbb{R}^{d_z} | g(\mathbf{z}) \leq 0, h(\mathbf{z}) = 0\}. \tag{2.4}$$

We define the optimal cost of Program (2.3) as:

$$p^* = \inf\{c(\mathbf{z}) | \, \mathbf{z} \in \mathbb{S}\}, \tag{2.5}$$

with corresponding optimal solution:

$$\mathbf{z}^* = \{\mathbf{z} \in \mathbb{S} | c(\mathbf{z}^*) = p^*\}. \tag{2.6}$$

An important property of convex optimization is that any locally optimal solution of Program (2.3) is globally optimal and we say that $\mathbf{z}^*$ is the optimal solution. This result is derived from **duality theory** and is fundamental to the design and convergence properties of algorithms for solving convex optimization problems. We conclude this section by defining the Lagrange dual and stating the well-known Karush-Kuhn-Tucker (KKT) conditions that are necessary and sufficient for optimality of Program (2.3). Again, we refer to Boyd and Vandenberghe [20] for a complete overview of duality theory and optimality conditions in convex programming.

Definition 3. *The **Lagrangian**, $\mathcal{L}\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_{iq}} \times \mathbb{R}^{d_{eq}} \to \mathbb{R}$ of Program* (2.3) *is defined as:*

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\eta}) = c(\mathbf{z}) + \boldsymbol{\lambda}^T g(\mathbf{z}) + \boldsymbol{\eta}^T h(\mathbf{z}), \tag{2.7}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{d_{iq}}$ and $\boldsymbol{\eta} \in \mathbb{R}^{d_{eq}}$ are the dual variables of the inequality and equality constraints, respectively.

Observe that the Lagrangian is a weighted sum of the cost and constraint functions and is therefore a relaxation of Program (2.3).

Definition 4. *The **Lagrange dual function** $q\colon \mathbb{R}^{d_{iq}} \times \mathbb{R}^{d_{eq}} \to \mathbb{R}$ of Program* (2.3) *is defined as the minimum value of the Lagrangian over all feasible* $\mathbf{z}$ *and is defined as:*

$$q(\boldsymbol{\lambda}, \boldsymbol{\eta}) = \inf_{\mathbf{z} \in \mathbb{S}} \mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\eta}). \tag{2.8}$$

and provides a lower bound on the optimal cost p^*: $q(\boldsymbol{\lambda}, \boldsymbol{\eta}) \leq p^*$.

Definition 5. *The **Lagrange dual problem** of Program* (2.3) *is defined as:*

$$\begin{aligned} \underset{\boldsymbol{\lambda}, \boldsymbol{\eta}}{\text{maximize}} \quad & q(\boldsymbol{\lambda}, \boldsymbol{\eta}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq 0 \end{aligned} \tag{2.9}$$

with optimal dual cost d^* *and optimal dual solution* $(\boldsymbol{\lambda}^*, \boldsymbol{\eta}^*)$.

The Lagrange dual problem (2.9), therefore seeks to determine the largest possible lower bound and is fundamental to proving convergence and optimality in convex programming. The difference $p^* - d^*$ is known as the **optimality gap** and measures the distance between the optimal value of the original (primal) problem and the largest lower bound attainable by the dual problem. For any mathematical optimization problem the inequality, $d^* \leq p^*$, holds and is known as **weak duality**. When Program (2.3) is convex then, under the regular constraint qualifications (such as Slater's condition), **strong duality** holds: $p^* = d^*$, and provides a certificate proving that the primal-dual solution, $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*)$, is globally optimal.

These optimality conditions are summarized by the KKT conditions, presented below:

Definition 6. *The Karush-Kuhn-Tucker (KKT) conditions associated with Program* (2.3) *for primal stationarity, primal feasibility, dual feasibility and constraint qualification are necessary and sufficient for optimality and are given by:*

$$\begin{aligned} \nabla_{\mathbf{z}} \mathcal{L}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*) = \nabla_{\mathbf{z}} c(\mathbf{z}^*) + \boldsymbol{\lambda}^{*T} \nabla_{\mathbf{z}} g(\mathbf{z}^*) + \boldsymbol{\eta}^{*T} \nabla_{\mathbf{z}} h(\mathbf{z}^*) = 0 \\ g(\mathbf{z}^*) \leq 0 \\ h(\mathbf{z}^*) = 0 \\ \operatorname{diag}(\boldsymbol{\lambda}^*) g(\mathbf{z}^*) = 0 \\ \boldsymbol{\lambda}^* \geq 0. \end{aligned} \tag{2.10}$$

2.2 Mean-variance portfolio optimization

A portfolio is a collection of assets, such as stocks, bonds, real-estate, and other investable financial instruments, in which we allocate available capital. Indeed, the price of an asset changes over time and the change in price from one period to the next is known as the rate of return. In general, the rate of return, or 'return', of an asset is not known with certainty at the time of investment. Therefore the fundamental objective of portfolio optimization is the determination of next period's 'optimal' asset allocation under conditions of uncertainty.

Determining what constitutes as an 'optimal' investment is primarily determined by the objectives of the investor along two important dimensions: risk and return. At the individual asset level, the risk, or volatility, of an asset quantifies the magnitude of uncertainty in returns around its mean and is measured by the standard deviation of returns. The joint uncertainty of multiple investments

is therefore measured by the covariance of asset returns and quantifies the risk of an asset and its correlation to all other assets in the portfolio.

More formally, in this thesis we consider a universe of d_z assets and denote the matrix of (excess) return observations as $\mathbf{Y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(m)}] \in \mathbb{R}^{m \times d_z}$ for each time period $i \in \{1, 2, ..., m\}$. Let $\mathbf{V}^{(i)} \in \mathbb{R}^{d_z \times d_z}$ denote the time-varying symmetric positive definite covariance matrix of asset returns at time period i. We define the portfolio at time i as $\mathbf{z}^{(i)} \in \mathbb{R}^{d_z}$, where the element, $\mathbf{z}_{\mathrm{j}}{}^{(i)}$, denotes the proportion of total capital invested in the j^{th} asset. Note that if we expect asset j's price to fall in value then it is possible to assign negative capital weight to said asset, $\mathbf{z}_{\mathrm{j}}{}^{(i)} \leq 0$, often referred to as short-selling, or 'shorting' the asset.

The return of a portfolio is therefore calculated as a weighted sum of the returns of the assets, specifically:

$$\mathrm{r_p}{}^{(i)} = \mathbf{z}^{T(i)}\, \mathbf{y}^{(i)}\,. \tag{2.11}$$

Similarly, the portfolio variance of returns is given as:

$$\sigma_{\mathrm{p}}^{2(i)} = \mathbf{z}^{T(i)}\, \mathbf{V}^{(i)}\, \mathbf{z}^{(i)}\,. \tag{2.12}$$

Markowitz [96], a pioneer of Modern Portfolio Theory, proposed that investors' preferences for return and risk are characterized by quadratic utility: mean-variance optimization (MVO). There are many ways to frame the MVO problem, but in general, MVO portfolios allocate capital such that the portfolio return is maximized for a given level of risk. The MVO asset allocation problem can be cast as a convex optimization program:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\mathbf{z}^T\, \mathbf{y} \\ \text{subject to} \quad & \mathbf{z}^T\, \mathbf{V}\, \mathbf{z} \leq \sigma^2. \\ & \mathbf{A}\, \mathbf{z} = \mathbf{b} \\ & \mathbf{G}\, \mathbf{z} \leq \mathbf{h} \end{aligned} \tag{2.13}$$

where σ^2 denotes a target maximum level of portfolio variance and $\mathbf{A} \in \mathbb{R}^{d_{\mathrm{eq}} \times d_{\mathbf{z}}}$, $\mathbf{b} \in \mathbb{R}^{d_{\mathrm{eq}}}$ and $\mathbf{G} \in \mathbb{R}^{d_{\mathrm{iq}} \times d_{\mathbf{z}}}$, $\mathbf{h} \in \mathbb{R}^{d_{\mathrm{iq}}}$ describe the linear equality and inequality constraints, respectively. Note that for ease of notation we have temporarily dropped the time index i.

Furthermore, as outlined below, we can relax the quadratic constraint in Program (2.13) and define the mean-variance cost function as a linear combination of the portfolio return and risk.

Definition 7. *The* **MVO cost function**, $c_{MVO}\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_z} \to \mathbb{R}$ *is defined as:*

$$c(\mathbf{z}, \mathbf{y}) = -\mathbf{z}^T \mathbf{y} + \frac{\delta}{2} \mathbf{z}^T \mathbf{V} \mathbf{z} \tag{2.14}$$

where $\delta \in \mathbb{R}_+$ is a risk-aversion parameter that controls the trade-off between minimizing portfolio variance and maximizing portfolio return.

Definition 8. *The* ***mean-variance portfolio*** *is a convex quadratic optimization program defined as:*

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\mathbf{z}^T \mathbf{y} + \frac{\delta}{2} \mathbf{z}^T \mathbf{V} \mathbf{z} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b} \\ & \mathbf{G}\,\mathbf{z} \leq \mathbf{h}\,. \end{aligned} \tag{2.15}$$

We note that Program (2.15) can be solved using standard off-the-shelf quadratic programming solvers. Following the methodology described in Section 2.1, the Lagrangian of program (2.15) is given by:

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = -\mathbf{z}^T \mathbf{y} + \frac{\delta}{2} \mathbf{z}^T \mathbf{V} \mathbf{z} + \boldsymbol{\lambda}^T (\mathbf{G}\,\mathbf{z} - \mathbf{h}) + \boldsymbol{\eta}^T (\mathbf{A}\,\mathbf{z} - \mathbf{b}), \tag{2.16}$$

where as before, $\boldsymbol{\lambda} \in \mathbb{R}^{d_{\text{iq}}}$ and $\boldsymbol{\eta} \in \mathbb{R}^{d_{\text{eq}}}$ are the dual variables associated with the linear inequality and equality constraints, respectively. The KKT conditions for stationarity, primal feasibility, and complementary slackness are fundamental to the methodology described in Chapter 4 and are given by equations (2.17).

$$\begin{aligned} -\mathbf{y} + \delta\,\mathbf{V}\,\mathbf{z}^* + \mathbf{G}^T \boldsymbol{\lambda}^* + \mathbf{A}^T \boldsymbol{\eta}^* &= 0 \\ \mathbf{G}\,\mathbf{z}^* - \mathbf{h} &\leq 0 \\ \mathbf{A}\,\mathbf{z}^* &= \mathbf{b} \\ \operatorname{diag}(\boldsymbol{\lambda}^*)(\mathbf{G}\,\mathbf{z}^* - \mathbf{h}) &= 0 \\ \boldsymbol{\lambda}^* &\geq 0 \end{aligned} \tag{2.17}$$

Finally, we note that the risk-aversion parameter, δ, controls the investors preference for minimizing risk versus maximizing return. Indeed, for each $\delta \in [0, \infty)$, the associated portfolio $\mathbf{z}^*_\delta$ minimizes Program (2.15), and therefore the set of portfolios $(\mathbf{z}^*_{\delta_1}, \mathbf{z}^*_{\delta_2}, ...)$ define the Pareto efficient frontier. In the following subsection we define three important portfolios on the efficient frontier: the maximum

return portfolio, the maximum Sharpe ratio portfolio, and the minimum variance portfolio.

Definition 9. *The **maximum return portfolio** maximizes the return of the portfolio and is an MVO portfolio with risk-aversion parameter $\delta = 0$:*

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\mathbf{z}^T \mathbf{y} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b} \\ & \mathbf{G}\,\mathbf{z} \leq \mathbf{h}\,. \end{aligned} \tag{2.18}$$

Observe that the maximum return portfolio, defined in Program (2.18), is the minimization of a linear cost function with linear constraints and can therefore be solved by standard linear programming.

As we increase the risk-aversion parameter, we move down the efficient frontier: portfolios have both lower return and lower risk. The portfolio on the efficient frontier that maximizes the ratio of portfolio return per unit risk is called the maximum Sharpe ratio portfolio, defined below.

Definition 10. *The **Sharpe ratio** of a portfolio measures the portfolio (excess) return per unit standard deviation and is defined as:*

$$S_R(\mathbf{z}, \mathbf{y}) = \frac{\mathbf{z}^T \mathbf{y}}{\sqrt{\mathbf{z}^T \mathbf{V}\,\mathbf{z}}} \tag{2.19}$$

Definition 11. *The **maximum Sharpe ratio portfolio** maximizes the Sharpe ratio of a portfolio:*

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\frac{\mathbf{z}^T \mathbf{y}}{\sqrt{\mathbf{z}^T \mathbf{V}\,\mathbf{z}}} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b} \\ & \mathbf{G}\,\mathbf{z} \leq \mathbf{h}\,. \end{aligned} \tag{2.20}$$

Proposition 1. *Let $\mathbf{z}^*$ denote the optimal solution to Program* (2.20)*. The maximum Sharpe ratio portfolio is an MVO portfolio with risk-aversion parameter:*

$$\delta = \frac{\mathbf{z}^{*T} \mathbf{y}}{\mathbf{z}^{*T} \mathbf{V}\,\mathbf{z}^*}.$$

Note that the maximum Sharpe ratio Program (2.20) is non-convex. The critical line algorithm [96, 101], however, allows one to compute the maximum Sharpe ratio portfolio by iterating through 'turning points' on the efficient frontier. Alternatively, following Cornuejols and Tutuncu [37], when

all constraints are homogenous of degree zero then we can recast Program (2.20) as a convex optimization program. Specifically, we define:

$$\mathbb{S}^0 = \{\mathbf{z} \in \mathbb{R}^{d_z}, k \in \mathbb{R} | k > 0, \frac{\mathbf{z}}{k} \in \mathbb{S}\} \cup (\mathbf{0}, 0)$$

as the union of constraints that are homogenous of degree zero and the zero vector. Then Program (2.21) is a convex quadratic program that maximizes the portfolio Sharpe ratio.

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{z}^T \mathbf{V}\, \mathbf{z} \\ \text{subject to} \quad & \mathbf{z}^T \mathbf{y} = 1 \\ & (\mathbf{z}, k) \in \mathbb{S}^0 \end{aligned} \tag{2.21}$$

Increasing the risk-aversion parameter further suggests that investors are more concerned with minimizing portfolio risk rather than maximizing portfolio return. The portfolio on the efficient that minimizes portfolio variance is called the minimum variance portfolio, defined below.

Definition 12. *The **minimum variance portfolio** minimizes the variance of a portfolio:*

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\, \mathbf{z}^T \mathbf{V}\, \mathbf{z} \\ \text{subject to} \quad & \mathbf{A}\, \mathbf{z} = \mathbf{b} \\ & \mathbf{G}\, \mathbf{z} \leq \mathbf{h}\,. \end{aligned} \tag{2.22}$$

2.3 Risk-based portfolio optimization

In general, computing optimal MVO portfolios requires an estimate of expected returns and covariances. Generating a reliable estimator for asset mean returns is particularly difficult as asset returns are characterized as both non-stationary and heterogeneous [51, 72, 104, 114]. Moreover, unlike asset prices, asset returns themselves typically display no significant auto-correlation over any lag [46] and are generally estimated with a high degree of uncertainty [99]. As such, much of the recent literature has moved away from forecasting returns and towards building resilient and diversified portfolios that focus solely on portfolio risk (see for example [34], [93], and [113]).

In acknowledgement that expected returns are often difficult to estimate, Choueifaty and Coignard [34], proposed optimizing portfolios in order to maximize portfolio diversification. They define the 'diversification ratio' as the ratio of weighted average asset volatility to portfolio volatility.

Definition 13. *The **diversification ratio** of a portfolio is defined as:*

$$D_R(\mathbf{z}, \mathbf{y}) = \frac{\mathbf{z}^T \sqrt{\operatorname{diag}(\mathbf{V})}}{\sqrt{\mathbf{z}^T \mathbf{V} \mathbf{z}}} \tag{2.23}$$

As outlined by Choueifaty and Coignard [34], the diversification ratio has many interesting properties. First, it is homogeneous of degree zero, and is therefore invariant under scalar multiplication of $\mathbf{z}$. Secondly, for $\mathbf{V} \succ 0$, the diversification ratio of any long-only portfolio will be strictly greater than or equal to 1. Equality is achieved when the portfolio holds a single asset. Furthermore, for long-only portfolios, the square of the diversification ratio, quantifies the number of independent sources of risk, or 'bets', in the portfolio [35]. Lastly, if expected (excess) returns are proportional to volatility, then maximizing the diversification ratio is equivalent to maximizing the portfolio Sharpe ratio.

Definition 14. *The **maximum diversification portfolio** maximizes the portfolio diversification ratio and is defined as:*

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\frac{\mathbf{z}^T \sqrt{\operatorname{diag}(\mathbf{V})}}{\sqrt{\mathbf{z}^T \mathbf{V} \mathbf{z}}} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b} \\ & \mathbf{G}\,\mathbf{z} \leq \mathbf{h}\,. \end{aligned} \tag{2.24}$$

Note that the maximum diversification portfolio is non-convex, and is typically solved by the critical line algorithm or by following the convex reformulation defined by Program (2.21).

The equal-risk-contribution (ERC) portfolio, first proposed by Maillard et al. [93], provides an alternative to traditional risk-based optimizations such as minimum variance and maximum diversification portfolios. As the name suggests, the ERC portfolio holds all assets such that the risk contribution of each asset is made equal. As a result, the portfolio generally achieves a high degree of ex-ante diversification, with portfolio weights that tend to be less concentrated than its minimum variance or maximum diversification counterparts.

In order to define the ERC portfolio we must first establish some definitions. We begin by noting that portfolio standard deviation, $\sigma_p = \sqrt{\mathbf{z}^T \mathbf{V} \mathbf{z}}$, satisfies Euler's identity:

$$\sigma_p = \sum_{j=1}^{n} \mathbf{z}_j \frac{\partial \sigma_p}{\partial \mathbf{z}_j} = \mathbf{z}^T \frac{d\sigma_p}{d\,\mathbf{z}}. \tag{2.25}$$

Definition 15. *The portfolio* ***marginal risk contribution*** *(MRC) is defined as:*

$$MRC(\mathbf{z}_j) = \frac{1}{\sigma_p}\frac{\partial \sigma_p}{\partial \mathbf{z}_j} = \frac{(\mathbf{V}\,\mathbf{z})_j\,\mathbf{z}_j}{\mathbf{z}^T\,\mathbf{V}\,\mathbf{z}}. \tag{2.26}$$

Definition 16. ***Equal risk contribution*** *is attained when the marginal risk contribution is equal across all assets in the portfolio, specifically:*

$$MRC(\mathbf{z}_j) = \frac{1}{d_z}, \forall j \in (1, 2, ..., d_z) \tag{2.27}$$

There are many ways to measure the portfolio concentration of risk. We follow Costa and Kwon [38] and consider the Herfindahl index of risk contributions as a measurement of ERC cost.

Definition 17. *The* ***ERC cost function*** *is defined as the Herfindahl index of portfolio risk contributions:*

$$c_{ERC}(\mathbf{z}, \mathbf{V}) = \sum_{j=1}^{m}\left(\frac{\mathbf{z}_j \odot (\mathbf{V}\,\mathbf{z})_j}{\mathbf{z}^T\,\mathbf{V}\,\mathbf{z}}\right)^2. \tag{2.28}$$

Note the values of $c_{ERC}(\mathbf{z}, \mathbf{V})$ range between $1/d_z$ for equal-risk-contributions, and 1 for a fully concentrated portfolio.

Minimizing the Herfindahl index directly is challenging as c_{ERC} is not convex in $\mathbf{z}$. Instead we follow Maillard et al. [93] and consider a minimum-variance optimization subject to a diversification constraint:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\,\mathbf{z}^T\,\mathbf{V}\,\mathbf{z} \\ \text{subject to} \quad & \sum_{j=1}^{d_z}\ln(\mathbf{z}_j) \geq h \\ & \mathbf{z} \geq 0 \end{aligned} \tag{2.29}$$

where h is an arbitrary constant. By relaxing the long-only constraint we obverse that various solutions to Program (2.29) exist. In general, there exists 2^{d_z} orthants in $\mathbb{R}^{d_z}$ and therefore there exists at most 2^{d_z} unique portfolios that achieve the ERC condition. In this thesis we choose to work with the more general risk-parity program, presented presented by Bai et al. [7], which allows for an arbitrary allocation of portfolio risk and supports positive and negative asset weights.

Definition 18. *The **risk parity portfolio** satisfies the condition $MRC(\mathbf{z}) = \mathbf{r}$ and is defined as:*

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\,\mathbf{z}^T\,\mathbf{V}\,\mathbf{z} - \sum_{j=1}^{d_z} \mathbf{r}_j \ln(-\,\mathbf{G}_{jj}\,\mathbf{z}_j) \\ \textit{subject to} \quad & \mathbf{G}\,\mathbf{z} \leq 0. \end{aligned} \tag{2.30}$$

Here $\mathbf{r} \in \mathbb{R}^{d_z}$ is a vector of target risk-contribution weights (i.e. $\mathbf{r}_j = 1/d_z$ for equal-risk-contribution). The diagonal matrix $\mathbf{G} \in \mathbb{R}^{d_z \times d_z}$ constrains the optimization to the relevant orthant of interest, with diagonal elements $\mathbf{G}_{jj} \in \{-1, 1\}$.

Note that Program (2.30) is as an inequality constrained convex program. Furthermore, as noted by Spinu [113], the objective function in Program (2.30) is self-concordant and therefore can be solved efficiently by Newton's Method. We conclude this section by providing the KKT optimality conditions for Program (2.30), which again are fundamental to the methodology described in Chapter 4. The Lagrangian of Program (2.30) is given by:

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}) = \frac{1}{2}\,\mathbf{z}^T\,\mathbf{V}\,\mathbf{z} - \sum_{j=1}^{d_z} \mathbf{r}_j \ln(-\,\mathbf{G}_{jj}\,\mathbf{z}_j) + \boldsymbol{\lambda}^T(\mathbf{G}\,\mathbf{z}) \tag{2.31}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{d_z}$ are the dual variables associated with the inequality constraints. The KKT conditions for stationarity, primal feasibility, and complementary slackness are then given by equations (2.32).

$$\begin{aligned} \mathbf{V}\,\mathbf{z}^* - \mathbf{r} \odot \mathbf{z}^{*-1} + \mathbf{G}^T\,\boldsymbol{\lambda}^* &= 0 \\ \mathbf{G}\,\mathbf{z}^* &\leq 0 \\ \operatorname{diag}(\boldsymbol{\lambda}^*)(\mathbf{G}\,\mathbf{z}^*) &= 0 \\ \boldsymbol{\lambda}^* &\geq 0 \end{aligned} \tag{2.32}$$

Chapter 3

Integrated prediction and optimization

In this chapter we describe the **integrated prediction and optimization** (IPO) framework in a generally setting. We begin with a review of existing literature on integrated prediction and optimization. We then define the traditional 'predict, then optimize' framework which will help draw the distinction to the IPO alternative. In Section 3.3 we formalize the IPO framework as a bi-level optimization program. We conclude with a motivating example from portfolio optimization and promote a prediction modelling process that is optimized for downstream decision errors.

3.1 Literature review

Many problems in machine learning, engineering, operations research and finance involve both predictive forecasting and decision-based optimization. Recall, that a traditional 'predict, then optimize' framework treats the prediction estimation problem independently from its use in the downstream decision optimization problem. As such, an 'objective mismatch' [84] can occur whereby improved prediction accuracy does not necessarily result in smaller decision error. Integrated prediction and optimization methods, on the other hand, date back to the work of Vapnik [122] who proposed training prediction models based on empirical risk minimization objectives. In quantitative finance, IPO methods were first presented by Bengio [12] who proposed training prediction models based on financial risk and return objectives, rather than traditional objectives such as least-squares or maximum likelihood.

In recent years there has been a growing body of research on data-driven decision-making and the relative merits of decoupled versus integrated predictive decision-making. Notably, Elmachtoub and Grigas [47] introduce Smart 'Predict, then Optimize' (SPO), a general framework for optimizing linear regression models in order to minimize the final downstream decision regret. They derive the SPO loss function and demonstrate Fisher consistency with the least-squares loss. Optimizing the SPO loss directly, however, is challenging due to non-convexity, non-differentiability and the presence of pathological solutions ($\mathbf{c} = 0$). Instead the author's derive a convex surrogate loss function which is optimized by first-order sub-gradient descent. They demonstrate that optimizing prediction models based on the SPO surrogate loss function can lead to reduced decision error in comparison to a traditional approach. Subsequently, Elmachtoub et al. [48] present SPO Trees (SPOT) which optimizes regression trees [23] and regression forests [22] to directly minimize the SPO loss. The author's demonstrate that SPOT models with low model complexity typically exhibit equal, and in some cases, smaller decision regret in comparison to traditional regression tree models of higher complexity.

Related, Kallus and Mao [78] explore contextual trees, a data-driven framework for training decision tree forests to minimize downstream decision objectives. Their framework is efficient in that they approximate the greedy determination of optimal tree splits by performing perturbation analysis at the optimality conditions of candidate split points. The author's present applications to mean-variance portfolio optimization and route planning and demonstrate that their contextual optimization framework can result in reduced decision risk. Similarly, Konishi and Fukunaga [83] consider a gradient boosting tree framework under the SPO loss for a subset of optimization problems; namely with linear inequality constraints. They demonstrate that their second-order boosting framework is competitive with state-of-the-art gradient boosting.

Bertsimas and Kallus [13] provide a general framework for conditional decision making whereby the conditional density is estimated through a variety of machine learning methods, including k-nearest neighbours, kernel smoothing and locally weighted regression. They generate optimal, feature dependent, decision policies and also consider the setting where the decision policy affects subsequent realizations of the uncertainty variables. They also consider an empirical risk minimization framework for generating predictive prescriptions and discuss the relative trade-offs of such an approach. Similarly, Ban and Rudin [9] present a direct empirical risk minimization approach using nonparametric kernel regression as the core prediction method. They consider a data-driven newsvendor problem and demonstrate that their approach outperforms the best-practice benchmark when evaluated out-of-sample. Related, Kannan et al. [79] present three frameworks for integrating

machine learning prediction models within a stochastic optimization setting. Their primary contribution is in using the out-of-sample residuals from leave-one-out prediction models to generate scenarios which are then optimized in the context of the decision optimization program. Their frameworks are flexible and accommodate parametric and nonparametric prediction models, for which they derive convergence rates and finite sample guarantees.

Alternatively, recent advances in neural network architecture allow for the integration of predictive modelling with downstream convex optimization programs in an end-to-end trainable neural network. This is discussed in more detail in Chapter 4. The neural network IPO formulation is related to the work of Gould et al. [66] who study bi-level optimization problems in a general setting and provide closed-form expressions for the gradient and Hessian under various constraint assumptions. More recently, Agrawal et al. [2] and Amos and Kolter [3] provide general frameworks for embedding convex optimization programs in an end-to-end neural network. The development of efficient and modular convex optimization layers has been fundamental to the advancement of end-to-end integration of prediction and optimization [1, 15]. Indeed, growing empirical evidence advocates for a fully integrated neural network learning approach. For example, Donti et al. [45] propose an end-to-end stochastic programming framework for estimating the parameters of a probability density function in order to minimize their downstream 'task-based' loss. They consider applications from power scheduling and battery storage and demonstrate that their task-based end-to-end approach can result in lower realized costs in comparison to traditional maximum likelihood estimation and a 'black-box' neural network. Similarly, Wilder et al. [126], Mandi and Guns [94] and Tan et al. [117] consider a bi-level formulation for learning linear programs and mixed integer linear programs from optimal decisions. In all cases the decision-based problem variables are trained to local optimality by (stochastic) gradient descent methods, discussed in more detail in Chapter 4.

Most relevant to the field of portfolio optimization is the work of Zhang et al. [129], who present a general end-to-end framework for portfolio optimization that is capable of supporting a variety of portfolio objectives and constraints. Here, the authors do not impose a specific portfolio optimization framework and instead portfolio objectives and constraints are achieved through standard neural network layer infrastructure. Similarly, Uysal et al. [121] present an end-to-end risk budgeting portfolio optimization framework for training risk-parity portfolios to maximize downstream investor objectives. The authors also consider a 'black-box' neural network approach that does not impose an explicit portfolio optimization framework. Most recently, Chevalier et al. [33] propose a supervised portfolio optimization framework for learning optimal portfolio risk and return characteristics and constraints. In general they find that the fully integrated approach results in improved risk-adjusted

performance. We note that the previous three pieces of work reference our initial findings on integrated prediction and optimization in mean-variance portfolio optimization, described in Chapter 5. Furthermore, the aforementioned work advocates strongly for an IPO approach, and in many cases the IPO models exhibit improved out-of-sample decision-making in comparison to a traditional 'predict, then optimize' approach. We now present the 'predict, then optimize' framework and then compare the traditional approach with the IPO alternative.

3.2 Traditional 'predict then optimize'

We assume that we have a decision cost function, $c\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_y} \to \mathbb{R}$, which takes as input a decision vector $\mathbf{z} \in \mathbb{R}^{d_z}$ and a quantity of interest $\mathbf{y}^{(i)} \in \mathbb{R}^{d_y}$. The decision vector is constrained to a feasible region $\mathbb{S} \subseteq \mathbb{R}^{d_z}$. Recall, in portfolio optimization, $\mathbf{z}$ represents the portfolio weights and $\mathbf{y}^{(i)}$ denotes the asset returns at time i. Clearly, if the realization $\mathbf{y}^{(i)}$ is known then generating optimal decisions amounts to solving the following deterministic optimization problem:

$$\underset{\mathbf{z}\in\mathbb{S}}{\text{minimize}}\, c(\mathbf{z}, \mathbf{y}^{(i)}), \tag{3.1}$$

with optimal solution:

$$\mathbf{z}^*(\mathbf{y}^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}}{\text{argmin}}\, c(\mathbf{z}, \mathbf{y}^{(i)}). \tag{3.2}$$

In reality, we do not know the true values of $\mathbf{Y} = \{\mathbf{y}^{(i)}\}_{i=1}^m$ at decision time. Instead, the quantities of interest must be estimated through associated feature data $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^m$, of covariates of $\mathbf{Y}$, with $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x}$. We define a prediction model by the function $f\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_y}$, with prediction model parameter $\boldsymbol{\theta}$ constrained to the feasible region $\Theta \subseteq \mathbb{R}^{d_\theta}$. We model the quantity of interest, $\mathbf{y}^{(i)}$, as:

$$\mathbf{y}^{(i)} = f(\mathbf{x}^{(i)}, \boldsymbol{\theta}) + \boldsymbol{\epsilon}^{(i)}, \tag{3.3}$$

where $\boldsymbol{\epsilon}^{(i)} \in \mathbb{R}^{d_y}$ is the vector of residual errors. For a particular $\boldsymbol{\theta}$, we define the estimate $\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}$ as:

$$\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)} = f(\mathbf{x}^{(i)}, \boldsymbol{\theta}). \tag{3.4}$$

In traditional parameter estimation, $\boldsymbol{\theta}$ would be chosen in order to minimize a prediction-based

loss function, $\ell\colon \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$, such as least-squares or negative log-likelihood. Specifically, given training data set $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$, a traditional 'predict, then optimize' framework first estimates $\hat{\boldsymbol{\theta}}$ such that:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, \mathbb{E}_D[\ell(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}, \mathbf{y}^{(i)})], \tag{3.5}$$

where $\mathbb{E}_D$ denotes the expectation operator with respect to the training distribution D. Once $\hat{\boldsymbol{\theta}}$ has been determined, a traditional 'predict, then optimize' framework, would then simply 'plug-in' the estimate, $\hat{\mathbf{y}}(\hat{\boldsymbol{\theta}})^{(i)}$, into program (3.1) in order to generate the optimal decision:

$$\mathbf{z}^*(\hat{\mathbf{y}}(\hat{\boldsymbol{\theta}})^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}}{\operatorname{argmin}}\, c(\mathbf{z}, \hat{\mathbf{y}}(\hat{\boldsymbol{\theta}})^{(i)}). \tag{3.6}$$

Observe that the prediction model problem and decision-based optimization are decoupled processes; first predict, then optimize. At first glance, a 'predict then optimize' approach seems reasonable, if not optimal. Clearly, if a prediction model provides perfect forecast accuracy ($\hat{\mathbf{y}}^{(i)} = \mathbf{y}^{(i)}$), then generating optimal decisions amounts to solving a deterministic optimization program. In reality, however, all prediction models do make some error and as such, an inefficiency exists in the traditional 'predict then optimize' paradigm: prediction models are optimized with unique *prediction-based* objectives and constraints, and thus are unaware of how those predictions will ultimately be used in the context of the *decision-based* optimization.

3.3 Integrated prediction and optimization

In an integrated prediction and optimization (IPO) framework we optimize the prediction model parameter, $\boldsymbol{\theta}$, in the context of the downstream decision optimization problem. Specifically, we seek to minimize the average realized cost of the decisions: $\{\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})\}_{i=1}^{m}$. We formally define the IPO prediction estimation problem as a stochastic bi-level optimization problem.

Definition 19. *The* ***integrated prediction and optimization*** *problem is a bi-level empirical risk minimization problem, where the objective is to choose $\boldsymbol{\theta}$ in order to minimize the average empirical decision cost:*

$$\begin{aligned} \underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \quad & \mathbb{E}_D[c(\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})), \mathbf{y})] \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})) = \underset{\mathbf{z}\in\mathbb{S}}{\operatorname{argmin}}\, c(\mathbf{z}, \hat{\mathbf{y}}(\boldsymbol{\theta})). \end{aligned} \tag{3.7}$$

In practice, we are typically presented with discrete observations $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$ and therefore we can approximate the expectation by its sample average approximation [110]. The IPO problem is presented in discrete form in program (3.9).

Definition 20. *Let* $\bar{c}\colon \mathbb{R}^{d_\theta} \to \mathbb{R}$ *denote the sample average cost induced by the decision policies* $\{\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})\}_{i=1}^{m}$*:*

$$\bar{c}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} c(\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}, \mathbf{y}^{(i)}). \tag{3.8}$$

The ***discrete integrated prediction and optimization*** *problem is then given by the sample average approximation:*

$$\begin{aligned} \underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \quad & \bar{c}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} c(\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}, \mathbf{y}^{(i)}) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}}{\operatorname{argmin}}\, c(\mathbf{z}, \hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}) \quad \forall i \in 1, ..., m. \end{aligned} \tag{3.9}$$

Observe that for a fixed instantiation $\boldsymbol{\theta}$, the decision policy, $\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})$, is optimal in the context of its decision optimization program with predictions of the form $\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)} = f(\mathbf{x}^{(i)}, \boldsymbol{\theta})$. The IPO formulation therefore results in a complicated dependency of the model parameters, $\boldsymbol{\theta}$, on the optimized values, $\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})$, connected through the argmin function. This problem is frequently encountered in modern machine learning applications and there are several approaches for overcoming this challenge. For example, the problem can be structured as a bi-level optimization problem in which, under special constraint cases, an analytical solution for the gradient and Hessian exists [66]. Alternatively, heuristic methods, such as the one provided by Sinha et al. [111], propose structuring the problem as a bi-level optimization and use kriging (Gaussian process models) to approximate the mapping to the lower-level optimization problem.

In general, solving the IPO Program (3.9) over prediction variables $\boldsymbol{\theta}$ is challenging for several reasons. First, even in the case where the decision program is convex, the resulting integrated program is likely not convex in $\boldsymbol{\theta}$ and therefore we have no guarantee that a particular local solution is globally optimal. Approximate locally optimal solutions are achievable by first-order gradient descent methods if $\bar{c}(\boldsymbol{\theta})$ is differentiable with respect to $\boldsymbol{\theta}$. Computing the gradient, $\nabla_{\boldsymbol{\theta}}\bar{c}$, however, remains difficult as it requires differentiation through the argmin operator. Moreover, solving program (3.8) through iterative descent methods requires that at each iteration we solve up to m instances of $\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})$, which can be computationally expensive. In the following chapters we address all of these challenges and provide practical and efficient IPO frameworks for a variety of downstream portfolio optimization problems.

3.4 Motivating example

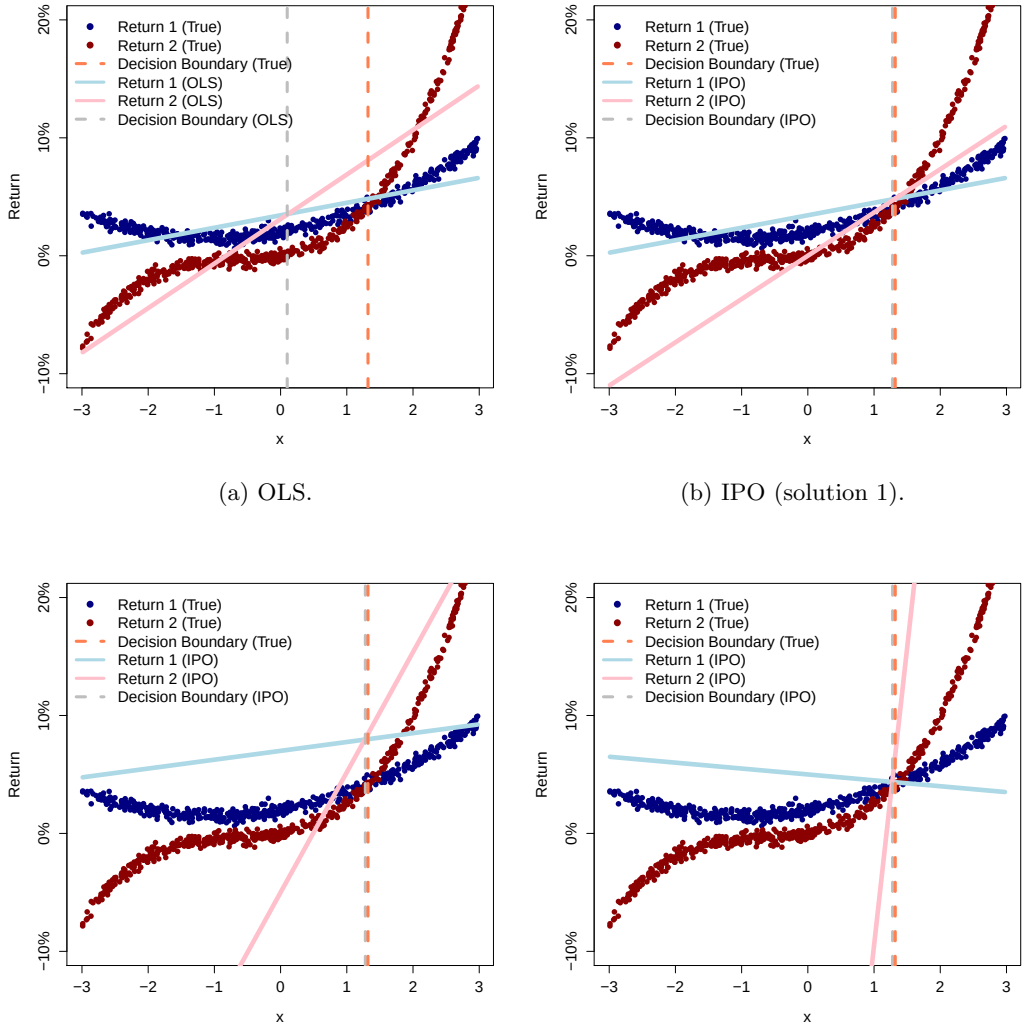
We provide a motivating example to help illustrate the behaviour of the IPO solution in comparison to a traditional 'predict, then optimize' approach. We are motivated by the work of [47, 48] who demonstrate that optimizing prediction model parameters to minimize decision regret produces prediction models with lower complexity and improved out-of-sample performance. We consider the following maximum return portfolio optimization problem:

$$\begin{aligned} \underset{\mathbf{z}}{\text{maximize}} \quad & \mathbf{z}_1\,\mathbf{r}_1 + \mathbf{z}_2\,\mathbf{r}_2 \\ \text{subject to} \quad & \mathbf{z}_1 + \mathbf{z}_2 = 1 \\ & \mathbf{z} \geq 0 \end{aligned} \tag{3.10}$$

where $\mathbf{z}_1$ and $\mathbf{z}_2$ denote the proportion of capital allocated to asset 1 and 2, respectively. We generate a dataset of 500 observations where the feature, $\mathbf{x}$, is drawn from the standard uniform distribution and the return of each asset is given by $\mathbf{r}_1 = \mathbf{x} + 0.5\,\mathbf{x}^2 + \epsilon$ and $\mathbf{r}_2 = \mathbf{x} + \mathbf{x}^2 + 0.5\,\mathbf{x}^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.30)$.

Figure 3.1 plots the return of each asset as a function of the feature variable $\mathbf{x}$. Observe that in this simple example, there exists an optimal decision boundary at $\mathbf{x} \approx 1.32$, denoted with a vertical orange line. Specifically, when $\mathbf{x} < 1.32$, the optimal decision is to assign 100% of the capital to asset 1 (in blue), and conversely when $\mathbf{x} > 1.32$, the optimal decision is to assign 100% of the capital to asset 2 (in red). In Figure 3.1a, the light red and light blue lines denote the optimal ordinary least-squares (OLS) predictions. We observe that in a traditional 'predict, then optimize' framework, while the prediction models are optimal from a prediction accuracy standpoint, they invariably lead to suboptimal decision making. Indeed, the decision boundary implied by the OLS prediction model occurs at $x \approx 0.10$. Therefore there is a decision optimality gap between $0.10 \leq \mathbf{x} \leq 1.32$ where the optimal OLS predictions induce sub-optimal decision making.

In contrast, in Figure 3.1b, the light red and light blue lines denote the optimal IPO predictions. We observe that the IPO predictions are necessarily sub-optimal from a prediction accuracy standpoint, however, they induce optimal decision making. Indeed, the decision boundary implied by the IPO prediction model overlaps perfectly with the true optimal decision boundary. Therefore, in this example, by optimizing the prediction model to minimize the downstream decision objective we are able to reduce the decision error to zero.



(a) OLS.

(b) IPO (solution 1).

(c) IPO (solution 2).

(d) IPO (solution 3).

Figure 3.1: Comparison of the OLS and IPO predictions as a function of the feature variable $\mathbf{x}$ for the maximum return problem (3.10).

We acknowledge that one can make several arguments against the IPO framework. An obvious first argument is that clearly we have misspecified the prediction model class. Namely, we are trying to approximate degree 2 and degree 3 polynomials with a linear prediction model. In this example, properly specified polynomial prediction models would also lead to zero decision error. However, in many real-world applications, the true underlying response relationship is not known, and instead prediction model complexity must be estimated through validation procedures. In highly noisy environments, as is the case with financial market data, estimating optimal model complexity can be challenging. Indeed, asset returns are often characterized as time-varying and

reactive [16, 46, 51, 72, 104, 114], and typically exhibit extremely low signal-to-noise ratios (SNRs) [69, 74]. As a result, low variance models, like simple linear regression, tend to generalize out-of-sample and are often preferred over models of higher complexity. By optimizing prediction models for the downstream decision cost, the IPO framework makes more efficient use of the training data and surrounding problem structure, while keeping prediction model complexity low.

Further arguments against an IPO approach is that, in most cases, the optimal prediction model parameterization is not unique and, moreover, can result in predictions that are counter-intuitive in the context of the training dataset. Indeed, Figures 3.1c and 3.1d provide two distinctly different, yet optimal, IPO predictions, and thus clearly the IPO solution is not unique. In fact, in this example, an entire family of solutions exist, and is characterized by the predicted return of asset 2 intersecting (from below) with the predicted return of asset 1 at $\mathbf{x} \approx 1.32$. Non-uniqueness of the IPO solution, in general, is a result of the non-convexity of the bi-level IPO formulation in Program (3.9). Furthermore, Figure 3.1d highlights a realization whereby the IPO predictions are counterintuitive when compared with the true response values. We observe the predicted returns in Figure 3.1d are vastly different than both the true returns and the corresponding OLS predictions. In fact, in this example, the slope of the IPO predictions of asset 1 is negative whereas the corresponding OLS slope is positive. The lack of intuitive and, to some extent, interpretable predictions is a potential weakness of the IPO approach. However, when we view the decision-making system holistically we note that a traditional 'predict, then optimize' framework can result in inconsistent or competing objectives whereby improved prediction accuracy does not necessarily result in smaller decision error. In contrast, the IPO framework optimizes prediction models in order to directly minimize the final downstream decision cost and thus prediction model and decision model incentives are aligned. In the following chapters we demonstrate the benefit of the IPO framework, which in many cases results in improved out-of-sample decision making.

Chapter 4

Differentiable optimization layers

Solving the IPO problem to global optimality is difficult as Program (3.9) is generally non-convex. In this chapter we discuss first-order gradient descent methods for computing locally optimal solutions to the IPO program. As mentioned perviously, computing the gradient, $\nabla_{\boldsymbol{\theta}} \bar{c}$, is non-trivial as it requires differentiation through the argmin operator, which ultimately requires computing the action of the Jacobian of the optimal solution, $\mathbf{z}^*$, with respect to the relevant optimization program input variables. We begin this chapter by introducing the concept of a differentiable optimization layer (DOL). Recent advances in neural network architecture allow for the embedding of optimization programs as specialized differentiable layers within a larger neural network structure. First-order methods that integrate predictive modelling with downstream decision-based optimization in an end-to-end trainable neural network are therefore possible through backpropagation [108]. We present two approaches for differentiating through the argmin operator of a convex optimization program, namely *unrolled differentiation* and *implicit differentiation* of a fixed-point mapping. We follow the work of Amos and Kolter [3] and Barratt [10], and demonstrate that the KKT conditions, which define optimality of $\mathbf{z}^*$, provide a fixed-point mapping for implicitly computing the required Jacobian. For completeness, we present the OptNet layer [3], a specialized DOL for implicit differentiation of small-scale linearly constrained quadratic programs. We conclude this chapter by providing the DOL architecture for equal-risk-contribution portfolios.

4.1 Introduction to differentiable optimization layers

Recent advances in neural network architecture allow for seamless integration of convex optimization programs as differentiable layers in an end-to-end trainable neural network [1, 2, 3]. In general, a

convex optimization program, as described in Section 2.1, can be viewed as a function that maps the program input variables to optimal primal (dual) solution(s). DOLs therefore provide an efficient and modular framework for the integrating prediction modelling with downstream decision-based optimization.

Modern neural network technology (such as *torch* or *tensorflow*) require that every layer in the network inherits a forward and backward-pass routine. For DOLs , the forward-pass is typically an iterative optimization algorithm that converts problem variables into optimal primal (dual) solution(s). The backward-pass therefore computes the action of the Jacobian of the optimal solution(s) with respect to all problem variables, and in backpropagation returns the left matrix-vector product of the Jacobian with the previous backward-pass gradient(s).

For example, the OptNet layer, described in more detail in Section 4.4, implements a primal-dual interior-point method for solving small-scale batch constrained quadratic programs [3]. For backward differentiation, the authors implement a novel and efficient argmin differentiation routine that implicitly differentiates the KKT system of equations at optimality. By strategically caching the factorized KKT left-hand side matrix then the resulting method is shown to be computationally tractable for small problems within the context of deep neural network architectures. The author's acknowledge, however, that the OptNet layer may be impractical for optimization problems with a moderate to large number of variables.

More recently, Agrawal et al. [2] provide a general framework for differentiable convex cone programming. Their forward-pass recasts the conic program in its equivalent homogeneous self-dual embedding form, which is then solved by operator splitting [103]. In the backward-pass, the relevant gradients are obtained by implicit differentiation of the residual map provided by the homogeneous self-dual equations. The resulting differentiable cone programming layer is flexible, but requires the user to transform their problem into a canonical form, which is often time-consuming, prone to error and requires a certain level of domain expertise.

Alternatively, Agrawal et al. [1], provide a domain-specific language for differentiable disciplined convex programs. Their approach abstracts away the process of converting problems to canonical form with minimal loss in computational efficiency in comparison to specialized convex optimization layers. They also provide an efficient sparse matrix solver, which for sparse quadratic programs is on average an order of magnitude faster than the OptNet layer. Similarly, Blondel et al. [15] provide an efficient and modular approach for implicit differentiation of optimization problems. They consider KKT, proximal gradient and mirror descent fixed-point implicit differentiation and provide a software infrastructure for efficiently integrating their modular implicit differentiation routines with state-of-

the-art optimization solvers. That said, for solving batches of convex optimization problems it is often preferred and more efficient to avail of optimization solvers that have the ability to exploit fast GPU-based batch solves.

Indeed, embedding optimization programs in a larger neural network system has been fundamental to recent innovations in signal processing, compressed sensing, imaging and statistics (see for example [42, 127, 128]). Moreover, DOLs provide a state-of-the-art framework for locally solving difficult integrated prediction and optimization problems, with recent applications in control [4], finance [27, 28, 121], energy management [45] and route planning [95, 94].

In this thesis, we often solve the IPO program (3.9) by restructuring the bi-level program as a fully-integrated end-to-end trainable neural network. The IPO equivalent neural network structure is depicted in Figure 4.1. In the forward pass, the input layer takes the feature variables $\mathbf{x}^{(i)}$ and passes them to a prediction model layer to produce the estimates, $\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}$. The predictions are then passed to a differentiable optimization layer which, for a given input $\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}$, solves the convex optimization program and returns the optimal decisions $\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})$. Finally, the quality of the optimal decision is evaluated by the cost function, $c(\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}), \mathbf{y}^{(i)})$ in the context of the true value $\mathbf{y}^{(i)}$.

The partial derivative of the cost with respect to the estimate, $\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}$, is computed by the argmin differentiation process described in the next section. The backpropagation algorithm then computes the remaining partial derivatives in the regular fashion in order to generate the gradient of the realized cost with respect to the prediction model parameter.

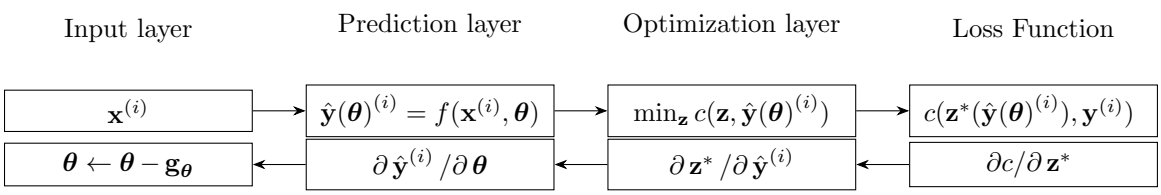


Figure 4.1: IPO program represented as an end-to-end neural network with predictive linear layer, differentiable optimization layer and decision cost loss function.

We now have what amounts to a first-order algorithm for optimizing prediction model parameters in the context of their final decision-based optimization cost. In practice, we search for a locally optimal solution using stochastic gradient descent (SGD). In this case, the descent direction, $\boldsymbol{g}_\theta$, at each iteration approximates the gradient, $\nabla_{\boldsymbol{\theta}} \bar{c}$, and is given by :

$$\boldsymbol{g}_\theta = \frac{1}{|B|} \sum_{i \in B} \Big(\frac{\partial c}{\partial \boldsymbol{\theta}} \Big)_{|(\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}), \mathbf{y}^{(i)})} \approx \nabla_{\boldsymbol{\theta}} \bar{c}$$

where in stochastic gradient descent, B represents a randomly drawn sample batch from the data.

4.2 Unrolled differentiation

The objective is to re-formulate the lower-level decision-based optimization problem of Program (3.9) as a differentiable optimization layer. Recall that, in our case, we have m decision-based programs of the form:

$$\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}) = \underset{\mathbf{z}\in\mathbb{Z}}{\operatorname{argmin}}\, c(\mathbf{z}, \hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}) \quad \forall\ i \in 1, ..., m. \tag{4.1}$$

Solving Program (4.1) in the forward-pass routine is straightforward; the convex optimization layer takes as input the problem variables, $\{\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}\}_{i=1}^m$, and returns the optimal decisions: $\{\mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})\}_{i=1}^m$. There are many ways to solve Program (4.1) to global optimality, including but not limited to interior-point and active-set methods, as well as a plethora of first-order gradient based methods. Most convex optimization algorithms are iterative in nature and we refer the reader to Griva et al. [68] for a general overview.

The backward-pass routine requires computing the action of the Jacobian of the optimal solution with respect to the input variables, namely $\nabla_{\boldsymbol{\theta}}\, \mathbf{z}^*(\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)})$. However, if each operation in the forward-pass routine is differentiable, then it is possible to compute the action of the Jacobian by simply applying the chain-rule to each operation that the input variable, $\hat{\mathbf{y}}(\boldsymbol{\theta})^{(i)}$, encounters in the forward-pass routine. As the name suggests, **unrolled differentiation** simply unrolls the forward-pass computational graph into its individual operations and applies standard backpropogation. We refer the reader to Domke [43] and Diamond et al. [42] for a more comprehensive overview of unrolled differentiation techniques. Indeed, while unrolled differentiation is conceptually simple, it is generally memory inefficient and typically requires substantially larger networks [4]. Furthermore, when the number of iterations required to solve the convex optimization program is large in the forward-pass then the unrolled gradient can be computationally expensive to compute [26]. In the following section, we present an alternative and computationally efficient implicit differentiation framework.

4.3 KKT implicit differentiation

In this section we present **implicit differentiation** as an efficient alternative to unrolled differentiation. We follow the work of [1, 2, 3, 15] and begin by noting that for many convex optimization algorithms, the optimization iterations that map problem input variables to optimal solutions can be reduced to a fixed-point mapping. In particular, for any differentiable convex optimization program, the KKT optimality conditions, described in Section 2.1, is a fixed-point mapping in the primal and dual variables. As outlined by [3, 10], it is therefore possible to apply the implicit function theorem

and derive the gradient of the primal-dual variables with respect to the input problem variables. In this section we provide the KKT implicit differentiation for general convex optimization problems. We begin with a few definitions.

Definition 21. *Let* $F\colon \mathbb{R}^{d_v} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_v}$ *be a continuously differentiable function with variable* $\mathbf{v}^*$ *and parameter* $\boldsymbol{\theta}$*. We define* $\mathbf{v}^*$ *as a* **fixed-point** *of* F *at* $(\mathbf{v}^*, \boldsymbol{\theta})$ *if:*

$$F(\mathbf{v}^*, \boldsymbol{\theta}) = \mathbf{v}^* .$$

Definition 22. *The* **residual map**, $G\colon \mathbb{R}^{d_v} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_v}$ *of a fixed-point,* $(\mathbf{v}^*, \boldsymbol{\theta})$*, of* F *is given by:*

$$G(\mathbf{v}^*, \boldsymbol{\theta}) = F(\mathbf{v}^*, \boldsymbol{\theta}) - \mathbf{v}^* = 0.$$

The **implicit function theorem**, as defined by Dontchev and Rockafellar [44], then provides the conditions on G for which the Jacobian of the solution mapping with respect to $\boldsymbol{\theta}$ is well defined.

Theorem 1. *Let* $G\colon \mathbb{R}^{d_v} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_v}$ *be a continuously differentiable function in a neighbourhood of* $(\mathbf{v}^*, \boldsymbol{\theta})$ *such that* $G(\mathbf{v}^*, \boldsymbol{\theta}) = 0$*. Denote the nonsingular partial Jacobian of* G *with respect to* $\mathbf{v}^*$ *as* $\nabla_{\mathbf{v}^*} G(\mathbf{v}^*, \boldsymbol{\theta})$*. Then* $\mathbf{v}^*(\boldsymbol{\theta})$ *is an implicit function of* $\boldsymbol{\theta}$ *and is continuously differentiable in a neighbourhood,* $\mathcal{O}$*, of* $\boldsymbol{\theta}$ *with Jacobian:*

$$\nabla_{\boldsymbol{\theta}}\, \mathbf{v}^*(\boldsymbol{\theta}) = -[\nabla_{\mathbf{v}^*} G(\mathbf{v}^*(\boldsymbol{\theta}), \boldsymbol{\theta})]^{-1} \nabla_{\boldsymbol{\theta}} G(\mathbf{v}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) \quad \forall \quad \boldsymbol{\theta} \in \mathcal{O}. \tag{4.2}$$

Corollary 1. *Let* $F\colon \mathbb{R}^{d_v} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_v}$ *be a continuously differentiable function with fixed-point* $(\mathbf{v}^*, \boldsymbol{\theta})$*. Then* $\mathbf{v}^*(\boldsymbol{\theta})$ *is an implicit function of* $\boldsymbol{\theta}$ *and is continuously differentiable in a neighbourhood,* $\mathcal{O}$*, of* $\boldsymbol{\theta}$ *with Jacobian:*

$$\nabla_{\boldsymbol{\theta}}\, \mathbf{v}^*(\boldsymbol{\theta}) = [\mathbf{I}_{\mathbf{v}} - \nabla_{\mathbf{v}^*} F(\mathbf{v}^*(\boldsymbol{\theta}), \boldsymbol{\theta})]^{-1} \nabla_{\boldsymbol{\theta}} F(\mathbf{v}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) \quad \forall \quad \boldsymbol{\theta} \in \mathcal{O}. \tag{4.3}$$

4.3.1 KKT implicit differentiation for convex optimization

Recall, from Section 2.1, a convex optimization program is a mathematical optimization program of the form:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & c(\mathbf{z}, \boldsymbol{\theta}) \\ \text{subject to} \quad & g(\mathbf{z}, \boldsymbol{\theta}) \leq 0 \\ & h(\mathbf{z}, \boldsymbol{\theta}) = 0 \end{aligned} \tag{4.4}$$

with optimal primal solution $\mathbf{z}^*$. Note that here we have augmented the cost and constraint functions with the parameter $\boldsymbol{\theta}$ to reflect the fact that some of the problem input variables are functions of $\boldsymbol{\theta}$. The Karush-Kuhn-Tucker (KKT) optimality conditions associated with Program (4.4) are derived from Lagrange duality and are necessary and sufficient for global optimality:

$$\begin{aligned} \nabla_{\mathbf{z}} \mathcal{L}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*, \boldsymbol{\theta}) = \nabla_{\mathbf{z}} c(\mathbf{z}^*, \boldsymbol{\theta}) + \boldsymbol{\lambda}^{*T} \nabla_{\mathbf{z}} g(\mathbf{z}^*, \boldsymbol{\theta}) + \boldsymbol{\eta}^{*T} \nabla_{\mathbf{z}} h(\mathbf{z}^*, \boldsymbol{\theta}) = 0 \\ g(\mathbf{z}^*, \boldsymbol{\theta}) \leq 0 \\ h(\mathbf{z}^*, \boldsymbol{\theta}) = 0 \\ \operatorname{diag}(\boldsymbol{\lambda}^*) g(\mathbf{z}^*, \boldsymbol{\theta}) = 0 \\ \boldsymbol{\lambda}^* \geq 0, \end{aligned} \tag{4.5}$$

with optimal dual variables $(\boldsymbol{\lambda}^*, \boldsymbol{\eta}^*)$. In general we assume that the regular constraint qualifications (Slater's condition) are satisfied and therefore strong duality holds ($p^* = d^*$). We denote the primal-dual solution by $\mathbf{v}^* = (\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*)$. Following Barratt [10], for a convex program the system of equations provided by the KKT conditions for stationarity, primal feasibility, and complementary slackness therefore defines a fixed-point at optimality $\mathbf{v}^*$ given by:

$$G(\mathbf{v}^*, \boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\mathbf{z}} \mathcal{L}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*, \boldsymbol{\theta}) \\ \operatorname{diag}(\boldsymbol{\lambda}) g(\mathbf{z}^*, \boldsymbol{\theta}) \\ h(\mathbf{z}^*, \boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{4.6}$$

Direct differentiation of the KKT conditions with respect to $\boldsymbol{\theta}$ and $\mathbf{v}^*$ gives the following partial Jacobians:

$$\nabla_{\boldsymbol{\theta}} G(\mathbf{v}^*, \boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}\, \mathbf{z}} \mathcal{L}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*, \boldsymbol{\theta}) \\ \operatorname{diag}(\boldsymbol{\lambda}^*) \nabla_{\boldsymbol{\theta}} g(\mathbf{z}^*, \boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}} h(\mathbf{z}^*, \boldsymbol{\theta}) \end{bmatrix}, \tag{4.7}$$

and

$$\nabla_{\mathbf{v}^*} G(\mathbf{v}^*, \boldsymbol{\theta}) = \begin{bmatrix} \nabla^2_{\mathbf{z}\,\mathbf{z}} \mathcal{L}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*, \boldsymbol{\theta}) & \nabla_{\mathbf{z}} g(\mathbf{z}^*, \boldsymbol{\theta})^T & \nabla_{\mathbf{z}} h(\mathbf{z}^*, \boldsymbol{\theta})^T \\ \operatorname{diag}(\boldsymbol{\lambda}) \nabla_{\mathbf{z}} g(\mathbf{z}^*, \boldsymbol{\theta}) & \operatorname{diag}(g(\mathbf{z}^*, \boldsymbol{\theta})) & 0 \\ \nabla_{\mathbf{z}} h(\mathbf{z}, \boldsymbol{\theta}) & 0 & 0 \end{bmatrix}. \tag{4.8}$$

If the partial Jacobian $\nabla_{\mathbf{v}^*} G(\mathbf{v}^*, \boldsymbol{\theta})$ is nonsingular and $\boldsymbol{\theta}$ is continuously differentiable in some neighbourhood, $\mathcal{O}$, of $\boldsymbol{\theta}$, then by Theorem 1 the Jacobian of the optimal primal-dual solution $\mathbf{v}^*$ with respect to parameter $\boldsymbol{\theta}$ is given by the solution to the following system of equations:

$$\nabla_{\boldsymbol{\theta}}\, \mathbf{v}^*(\boldsymbol{\theta}) = -[\nabla_{\mathbf{v}^*} G(\mathbf{v}^*(\boldsymbol{\theta}), \boldsymbol{\theta})]^{-1} \nabla_{\boldsymbol{\theta}} G(\mathbf{v}^*(\boldsymbol{\theta}), \boldsymbol{\theta}). \tag{4.9}$$

Finally, observe that the Jacobian $\nabla_{\boldsymbol{\theta}}\, \mathbf{v}^*(\boldsymbol{\theta})$ is defined as per Equation (4.9) when the following conditions hold:

1. The fixed point mapping, G, is continuously differentiable in a neighbourhood of $(\mathbf{v}^*, \boldsymbol{\theta})$.
2. The cost function, c, is twice continuously differentiable in $\mathbf{z}$ and continuously differentiable in $\boldsymbol{\theta}$.
3. The constraint functions f and h are continuously differentiable in $\boldsymbol{\theta}$.
4. The partial Jacobian, $\nabla_{\mathbf{v}^*} G(\mathbf{v}^*, \boldsymbol{\theta})$ is nonsingular and thus invertible.

4.4 KKT implicit differentiation for quadratic programming

We now present KKT implicit differentiation for linearly constrained quadratic programs. We note that many of the portfolio optimization programs described in Chapter 2 can be cast as linearly constrained quadratic programs and therefore the KKT implicit differentiation described herein would be applicable. In this section, without loss of generality, we consider the convex parametric quadratic program of the form:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\, \mathbf{z}^T\, \mathbf{Q}(\boldsymbol{\theta})\, \mathbf{z} + \mathbf{z}^T\, \mathbf{p}(\boldsymbol{\theta}) \\ \text{subject to} \quad & \mathbf{A}(\boldsymbol{\theta})\, \mathbf{z} = \mathbf{b}(\boldsymbol{\theta}), \quad \mathbf{G}(\boldsymbol{\theta})\, \mathbf{z} \leq \mathbf{h}(\boldsymbol{\theta}), \end{aligned} \tag{4.10}$$

with decision variable $\mathbf{z} \in \mathbb{R}^{d_z}$. The objective function is therefore defined by a vector $\mathbf{p}(\boldsymbol{\theta}) \in \mathbb{R}^{d_z}$ and symmetric positive definite matrix $\mathbf{Q}(\boldsymbol{\theta}) \in \mathbb{R}^{d_z \times d_z}$. Here, $\mathbf{A}(\boldsymbol{\theta}) \in \mathbb{R}^{d_{eq} \times d_z}$, $\mathbf{b}(\boldsymbol{\theta}) \in \mathbb{R}^{d_{eq}}$ and $\mathbf{G}(\boldsymbol{\theta}) \in \mathbb{R}^{d_{iq} \times d_z}$, and $\mathbf{h}(\boldsymbol{\theta}) \in \mathbb{R}^{d_{iq}}$ define the linear equality and inequality constraints, respectively.

We assume that all problem variables are parameterized by $\boldsymbol{\theta}$ and are therefore trainable when integrated in an end-to-end neural network. In the following analysis, we drop the parameterization $\boldsymbol{\theta}$ for ease of notation.

The OptNet layer, proposed by Amos and Kolter [3] is a specialized differentiable optimization layer that uses a primal-dual interior-point method for small scale batch quadratic programs. The authors demonstrate that the solution to the system of equations provided by the KKT conditions at optimality provide a system of equations for implicit differentiation with respect to all relevant problem variables. We follow the procedure as outlined in Section 4.3.1 and derive the KKT implicit differentiation formula for the convex constrained quadratic program (4.10) . We denote the primal-dual solution at optimality by $\mathbf{v}^* = (\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*)$. Note that all constraints are affine and therefore Slater's condition reduces to feasibility. Following Amos and Kolter [3], the KKT conditions for stationarity, primal feasibility, and complementary slackness therefore defines a fixed-point at optimality $\mathbf{v}^*$ given by:

$$G(\mathbf{v}^*, \boldsymbol{\theta}) = \begin{bmatrix} \mathbf{p} + \mathbf{Q}\,\mathbf{z}^* + \mathbf{G}^T\,\boldsymbol{\lambda}^* + \mathbf{A}^T\,\boldsymbol{\eta}^* \\ \operatorname{diag}(\boldsymbol{\lambda}^*)(\mathbf{G}\,\mathbf{z}^* - \mathbf{h}) \\ \mathbf{A}\,\mathbf{z}^* - \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.11}$$

Applying Theorem 1, we take the differential of these conditions to give the following system of equations:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{G}^T & \mathbf{A}^T \\ \operatorname{diag}(\boldsymbol{\lambda}^*)\,\mathbf{G} & \operatorname{diag}(\mathbf{G}\,\mathbf{z}^* - \mathbf{h}) & 0 \\ \mathbf{A} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{dz} \\ \mathbf{d}\,\boldsymbol{\lambda} \\ \mathbf{d}\,\boldsymbol{\nu} \end{bmatrix} = - \begin{bmatrix} \mathbf{dQ}\,\mathbf{z}^* + \mathbf{dp} + \mathbf{dG}^T\,\boldsymbol{\lambda}^* + \mathbf{dA}^T\,\boldsymbol{\eta}^* \\ \operatorname{diag}(\boldsymbol{\lambda}^*)\mathbf{dG}\,\mathbf{z}^* - \operatorname{diag}(\boldsymbol{\lambda}^*)\mathbf{dh} \\ \mathbf{dA}\,\mathbf{z}^* - \mathbf{db} \end{bmatrix}. \tag{4.12}$$

Observe that the left side matrix gives the optimality conditions of the convex quadratic problem, which, when solving by interior-point methods, must be factorized in order to obtain the solution to the quadratic program [20]. The right side gives the differentials of the relevant functions at the achieved solution with respect to any of the problem variables. For example, as explained by Amos and Kolter [3], the Jacobian $\partial\,\mathbf{z}^*\,/\partial\,\mathbf{p}$ can be obtained by setting $\mathbf{dp} = \mathbf{I}$ (setting all other differential terms to zero) and solving the resulting system of equations for $\mathbf{dz}$. The Jacobian $\partial\,\mathbf{z}^*\,/\partial\,\boldsymbol{\theta}$ can then be computed by the usual chain-rule:

$$\frac{\partial\,\mathbf{z}^*}{\partial\,\boldsymbol{\theta}} = \frac{\partial\,\mathbf{z}^*}{\partial\,\mathbf{p}}\frac{\partial\,\mathbf{p}}{\partial\,\boldsymbol{\theta}}.$$

In the backpropagation algorithm, however, it is inefficient to explicitly form the right-side Jacobian matrix. Instead we compute the left matrix-vector product of the Jacobian with the previous backward-pass gradient, $\partial \ell / \partial \, \mathbf{z}^*$, where $\ell(\cdot)$ denotes an arbitrary loss function.

$$\begin{bmatrix} \bar{\mathbf{d}}_{\mathbf{z}} \\ \bar{\mathbf{d}}_{\boldsymbol{\lambda}} \\ \bar{\mathbf{d}}_{\boldsymbol{\eta}} \end{bmatrix} = - \begin{bmatrix} \mathbf{Q} & \mathbf{G}^T \operatorname{diag}(\boldsymbol{\lambda}^*) & \mathbf{A}^T \\ \mathbf{G} & \operatorname{diag}(\mathbf{G}\,\mathbf{z}^* - \mathbf{h}) & 0 \\ \mathbf{A} & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \left(\frac{\partial \ell}{\partial \, \mathbf{z}^*}\right)^T \\ 0 \\ 0 \end{bmatrix}. \tag{4.13}$$

Equation (4.13) therefore allows for efficient computation of the gradients with respect to any of the relevant problem variables. This is particularly true when using interior-point methods as the required gradients are effectively obtained 'for free' upon factorization of the left matrix when obtaining the solution, $\mathbf{z}^*$, in the forward-pass. The gradients of the loss with respect to all QP problem variables are presented below.

$$\begin{aligned} \frac{\partial \ell}{\partial \, \mathbf{Q}} &= \frac{1}{2}\left(\bar{\mathbf{d}}_{\mathbf{z}}\, \mathbf{z}^{*T} + \mathbf{z}^*\, \bar{\mathbf{d}}_{\mathbf{z}}^T\right) & \frac{\partial \ell}{\partial \, \mathbf{p}} &= \bar{\mathbf{d}}_{\mathbf{z}} \\ \frac{\partial \ell}{\partial \, \mathbf{A}} &= \bar{\mathbf{d}}_{\boldsymbol{\eta}}\, \mathbf{z}^{*T} + \boldsymbol{\eta}^*\, \bar{\mathbf{d}}_{\mathbf{z}}^T & \frac{\partial \ell}{\partial \, \mathbf{b}} &= -\bar{\mathbf{d}}_{\boldsymbol{\eta}} \\ \frac{\partial \ell}{\partial \, \mathbf{G}} &= \operatorname{diag}(\boldsymbol{\lambda}^*)\bar{\mathbf{d}}_{\boldsymbol{\lambda}}\, \mathbf{z}^{*T} + \boldsymbol{\lambda}^*\, \bar{\mathbf{d}}_{\mathbf{z}}^T & \frac{\partial \ell}{\partial \, \mathbf{h}} &= -\operatorname{diag}(\boldsymbol{\lambda}^*)\bar{\mathbf{d}}_{\boldsymbol{\lambda}} \end{aligned} \tag{4.14}$$

4.5 KKT implicit differentiation for equal-risk-contribution portfolios

The equal-risk-contribution portfolio, described in Section 2.3 is not a quadratic program, and therefore the KKT conditions at optimality and the resulting implicit differentiation equations are different than those presented by Equation (4.14). In this Section we derive the relevant system of equations required for computing gradients of the optimal solution, $\mathbf{z}^*$ with respect to the program input variables. We begin by restating the ERC portfolio optimization program in its most general form:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\, \mathbf{z}^T\, \mathbf{V}(\boldsymbol{\theta})\, \mathbf{z} - \sum_{j=1}^{d_z} \mathbf{r}_j(\boldsymbol{\theta}) \ln(-\,\mathbf{G}(\boldsymbol{\theta})_{jj}\, \mathbf{z}_j) \\ \text{subject to} \quad & \mathbf{G}(\boldsymbol{\theta})\, \mathbf{z} \leq 0 \end{aligned} \tag{4.15}$$

Let $\boldsymbol{\nu}^* = (\mathbf{z}^*, \boldsymbol{\lambda}^*)$ denote the optimal primal-dual solution, where $\boldsymbol{\lambda}^*$ denotes optimal dual variable associated with the linear inequality constraint. The KKT conditions for stationarity, primal feasibility, and complementary slackness provided by equations (2.32) therefore defines a fixed-point at optimality $\boldsymbol{\nu}^*$ given by:

$$G(\boldsymbol{\nu}^*, \boldsymbol{\theta}) = \begin{bmatrix} \mathbf{V}\,\mathbf{z}^* - \mathbf{r} \odot \mathbf{z}^{*-1} + \mathbf{G}^T\,\boldsymbol{\lambda}^* \\ \operatorname{diag}(\boldsymbol{\lambda}^*)(\mathbf{G}\,\mathbf{z}^*) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.16}$$

Note that combining complementary slackness and primal feasibility with the log-barrier yields $\boldsymbol{\lambda}^* = \mathbf{0}$ at optimality. Taking the differentials of these conditions gives the following system of equations:

$$\begin{aligned} \mathbf{d}\mathbf{V}\,\mathbf{z}^* + \mathbf{V}\,\mathbf{d}\mathbf{z} - \mathbf{d}\mathbf{r} \odot \mathbf{z}^{*-1} + \operatorname{diag}(\mathbf{r} \odot \mathbf{z}^{*-2})\,\mathbf{d}\mathbf{z} + \mathbf{d}\mathbf{G}^T\,\boldsymbol{\lambda}^* + \mathbf{G}^T\,\mathbf{d}\,\boldsymbol{\lambda} = 0 \\ \operatorname{diag}(\mathbf{G}\,\mathbf{z}^*)\,\mathbf{d}\,\boldsymbol{\lambda} + \operatorname{diag}(\boldsymbol{\lambda}^*)(\mathbf{d}\mathbf{G}\,\mathbf{z}^* + \mathbf{G}\,\mathbf{d}\mathbf{z}) = 0. \end{aligned} \tag{4.17}$$

Simplifying Equation (4.17) yields the following system of equations:

$$\begin{bmatrix} \mathbf{V} + \operatorname{diag}(\mathbf{r} \odot \mathbf{z}^{*-2}) & \mathbf{G}^T \\ \operatorname{diag}(\boldsymbol{\lambda}^*)\,\mathbf{G} & \operatorname{diag}(\mathbf{G}\,\mathbf{z}^*) \end{bmatrix} \begin{bmatrix} d\boldsymbol{z} \\ d\boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \mathbf{d}\mathbf{V}\,\mathbf{z}^* - \mathbf{d}\mathbf{r} \odot \mathbf{z}^{*-1} + \mathbf{d}\mathbf{G}^T\,\boldsymbol{\lambda}^* \\ \operatorname{diag}(\boldsymbol{\lambda}^*)\,\mathbf{d}\mathbf{G}\,\mathbf{z}^* \end{bmatrix}. \tag{4.18}$$

As in the case of the quadratic program, the left side matrix gives the Hessian of the optimality conditions of Program (4.15), which, when solving by Newton's method, must be factorized in order to obtain the optimal solution $\mathbf{z}^*$ in the forward-pass. The right side gives the differentials of the relevant functions at the achieved solution with respect to any problem input variable. As before, in the backpropagation algorithm we compute the left matrix-vector product of the Jacobian with the previous backward-pass gradient, $\partial \ell / \partial\, \mathbf{z}^*$, demonstrated below:

$$\begin{bmatrix} \bar{\mathbf{d}}_{\mathbf{z}} \\ \bar{\mathbf{d}}_{\boldsymbol{\lambda}} \end{bmatrix} = - \begin{bmatrix} \mathbf{V} + \operatorname{diag}(\mathbf{r} \odot \mathbf{z}^{*-2}) & \mathbf{G}^T \operatorname{diag}(\boldsymbol{\lambda}^*) \\ \mathbf{G} & \operatorname{diag}(\mathbf{G}\,\mathbf{z}^*) \end{bmatrix}^{-1} \begin{bmatrix} \left(\frac{\partial \ell}{\partial\, \mathbf{z}^*}\right)^T \\ 0 \end{bmatrix}. \tag{4.19}$$

The gradients of the loss function with respect to all relevant problem variables are presented below.

$$\frac{\partial \ell}{\partial\, \mathbf{V}} = \frac{1}{2}\left(\bar{\mathbf{d}}_{\mathbf{z}}\,\mathbf{z}^{*T} + \mathbf{z}^*\,\bar{\mathbf{d}}_{\mathbf{z}}^T\right) \qquad \frac{\partial \ell}{\partial\, \mathbf{r}} = -\operatorname{diag}(\mathbf{z}^{*-1})\bar{\mathbf{d}}_{\mathbf{z}} \qquad \frac{\partial \ell}{\partial\, \mathbf{G}} = \operatorname{diag}(\boldsymbol{\lambda}^*)\bar{\mathbf{d}}_{\boldsymbol{\lambda}}\,\mathbf{z}^{*T} + \boldsymbol{\lambda}^*\,\bar{\mathbf{d}}_{\mathbf{z}}^T \tag{4.20}$$

Chapter 5

Integrating prediction in mean-variance portfolio optimization

5.1 Introduction

In this chapter, we consider the integration of linear regression prediction models with downstream mean-variance portfolio optimization. Linear regression and mean-variance optimization (MVO) are two foundational pillars of quantitative asset management. Indeed, we are motivated by the long established history of regression forecasting in the financial literature (see for example [55, 56, 57]). Moreover, from a practical standpoint, forecasting asset mean returns is notoriously difficult as asset returns are characterized as non-stationary [51, 114] and reactive, and typically exhibit extremely low signal-to-noise ratios [74] and insignificant auto-correlation [46]. As a result, low variance models, like simple linear regression, tend to generalize out-of-sample and are often preferred over models of higher complexity. Furthermore, mean-variance optimization, proposed by Markowitz [96], has been widely studied in the portfolio optimization literature and is fundamental to the current-day practice of quantitative asset management (see for example [14, 34, 54, 98]).

The vast majority of the portfolio optimization literature, however, considers a traditional 'predict, then optimize' approach for return forecasting and subsequent portfolio decision-making [32, 36, 64, 91, 116]. In contrast, in this chapter we present the first rigorous study of a general and

direct method for integrating linear regression predictions in a mean-variance portfolio optimization setting, with experimentation using real asset price data. The remainder of the chapter is outlined as follows. In Section 5.2 we provide the IPO formulation for a mean-variance portfolio optimization with a linear prediction model for estimating asset returns. In Section 5.2.1 we follow the methodology as described in Section 4.4 and present a first-order method for locally solving the IPO program for general inequality constrained MVO problems. We then consider several special instances of the MVO decision optimization problem. In particular, we demonstrate that when the MVO program is either unconstrained or contains only linear equality constraints then the IPO problem can be recast as a convex quadratic program and solved analytically. We discuss the sampling distribution properties of the optimal IPO regression coefficients and demonstrate that the IPO solution explicitly minimizes the tracking-error to ex-post optimal mean-variance portfolios.

In Section 5.3 we perform several simulation studies using synthetically generated data. We compare the IPO and ordinary least-squares models in an unconstrained and linear equality constrained setting and demonstrate the resilience of the IPO model to estimation error in both the mean and covariance. In Sections 5.3.2 we discuss the computational challenges of the current state-of-the art solution based on implicit differentiation and first-order gradient descent. We demonstrate the computational advantage of the closed-form IPO solution, which guarantees optimality and is approximately an order of magnitude faster than the state-of-the-art iterative method. In Section 5.3.3 we revisit the more general IPO program whereby the MVO program contains inequality constraints. A simulation study demonstrates that the analytical IPO solution, with inequality constraints removed, can provide superior out-of-sample performance over a wide range of problem parameterizations. Finally in Section 5.4 we conclude with a simulation study using global futures data and demonstrate that the IPO methodology can provide lower realized costs and improved economic outcomes in comparison to the 'predict, then optimize' alternative.

5.1.1 Main Contributions

The IPO framework presented in this chapter is most similar to, and is largely inspired by, the work of Amos and Kolter [3] and Donti et al. [45], as well as the 'smart predict, then optimize' framework proposed by Elmachtoub and Grigas [47]. We refer the reader to the discussion in Section 3.1 for a comprehensive overview of the IPO literature.

While our methodology follows closely to that of Donti et al. [45] and Elmachtoub and Grigas [47], in this chapter we provide several notable differences and extensions. First, Donti et al.

[45] considers the parametric density estimation problem in a stochastic optimization framework whereas we consider a regression based estimation problem with deterministic downstream decision optimization. Secondly, we consider applications from portfolio optimization; specifically the decision program is a convex mean-variance quadratic program. Moreover, we demonstrate that in the case where the solution to the MVO program is linear in the parameter $\boldsymbol{\theta}$, then we can efficiently make use of second-order Hessian information and optimize the integrated problem analytically, or more generally through standard quadratic programming.

The IPO program presented in Equation (3.9) is the most direct and perhaps simplest expression of an integrated prediction and optimization problem. To our knowledge this is the first rigorous study of these equations in a mean-variance portfolio optimization setting. While Elmachtoub and Grigas [47] provide an application to a mean-variance optimization (MVO) problem, their framework requires optimization of a customized and more complicated SPO surrogate loss function. Moreover, the SPO framework can only support optimization problems with a linear objective and therefore the integration is limited to prediction models that remain linear across all relevant problem variables.

In contrast, the simplicity of our IPO formulation lends itself to a very practical and approachable integrated prediction and portfolio optimization solution. We demonstrate that, in many cases, the integrated problem is no more complicated than a least-squares problem, which can be readily solved through standard quadratic programming. We consider the MVO problem under several constraint settings using both univariate and multivariate regression for the prediction model. Under special circumstances, discussed below, the IPO program reduces to a closed-form analytical solution and thus obviates the need for a more computationally demanding iterative optimization algorithm. Furthermore, unlike many machine-learning applications, the required action of the inverse of the Hessian of the IPO solution is computationally tractable in most practical settings. This is discussed in more detail in Section 5.2.2.

Numerical experiments are performed on both synthetically generated data and a universe of 24 global futures markets, with daily return data starting in March 1986 and extending through December 2020. From an experimental standpoint, our goal is to expand the understanding of IPO methods from both a computational and performance expectation perspective. Our experimental results on real asset price data provides proof of concept of the IPO approach, thus filling a notable gap in this fast moving field. In summary, our analytical and experimental contributions are as follows:

1. For the case where the MVO program is either unconstrained or contains only linear equality

constraints then the integrated program can be recast as an unconstrained quadratic program. We provide the necessary conditions for convexity and provide analytical solutions for the optimal IPO coefficients, $\boldsymbol{\theta}^*$. We provide conditions for which $\boldsymbol{\theta}^*$ is an unbiased estimator of $\boldsymbol{\theta}$ and derive the analytical expression for the variance. We demonstrate that the IPO coefficients explicitly minimize the tracking error to the unconstrained ex-post optimal MVO portfolio and provide the equivalent minimum-tracking error optimization program.

2. We conduct a simulation study based on synthetically generated data and compare the out-of-sample performance of the IPO and OLS models under varying degrees of estimation error in both the mean and covariance. We demonstrate that, in general, the IPO models produce consistently lower out-of-sample MVO costs, even when the underlying generating process for asset returns is linear in the feature variables. Specifically, we show that, under such circumstances, and even when the asset return SNRs are relatively large, the estimation error in the covariance matrix can result in OLS models that generate sub-optimal out-of-sample MVO costs. The IPO model, on the other hand, corrects for estimation error in the sample covariance and provides consistently lower out-of-sample MVO costs in both low and moderate SNR regimes.

3. We conduct a simulation study based on synthetically generated data and demonstrate the computational advantage of the analytical IPO solution over that of the current state-of-the art method, based on implicit differentiation and gradient descent. We demonstrate that the compute time required for determining the IPO coefficients analytically is comparable to that of the equivalent least-squares problem, and is on average a full order of magnitude faster than the corresponding iterative method.

4. We discuss the computational complexity of the IPO framework as a function of the number training observations and number assets in the portfolio. We conduct a simulation study based on synthetically generated data and consider linear inequality constrained MVO programs under varying degrees of model misspecification. We approximate the non-convex problem with the analytical IPO solution whereby the inequality constraints are ignored. We demonstrate the computational and performance advantage of the analytical IPO solution, which is on average 100x - 1000x times faster than the current state-of-the-art method and produces solutions with lower out-of-sample variance and, in many instances, improved MVO costs.

5. We perform a simulation study using global futures data, considering both unconstrained and

constrained MVO programs and univariate and multivariate regression models. Out-of-sample results demonstrate that the IPO model can provide lower realized cost and superior economic performance in comparison to the traditional OLS 'predict then optimize' approach.

5.2 Methodology

We follow the portfolio optimization methodology described in Chapter 2 and consider a universe of d_z assets and denote the matrix of (excess) return observations as $\mathbf{Y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(m)}] \in \mathbb{R}^{m \times d_z}$ with $m > d_z$. Let $\mathbf{V}^{(i)} \in \mathbb{R}^{d_z \times d_z}$ denote the time-varying symmetric positive definite covariance matrix of asset returns. We define the portfolio $\mathbf{z} \in \mathbb{R}^{d_z}$, where as before, the element, $\mathbf{z}_j$, denotes the proportion of total capital invested in the j^{th} asset. Recall that the mean variance cost function at time i is given as:

$$c(\mathbf{z}, \mathbf{y}^{(i)}) = -\mathbf{z}^T \mathbf{y}^{(i)} + \frac{\delta}{2} \mathbf{z}^T \mathbf{V}^{(i)} \mathbf{z} \,. \tag{5.1}$$

Therefore, for a particular return observation $\mathbf{y}^{(i)}$, the optimal portfolio weights, $\mathbf{z}^*(\mathbf{y}^{(i)})$, is given by the solution to the following convex quadratic program:

$$\mathbf{z}^*(\mathbf{y}^{(i)}) = \underset{\mathbf{z} \in \mathbb{S}}{\operatorname{argmin}} -\mathbf{z}^T \mathbf{y}^{(i)} + \frac{\delta}{2} \mathbf{z}^T \mathbf{V}^{(i)} \mathbf{z} \tag{5.2}$$

where $\delta \in \mathbb{R}_+$ is a risk-aversion parameter that controls the trade-off between minimizing variance and maximizing return.

In practice, we do not know the value $\mathbf{y}^{(i)}$ or $\mathbf{V}^{(i)}$ at decision time. In this chapter we choose to estimate the time-varying covariance matrix using a traditional weighted moving average approach and denote the covariance estimate as $\hat{\mathbf{V}}^{(i)}$. Asset returns are modelled according to the following linear model:

$$\mathbf{y}^{(i)} = \mathbf{P} \operatorname{diag}(\mathbf{x}^{(i)}) \boldsymbol{\theta} + \boldsymbol{\epsilon}^{(i)} \tag{5.3}$$

with residuals $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. Here $\operatorname{diag}(\cdot)$ denotes the usual diagonal operator and $\mathbf{P} \in \mathbb{R}^{d_y \times d_x}$ controls the regression design with each element $\mathbf{P}_{jk} \in \{0, 1\}$. In particular, we assume that each asset has its own, perhaps unique, set of feature variables. For example, if the feature variables represent price-to-earnings (P/E) and debt-to-equity (D/E) ratios for each asset under consideration, then it would be unrealistic to model a particular asset's return as a function of all available P/E and D/E ratios. Indeed, doing so would almost certainly lead to model overfit. Instead, we choose to

model asset j's return as a linear function of the P/E and D/E ratios relevant to asset j, specifically:

$$\hat{\mathbf{y}}_{\mathrm{j}}^{(i)} = \boldsymbol{\theta}_{\mathbf{a}(j)}^{T}\, \mathbf{x}_{\mathbf{a}(\mathrm{j})}{}^{(i)}, \tag{5.4}$$

where $\mathbf{a}(j)$ denotes the indices of the feature variables relevant to asset j. Therefore,

$$\mathbf{P}_{jk} = \begin{cases} 1, & \text{if } k \in \mathbf{a}(j) \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

and for observation i, the estimate of asset expected returns is given by:

$$\hat{\mathbf{y}}^{(i)} = f(\mathbf{x}^{(i)}, \boldsymbol{\theta}) = \mathbf{P}\, \mathrm{diag}(\mathbf{x}^{(i)})\, \boldsymbol{\theta}\,. \tag{5.6}$$

Finally, under the estimation hypothesis, the mean-variance cost function has the following form:

$$c(\mathbf{z}, \hat{\mathbf{y}}^{(i)}) = -\,\mathbf{z}^{T}\, \hat{\mathbf{y}}^{(i)} + \frac{\delta}{2}\, \mathbf{z}^{T}\, \hat{\mathbf{V}}^{(i)}\, \mathbf{z} = -\,\mathbf{z}^{T}\, \mathbf{P}\, \mathrm{diag}(\mathbf{x}^{(i)})\, \boldsymbol{\theta} + \frac{\delta}{2}\, \mathbf{z}^{T}\, \hat{\mathbf{V}}^{(i)}\, \mathbf{z} \tag{5.7}$$

In the IPO framework, the objective is to choose $\boldsymbol{\theta}$ in order to minimize the average realized cost induced by the optimal decisions $\{\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})\}_{i=1}^{m}$. Substituting the estimated and realized costs into program (3.9) gives the full IPO program in discrete form, presented in Program (5.8):

$$\begin{aligned} \underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \quad & \frac{1}{m}\sum_{i=1}^{m}\Big(-\,\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})^{T}\, \mathbf{y}^{(i)} + \frac{\delta}{2}\, \mathbf{z}^*(\hat{\mathbf{y}}^{(i)})^{T}\, \mathbf{V}^{(i)}\, \mathbf{z}^*(\hat{\mathbf{y}}^{(i)})\Big) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}}{\mathrm{argmin}} -\,\mathbf{z}^{T}\, \mathbf{P}\, \mathrm{diag}(\mathbf{x}^{(i)})\, \boldsymbol{\theta} + \frac{\delta}{2}\, \mathbf{z}^{T}\, \hat{\mathbf{V}}^{(i)}\, \mathbf{z} \quad \forall i = 1, ..., m, \end{aligned} \tag{5.8}$$

and as before the objective function, $\bar{c}(\boldsymbol{\theta})$, is given by:

$$\bar{c}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\Big(-\,\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})^{T}\, \mathbf{y}^{(i)} + \frac{\delta}{2}\, \mathbf{z}^*(\hat{\mathbf{y}}^{(i)})^{T}\, \mathbf{V}^{(i)}\, \mathbf{z}^*(\hat{\mathbf{y}}^{(i)})\Big). \tag{5.9}$$

Note at this point we have not described the feasible region, $\mathbb{S}$, of the MVO program. In the following subsections we discuss the general case where $\mathbb{S}$ describes a set of linear equality and inequality constraints and formalize the current state-of-the-art neural network framework. We then discuss several special cases in which an analytical solution to the MVO problem is possible and derive the relevant theory.

5.2.1 Current state-of-the-art methodology

We begin with the general case whereby the feasible region of the MVO program is defined by both linear equality and inequality constraints. Specifically we have the following MVO program:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\mathbf{z}^T \hat{\mathbf{y}}^{(i)} + \frac{\delta}{2} \mathbf{z}^T \hat{\mathbf{V}}^{(i)} \mathbf{z} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b} \\ & \mathbf{G}\,\mathbf{z} \leq \mathbf{h} \end{aligned} \tag{5.10}$$

where as before $\mathbf{A} \in \mathbb{R}^{d_{\text{eq}} \times d_{\mathbf{z}}}$, $\mathbf{b} \in \mathbb{R}^{d_{\text{eq}}}$ and $\mathbf{G} \in \mathbb{R}^{d_{\text{iq}} \times d_{\mathbf{z}}}$, $\mathbf{h} \in \mathbb{R}^{d_{\text{iq}}}$ describe the linear equality and inequality constraints, respectively. In general, there is no known analytical solution to Program (5.10) and instead the solution, $\mathbf{z}^*$, is obtained through iterative interior-point methods.

Furthermore, because of the inequality constraints, the IPO objective, $\bar{c}(\boldsymbol{\theta})$, is generally not a convex function of $\boldsymbol{\theta}$. Nonetheless, we can solve for locally optimal solutions by applying (stochastic) gradient descent methods. Specifically, we apply the methodology described in Chapter 4 and reformulate the IPO Program (5.8) as an end-to-end trainable neural network with a differentiable quadratic optimization layer.

The IPO equivalent neural network structure is depicted in Figure 5.1. In the forward pass, the input layer takes the feature variables $\mathbf{x}^{(i)}$ and passes them to a simple linear layer to produce the estimates, $\hat{\mathbf{y}}^{(i)}$. The predictions are then passed to a differentiable quadratic programming layer which, for a given input $\hat{\mathbf{y}}^{(i)}$, solves the MVO program and returns the optimal portfolio weights $\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})$. At this point, we cache the factorized left-hand side matrix at the optimal solution, which will be invoked during backpropagation. Finally, the quality of the portfolio weights are evaluated by the cost function, $c(\mathbf{z}^*(\mathbf{x}, \boldsymbol{\theta}), \mathbf{y}^{(i)})$ in the context of the true $\mathbf{y}^{(i)}$ values.

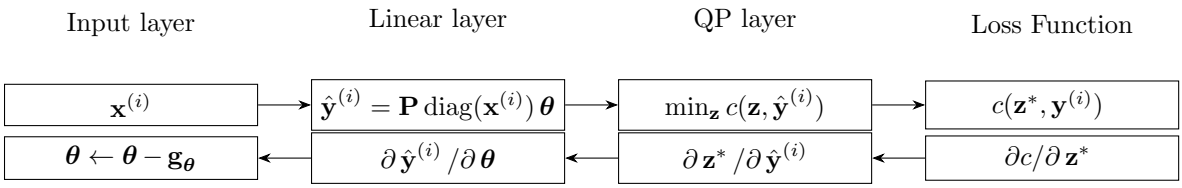


Figure 5.1: IPO program represented as an end-to-end neural network with predictive linear layer, differentiable quadratic programming layer and MVO cost loss function.

When the MVO program contains linear inequality constraints, then we search for a locally optimal solution using stochastic gradient descent (SGD). In this case, the descent direction, $\mathbf{g}_{\boldsymbol{\theta}}$, at

each iteration approximates the gradient, $\nabla_\theta \bar{c}$, and is given by :

$$\mathbf{g}_\theta = \frac{1}{m} \sum_{i \in B} \Big(\frac{\partial c}{\partial \boldsymbol{\theta}} \Big)_{|(\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}), \mathbf{y}^{(i)})} \approx \nabla_\theta \bar{c}$$

where in standard stochastic gradient descent, B represents a randomly drawn sample batch.

Lastly, Equations (4.14) in Section 4.4 allows for efficient computation of the gradients with respect to any of the relevant problem variables. While in this chapter we are solely concerned with linear prediction models for the vectors of expected returns, $\mathbf{y}^{(i)}$, we note that the IPO framework can easily support integrated optimization of prediction model parameters (linear or otherwise) for the remaining problem variables. We refer the reader to Section 4.4 for more details.

5.2.2 Special case 1: $\mathbb{S} = \mathbb{R}^{d_z}$

We consider the case where the MVO program is unconstrained ($\mathbb{S} = \mathbb{R}^{d_z}$) and therefore an analytical solution is given by Equation (5.11).

$$\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) = \frac{1}{\delta} \hat{\mathbf{V}}^{-1^{(i)}} \hat{\mathbf{y}}^{(i)} = \frac{1}{\delta} \hat{\mathbf{V}}^{-1^{(i)}} \mathbf{P} \operatorname{diag}(\mathbf{x}^{(i)}) \boldsymbol{\theta} \tag{5.11}$$

Proposition 2. *Let $\mathbb{S} = \mathbb{R}^{d_z}$ and $\Theta = \mathbb{R}^{d_\theta}$. We define*

$$\mathbf{d}(\mathbf{x}, \mathbf{y}) = \frac{1}{m\delta} \sum_{i=1}^{m} \Big(\operatorname{diag}(\mathbf{x}^{(i)}) \mathbf{P}^T \hat{\mathbf{V}}^{-1^{(i)}} \mathbf{y}^{(i)} \Big) \tag{5.12}$$

and

$$\mathbf{H}(\mathbf{x}) = \frac{1}{m\delta} \sum_{i=1}^{m} \Big(\operatorname{diag}(\mathbf{x}^{(i)}) \mathbf{P}^T \hat{\mathbf{V}}^{-1^{(i)}} \mathbf{V}^{(i)} \hat{\mathbf{V}}^{-1^{(i)}} \mathbf{P} \operatorname{diag}(\mathbf{x}^{(i)}) \Big). \tag{5.13}$$

Then the IPO program (5.8) *is an unconstrained quadratic program (QP) given by:*

$$\underset{\boldsymbol{\theta} \in \Theta}{\text{minimize}} \; \frac{1}{2} \boldsymbol{\theta}^T \mathbf{H}(\mathbf{x}) \boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{d}(\mathbf{x}, \mathbf{y}). \tag{5.14}$$

Furthermore, if there exists an $\mathbf{x}^{(i)}$ such that $\mathbf{x}_\mathrm{j}{}^{(i)} \neq 0 \quad \forall j \in 1, ..., d_x$ then $\mathbf{H}(\mathbf{x}) \succ 0$ and therefore program (5.14) *is an unconstrained convex quadratic program with unique minimum:*

$$\boldsymbol{\theta}^* = \mathbf{H}(\mathbf{x})^{-1} \mathbf{d}(\mathbf{x}, \mathbf{y}). \tag{5.15}$$

All proofs are provided in Appendix A. We make a few important observations. The first, is

that for the realistic case where there exists an $\mathbf{x}^{(i)}$ such that each $\mathbf{x}_\mathrm{j}{}^{(i)}$ are not exactly zero, then the optimal IPO regression coefficients, $\boldsymbol{\theta}^*$, are unique. Furthermore, we observe that the solution is independent of the risk-aversion parameter. This is intuitive, as when the MVO program is unconstrained, then the risk-aversion parameter simply controls the scale of the resulting portfolio.

We note that the solution presented in Equation (5.15) requires the action of the inverse of the Hessian: $\mathbf{H}(\mathbf{x})$. In many applications of machine learning, such as computer vision or statistical meta-modelling, it is difficult, if not impossible, to solve the inverse problem without customized algorithms or prior knowledge of the data (see for example Jones and Taylor [77], Ranjan et al. [107], Ongie et al. [105]). In many cases, the dimension of the relevant Hessian is either too large for both forward-mapping and inversion in reasonable compute time or is computationally unstable due to near-singularity. In our IPO framework, we fortunately do not encounter these technical difficulties surrounding the action of the inverse. In most practical settings, the dimension of the Hessian matrix, is on the order of 10 or 100, whereas the number of observations, m, is on the order of 1000 or 10000. The Hessian is therefore likely to be computationally stable and the action of the inverse is computationally tractable. This is validated numerically in Section 5.3 and we demonstrate the computational advantage of the analytical solution over the iterative descent method.

Furthermore, while outside of the scope of this chapter, we note that under the QP formulation (5.14), it is trivial to incorporate both regularization and constraints on $\boldsymbol{\theta}$. This is demonstrated by Program (5.16):

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\,\boldsymbol{\theta}^T\,\mathbf{H}(\mathbf{x})\,\boldsymbol{\theta} - \boldsymbol{\theta}^T\,\mathbf{d}(\mathbf{x},\mathbf{y}) + \Omega(|\,\boldsymbol{\theta}\,|) \\ \text{subject to} \quad & \mathbf{A}_{\boldsymbol{\theta}}\,\boldsymbol{\theta} = \mathbf{b}_{\boldsymbol{\theta}} \\ & \mathbf{G}_{\boldsymbol{\theta}}\,\boldsymbol{\theta} \leq \mathbf{h}_{\boldsymbol{\theta}} \end{aligned} \tag{5.16}$$

where $\Omega\colon \mathbb{R}^{d_\theta} \to \mathbb{R}$ is a convex regularization function. In most cases, Program (5.16) can be solved to global optimality by standard quadratic programming techniques, whereas the incorporation of regularization and constraints in the current state-of-the-art solution is structurally more challenging.

We now discuss the properties of sampling distribution of the IPO parameter estimate, $\boldsymbol{\theta}^*$, and derive an estimate of the variance, $\mathrm{Var}\,(\boldsymbol{\theta}^*)$. Recall, from Equation (5.3) we have:

$$\mathbf{y}^{(i)} \sim \mathcal{N}(\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}, \Sigma).$$

Proposition 3. *Let*

$$\mathbf{d_u}(\mathbf{x}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\operatorname{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\Big), \tag{5.17}$$

then the optimal IPO estimate, $\boldsymbol{\theta}^*$, is a biased estimate of $\boldsymbol{\theta}$ with bias $\mathbf{H}(\mathbf{x})^{-1}\,\mathbf{d_u}(\mathbf{x})$.

Corollary 2. *Let $\boldsymbol{\theta}^*_{\mathbf{u}} = \mathbf{d_u}(\mathbf{x})^{-1}\,\mathbf{H}(\mathbf{x})\,\boldsymbol{\theta}^*$. Then $\boldsymbol{\theta}^*_{\mathbf{u}}$ is an unbiased estimator of $\boldsymbol{\theta}$.*

Corollary 3. *Let $\hat{\mathbf{V}}^{(i)} = \mathbf{V}^{(i)}\ \forall i \in \{1, ..., m\}$. Then $\boldsymbol{\theta}^*$ is an unbiased estimator of $\boldsymbol{\theta}$.*

We observe from Proposition 3 that differences, or estimation errors, between $\hat{\mathbf{V}}^{(i)}$ and $\mathbf{V}^{(i)}$, make $\boldsymbol{\theta}^*$ a biased estimator in $\boldsymbol{\theta}$. In particular, the bias can be corrected by left multiplication of $\boldsymbol{\theta}^*$ by $\mathbf{d_u}(\mathbf{x})^{-1}\,\mathbf{H}(\mathbf{x})$. This observation leads to Corollary 3, which shows that when the estimation error in the covariance is zero then $\boldsymbol{\theta}^*$ is an unbiased estimator of $\boldsymbol{\theta}$. Moreover, unlike the OLS estimate, $\hat{\boldsymbol{\theta}}$, the IPO estimate, $\boldsymbol{\theta}^*$, will correct for estimation error in the sample covariance in the (likely) event that the estimation error is nonzero. This is validated numerically and discussed in more detail in in Section 5.3.

Proposition 4. *Let $\{\mathbf{y}^{(i)}\}_{i=1}^m$ be independent random variables with $\mathbf{y}^{(i)} \sim \mathcal{N}(\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}, \Sigma)$. Let $\hat{\Sigma}$ be an unbiased estimate of the sample covariance of residuals, given by:*

$$\hat{\Sigma} = \frac{1}{m-1}\sum_{i=1}^{m}\Big(\mathbf{y}^{(i)} - \mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}\Big)^2. \tag{5.18}$$

Let

$$\mathbf{M} = \frac{1}{\delta^2 m^2}\Big(\operatorname{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\hat{\Sigma}\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\Big), \tag{5.19}$$

then the variance, $\operatorname{Var}(\boldsymbol{\theta}^*)$*, is given by:*

$$\operatorname{Var}(\boldsymbol{\theta}^*) = \mathbf{H}(\mathbf{x})^{-1}\,\mathbf{M}\,\mathbf{H}(\mathbf{x})^{-1} \tag{5.20}$$

We conclude this section by providing an alternative, and perhaps more intuitive expression of the optimal IPO coefficients derived from portfolio tracking-error optimization. Let $\|\cdot\|_{\mathbf{V}}$ denote the elliptic norm with respect to the symmetric positive definite matrix $\mathbf{V}$, defined as:

$$\|\mathbf{w}\|_{\mathbf{V}} = \sqrt{\mathbf{w}^T\,\mathbf{V}\,\mathbf{w}}. \tag{5.21}$$

More specifically, $\|\mathbf{z}^{(1)} - \mathbf{z}^{(2)}\|^2_{\mathbf{V}}$ measures the tracking-error between two portfolio weights with respect to the covariance $\mathbf{V}$.

Proposition 5. *Let* $\mathbf{z}^*(\mathbf{y}^{(i)})$ *and* $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ *be as defined in Equation* (5.2) *and Equation* (5.11)*, respectively. Then the optimal IPO coefficients,* $\boldsymbol{\theta}^*$*, minimizes the average tracking error between* $\mathbf{z}^*(\mathbf{y}^{(i)})$ *and* $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ *with respect to the realized covariance* $\mathbf{V}^{(i)}$*:*

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} \frac{1}{2m}\sum_{i=1}^{m}\|\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) - \mathbf{z}^*(\mathbf{y}^{(i)})\|^2_{\mathbf{V}^{(i)}} \tag{5.22}$$

Indeed, Proposition 5 states that the IPO coefficients $\boldsymbol{\theta}^*$ minimizes the average tracking error between the estimated optimal weights, $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ and the ex-post optimal weight $\mathbf{z}^*(\mathbf{y}^{(i)})$.

5.2.3 Special case 2: $\mathbb{S} = \{\mathbf{A}\,\mathbf{z} = \mathbf{b}\}$

We now consider the case where the MVO program is constrained by a set of linear equality constraints:

$$\mathbb{S} = \{\mathbf{A}\,\mathbf{z} = \mathbf{b}\},$$

where $\mathbf{A} \in \mathbb{R}^{d_{\text{eq}}\times d_z}$ and $\mathbf{b} \in \mathbb{R}^{d_{\text{eq}}}$. We assume the non-trivial case where $\mathbf{A}$ is not full rank. Let the columns of $\mathbf{F}$ form a basis for the nullspace of $\mathbf{A}$ defined as:

$$\text{Null}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{R}^{d_z} \,|\, \mathbf{A}\,\mathbf{z} = 0\}.$$

Let $\mathbf{z}_0$ be a particular element of $\mathbb{S}$. It follows that $\forall\ \mathbf{w} \in \mathbb{R}^{d_\mathbf{z}-d_n}$ then $\mathbf{z} = \mathbf{F}\,\mathbf{w} + \mathbf{z}_0$ is also an element of $\mathbb{S}$, where $d_n = \text{nullity}(\mathbf{A})$. We follow Boyd and Vandenberghe [20] and recast the MVO program as an unconstrained convex quadratic program:

$$\min_{\mathbf{w}} c(\mathbf{F}\,\mathbf{w} + \mathbf{z}_0, \hat{\mathbf{y}}^{(i)}), \tag{5.23}$$

with unique global minimum:

$$\mathbf{w}^*(\hat{\mathbf{y}}^{(i)}) = \frac{1}{\delta}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\left(\hat{\mathbf{y}}^{(i)} - \delta\,\hat{\mathbf{V}}^{(i)}\,\mathbf{z}_0\right) \tag{5.24}$$

The solution to the MVO Program (5.2) is then given by:

$$\begin{aligned}\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) &= \frac{1}{\delta}\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T(\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta} - \delta\,\hat{\mathbf{V}}^{(i)}\,\mathbf{z}_0) + \mathbf{z}_0 \\ &= \frac{1}{\delta}\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta} + (\mathbf{I} - \mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)})\,\mathbf{z}_0\end{aligned} \tag{5.25}$$

Proposition 6. *Let* $\mathbb{S} = \{\mathbf{A}\,\mathbf{z} = \mathbf{b}\}$ *and* $\Theta = \mathbb{R}^{d_\theta}$*. Define*

$$\mathbf{d}_{eq}(\mathbf{x},\mathbf{y}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\operatorname{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T(\mathbf{y}^{(i)}-\mathbf{V}^{(i)}(\mathbf{I}-\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)})\,\mathbf{z}_0)\Big) \tag{5.26}$$

and

$$\mathbf{H}_{eq}(\mathbf{x}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\operatorname{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{V}^{(i)}\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\Big). \tag{5.27}$$

Then the IPO program (5.8) *is an unconstrained quadratic program given by:*

$$\underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}}\;\frac{1}{2}\,\boldsymbol{\theta}^T\,\mathbf{H}_{eq}(\mathbf{x})\,\boldsymbol{\theta}-\boldsymbol{\theta}^T\,\mathbf{d}_{eq}(\mathbf{x},\mathbf{y}). \tag{5.28}$$

Furthermore, if there exists an $\mathbf{x}^{(i)}$ *such that* $\mathbf{x}_{\mathrm{j}}{}^{(i)} \neq 0 \quad \forall j \in 1,...,d_x$ *then* $\mathbf{H}_{eq}(\mathbf{x}) \succ 0$ *and therefore program* (5.28) *is an unconstrained convex quadratic program with unique minimum:*

$$\boldsymbol{\theta}^*_{eq} = \mathbf{H}_{eq}(\mathbf{x})^{-1}\,\mathbf{d}_{eq}(\mathbf{x},\mathbf{y}). \tag{5.29}$$

As before we briefly discuss the properties of the sampling distribution of the equality constrained IPO parameter estimate, $\boldsymbol{\theta}^*_{\mathrm{eq}}$, and derive an estimate of the variance, $\operatorname{Var}(\boldsymbol{\theta}^*_{\mathrm{eq}})$.

Proposition 7. *Let*

$$\mathbf{d}_{\mathbf{e}}(\mathbf{x}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\operatorname{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\Big), \tag{5.30}$$

then the optimal IPO estimate, $\boldsymbol{\theta}^*_{eq}$*, is a biased estimate of* $\boldsymbol{\theta}$ *with bias* $\mathbf{H}_{eq}(\mathbf{x})^{-1}\,\mathbf{d}_{\mathbf{e}}(\mathbf{x})$.

Corollary 4. *Let* $\hat{\mathbf{V}}^{(i)} = \mathbf{V}^{(i)}\,\forall i \in \{1,...,m\}$. *Then* $\boldsymbol{\theta}^*_{eq}$ *is an unbiased estimator of* $\boldsymbol{\theta}$.

Again we observe from Proposition 7 that in general $\boldsymbol{\theta}^*_{\mathrm{eq}}$ a biased estimator of $\boldsymbol{\theta}$. In particular, the bias can be corrected by left multiplication of $\boldsymbol{\theta}^*_{\mathrm{eq}}$ by $\mathbf{d}_{\mathbf{e}}(\mathbf{x})^{-1}\,\mathbf{H}_{\mathrm{eq}}(\mathbf{x})$. Furthermore when the estimation error in the covariance is zero then $\boldsymbol{\theta}^*_{\mathrm{eq}}$ is an unbiased estimator of $\boldsymbol{\theta}$.

Proposition 8. *Let* $\{\mathbf{y}^{(i)}\}_{i=1}^m$ *be independent random variables with* $\mathbf{y}^{(i)} \sim \mathcal{N}(\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta},\Sigma)$. *Let*

$$\mathbf{M}_{eq} = \frac{1}{\delta^2 m^2}\sum_{i=1}^{m}\Big(\operatorname{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\hat{\Sigma}\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\operatorname{diag}(\mathbf{x}^{(i)})\Big). \tag{5.31}$$

then the variance, $\text{Var}(\boldsymbol{\theta}^*_{eq})$*, is given by:*

$$\text{Var}(\boldsymbol{\theta}^*_{eq}) = \mathbf{H}_{eq}(\mathbf{x})^{-1}\,\mathbf{M}_{eq}\,\mathbf{H}_{eq}(\mathbf{x})^{-1} \tag{5.32}$$

As before, we conclude this subsection with the following proposition that states that the IPO coefficients $\boldsymbol{\theta}^*$ minimizes the average tracking error between the estimated optimal weights, $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ and the ex-post optimal weight $\mathbf{z}^*(\mathbf{y}^{(i)})$.

Proposition 9. *Let* $\mathbf{z}^*(\mathbf{y}^{(i)})$ *and* $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ *be as defined in Equation* (5.2) *and Equation* (5.25)*, respectively. Then the optimal IPO coefficients,* $\boldsymbol{\theta}^*_{eq}$*, minimizes the average tracking error between* $\mathbf{z}^*(\mathbf{y}^{(i)})$ *and* $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ *with respect to the realized covariance* $\mathbf{V}^{(i)}$*:*

$$\begin{aligned} \boldsymbol{\theta}^*_{\text{eq}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \quad & \frac{1}{2m}\sum_{i=1}^{m} \|\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) - \mathbf{z}^*(\mathbf{y}^{(i)})\|^2_{\mathbf{V}^{(i)}} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) = \mathbf{b}, \quad \mathbf{A}\,\mathbf{z}^*(\mathbf{y}^{(i)}) = \mathbf{b}\,. \end{aligned} \tag{5.33}$$

5.3 Simulated experiments

5.3.1 Simulation 1: estimation error in $\hat{\mathbf{V}}$

Elmachtoub and Grigas [47] consider the integration of predictive forecasting with downstream optimization problems that have linear cost functions. Their simulated experiments demonstrate that the benefit of the 'smart predict, then optimize' (SPO) framework increases as the amount of model misspecification increases. Specifically, model misspecification is introduced by synthetically generating cost vectors that are polynomial functions of the simulated feature data and modelling the relationship as though it is linear. In particular, they demonstrate that a linear forecasting model trained with SPO can outperform traditional prediction models, such as OLS and random forest, and the amount of outperformance increases as the degree of nonlinearity in the ground truth increases.

Here, we demonstrate that, for a mean-variance decision program, the IPO model can provide lower out-of-sample MVO costs in comparison to a traditional OLS-based 'predict, then optimize' model, even when the underlying ground truth is *linear* in the feature variables. In particular, we demonstrate that the OLS model is vulnerable to estimation error in $\hat{\mathbf{V}}^{(i)}$, resulting in sub-optimal decision-making with out-of-sample MVO costs increasing as estimation error in $\hat{\mathbf{V}}^{(i)}$ increases. The IPO model, on the other hand, corrects for estimation error in the covariance matrix. The

simulated experiment below demonstrates that the IPO model consistently outperforms the OLS model in terms of minimizing the out-of-sample MVO cost. Moreover, the outperformance is shown to be consistent over a wide range of signal-to-noise ratios (SNRs) and asset correlation assumptions commonly observed in financial forecasting. In general we observe that the benefit of the IPO model increases as the estimation error in $\hat{\mathbf{V}}^{(i)}$ increases, even when the underlying ground truth is linear in the feature variables.

We follow an experimental design similar to Hastie et al. [69]. Asset returns are assumed to be normally distributed, $\mathbf{y}^{(i)} \sim \mathcal{N}(\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}_0, \mathbf{V})$ where $\mathbf{V} \in \mathbb{R}^{d_z \times d_z}$ has entry (j,k) equal to $\sigma^2 \rho^{|j-k|}$, and $\sigma = 0.0125$ (20% annualized). Asset mean returns are modelled according to univariate model of the form:

$$\mathbf{y}^{(i)} = \text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}_0 + \tau\,\boldsymbol{\epsilon}^{(i)},$$

where feature data, $\mathbf{x}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_x})$, and residuals, $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$. The scalar value τ controls the SNR level, where $\text{SNR} = \text{Var}(f(\mathbf{x}, \boldsymbol{\theta}_0))/\,\text{Var}(\boldsymbol{\epsilon})$. We consider asset correlation values in the range of: $\rho \in \{0, 0.25, 0.5, 0.75\}$, and SNR values: $\text{SNR} \in \{0.001, 0.002, 0.003, 0.004, 0.005, 0.01, 0.05, 0.10\}$. Note that it may appear that these SNR values are extremely low. However, in most applications of asset return forecasting, the SNRs are typically found to be much less than 1%. Indeed, a moderate sized universe (25 assets) with each asset having SNRs of 1% can generate annualized Sharpe ratios in the low double digits - which is extremely rare - and SNRs of 10% are extremely unlikely at a daily trading frequency.

We introduce estimation error in $\hat{\mathbf{V}}^{(i)}$ by varying the sample size, $s = \text{res} * d_z$, used for estimation. We set the number of assets, $d_z = 10$, and consider resolutions, $\text{res} \in \{5, 10, 20\}$, thus giving covariance sample sizes of $s \in \{50, 100, 200\}$. In all experiments we set the risk aversion parameter $\delta = 1$.

The simulation process can be described as follows:

1. Generate the ground truth coefficients: $\boldsymbol{\theta}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_\theta})$.
2. Generate feature variables: $\mathbf{x}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_x})$.
3. Generate 2000 return observations: $\mathbf{y}^{(i)} \sim \mathcal{N}(\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}, \tau\,\boldsymbol{\epsilon}^{(i)})$, where τ is chosen to meet the desired SNR.
4. Divide the sample data into two equally sized disjoint data sets: in-sample and out-of-sample.
5. Generate estimates $\hat{\mathbf{V}}^{(i)}$ using the chosen sample size, s.

6. Estimate the optimal OLS and IPO coefficients on the in-sample data.

7. Generate the optimal out-of-sample MVO decisions, $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$, using the covariance estimates $\hat{\mathbf{V}}^{(i)}$ and corresponding estimated regression coefficients for predicting $\hat{\mathbf{y}}^{(i)}$.

8. Evaluate several performance metrics (described below) on the out-of-sample data.

9. Repeat steps 1-8 a total of 100 times and average the results.

Performance metrics: Let $\boldsymbol{\theta}$ denote an estimated (OLS or IPO) regression coefficient. Let $\mathbf{V}$ denote the true asset covariance and let $\{\mathbf{y}^{(i)}\}_{i=1}^m$ denote the realized return observations.

- **MVO Cost**: Let $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ be as defined in Equation (5.2). The out-of-sample MVO cost is then given by:

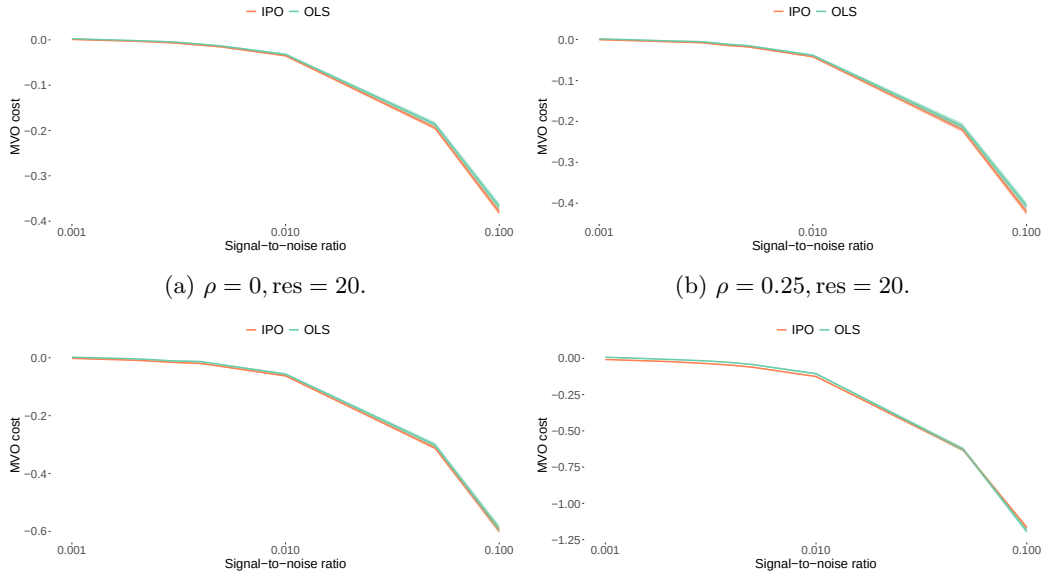
$$c(\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}), \mathbf{y}^{(i)}) = -\,\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})^T\,\mathbf{y}^{(i)} + \frac{\delta}{2}\,\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})^T\,\mathbf{V}\,\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$$

- **Proportion of variance explained:** a measure of return prediction accuracy on defined as:

$$\mathrm{PVE}(\boldsymbol{\theta}) = 1 - \mathbb{E}[(\mathbf{y}^{(i)} - \mathrm{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta})^2]/\,\mathrm{Var}\,(\mathbf{y}^{(i)}).$$

We consider the case where the MVO program contains equality constraints. In particular, we enforce that the sum of the weights must be equal to one: $\mathbb{S} = \{\mathbf{z}^T\,\mathbf{1} = 1\}$. Figures 5.2 and 5.3 report the average and 95%-ile range of the out-of-sample MVO costs and PVE values, respectively, as a function of the SNR. Here, the covariance resolution is set to 20 and therefore the expected estimation error in $\hat{\mathbf{V}}^{(i)}$ is relatively low. As a result, we observe that the difference in both out-of-sample MVO cost and PVE is negligible, with the IPO model producing marginally lower MVO costs and the OLS model producing marginally higher PVE, as expected. Observe that even in the most optimistic case where estimation error in $\hat{\mathbf{V}}^{(i)}$ is low and the ground truth relationship is linear, there is no adverse repercussions in using the IPO model. Furthermore, in order to effectively eliminate estimation error we require a covariance resolution on the order of 20; which in practical terms implies that for a 100 asset portfolio we require a sample size of 2000 return observations. In many forecasting applications a sample size of this magnitude would be impractical and would potentially interfere with the observed time-varying dependency of asset volatilities and correlations [51, 16, 18, 17, 19]. Furthermore, as estimation error increases, we observe that for the majority of relevant SNRs, the IPO model produces a lower realized out-of-sample MVO cost. In particular

Figures 5.4 and 5.6 demonstrate a statistically significant reduction in out-of-sample MVO costs as the covariance resolution decreases to 10 and 5, respectively. Interestingly, Figures 5.5 and 5.7 demonstrate that the IPO model produces lower realized MVO costs, despite providing lower average prediction accuracy, as measured by PVE. Note that this finding is consistent with the results presented in [47]. Finally, we observe in Figures 5.4 and 5.6 that the benefit of the IPO model is greatest when the ground truth correlation values, ρ, are closest to zero. Indeed this is intuitive as the extent of covariance estimation error in both magnitude and sign is largest when correlation values approach zero.



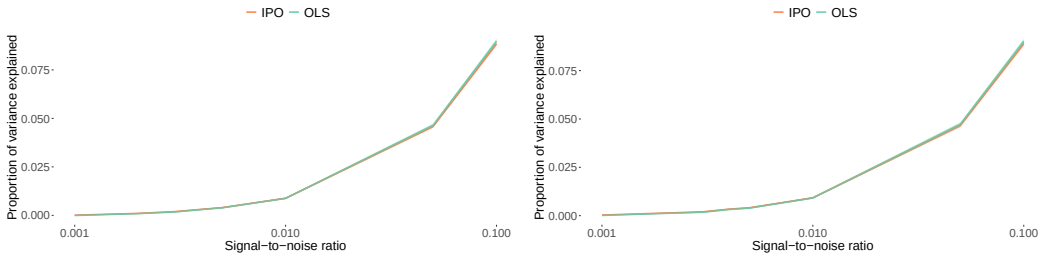
(a) $\rho = 0, \text{res} = 20$.

(b) $\rho = 0.25, \text{res} = 20$.

(c) $\rho = 0.50, \text{res} = 20$.

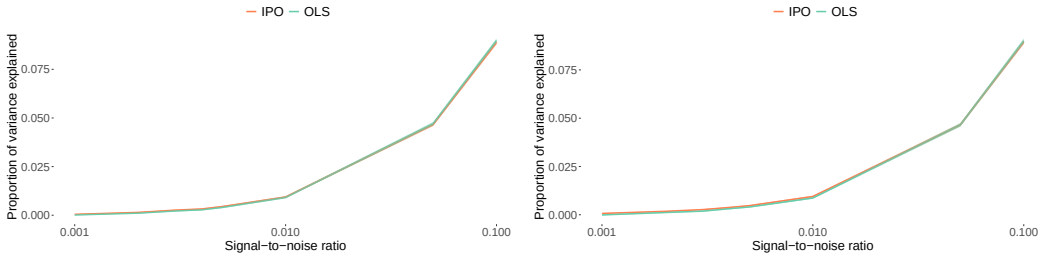
(d) $\rho = 0.75, \text{res} = 20$.

Figure 5.2: Out-of-sample MVO cost for IPO and OLS as of function of return signal-to-noise ratios.



(a) $\rho = 0, \text{res} = 20$.

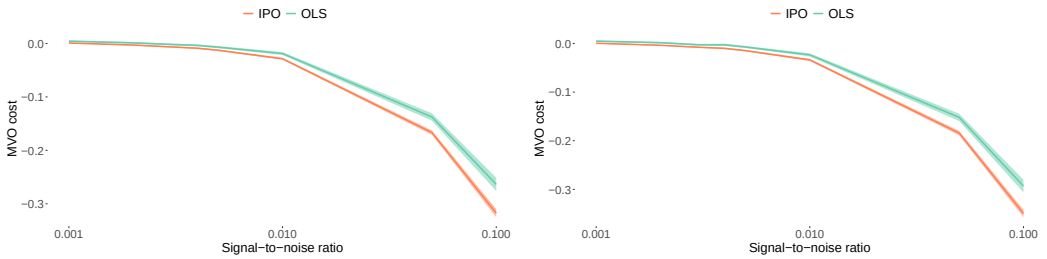
(b) $\rho = 0.25, \text{res} = 20$.



(c) $\rho = 0.50, \text{res} = 20$.

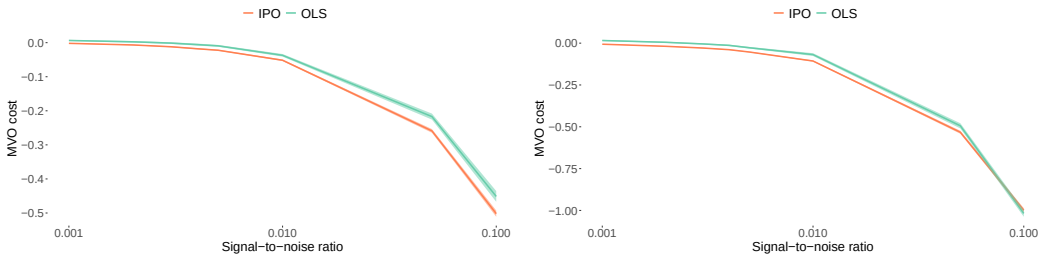
(d) $\rho = 0.75, \text{res} = 20$.

Figure 5.3: Out-of-sample PVE for IPO and OLS as of function of return signal-to-noise ratios.



(a) $\rho = 0, \text{res} = 10$.

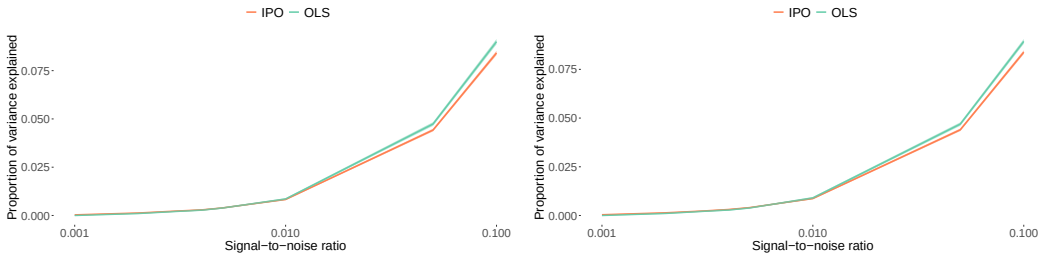
(b) $\rho = 0.25, \text{res} = 10$.



(c) $\rho = 0.50, \text{res} = 10$.

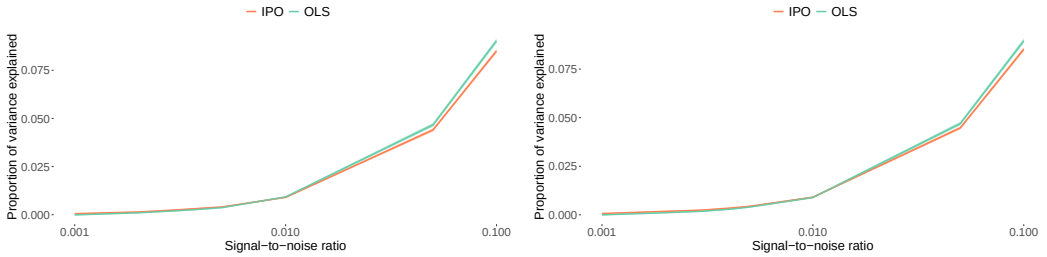
(d) $\rho = 0.75, \text{res} = 10$.

Figure 5.4: Out-of-sample MVO cost for IPO and OLS as of function of return signal-to-noise ratios.



(a) $\rho = 0, \text{res} = 10$.

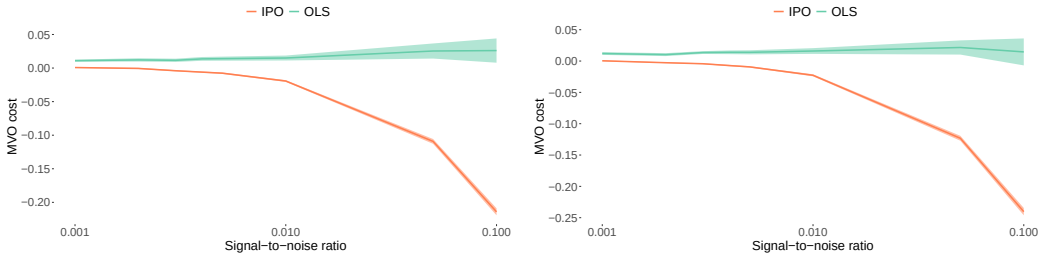
(b) $\rho = 0.25, \text{res} = 10$.



(c) $\rho = 0.50, \text{res} = 10$.

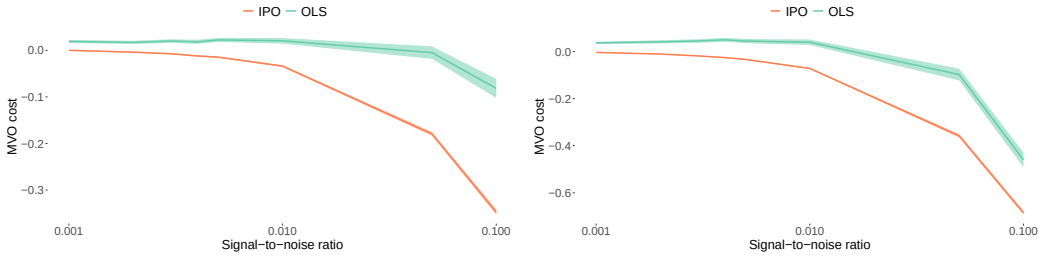
(d) $\rho = 0.75, \text{res} = 10$.

Figure 5.5: Out-of-sample PVE for IPO and OLS as of function of return signal-to-noise ratios.



(a) $\rho = 0, \text{res} = 5$.

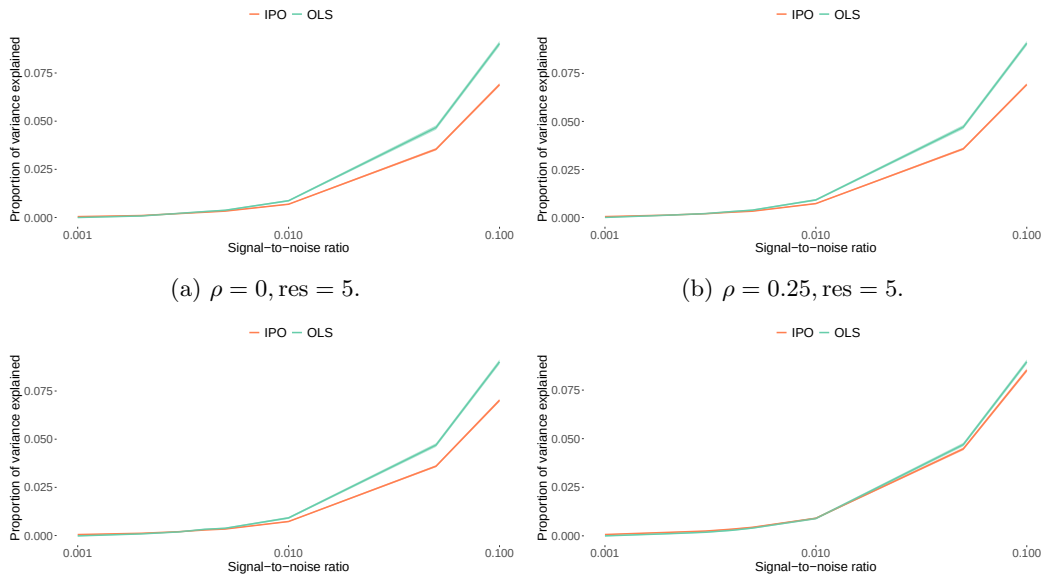
(b) $\rho = 0.25, \text{res} = 5$.



(c) $\rho = 0.50, \text{res} = 5$.

(d) $\rho = 0.75, \text{res} = 5$.

Figure 5.6: Out-of-sample MVO cost for IPO and OLS as of function of return signal-to-noise ratios.



(a) $\rho = 0, \text{res} = 5$.

(b) $\rho = 0.25, \text{res} = 5$.

(c) $\rho = 0.50, \text{res} = 5$.

(d) $\rho = 0.75, \text{res} = 5$.

Figure 5.7: Out-of-sample PVE cost for IPO and OLS as of function of return signal-to-noise ratios.

5.3.2 Simulation 2: computational efficiency

Here, we compare the computational efficiency of the analytical IPO solution with the current state-of-the-art method based on implicit differentiation and iterative gradient descent, from here on denoted as IPO-GRAD. Note that the IPO-GRAD implementation is optimized such that the matrix factorization (Equation (4.13)), required to compute the gradient, is performed once at the initialization of the algorithm. The IPO-GRAD coefficients are initialized by drawing from the standard normal distribution and the algorithm terminates when $\|\partial \bar{c}/\partial \boldsymbol{\theta}\| < 10^{-6}$.

We generate synthetic asset returns, following the procedure outlined in Section 5.3.1, with $\rho = 0$, SNR $= 0.005$ and varying the number of assets, $d_z \in \{25, 50, 100, 250\}$. Each asset is assumed to have 3 unique features, and therefore $d_\theta = 3d_z$. Tables 5.1 and 5.2, report the time, in seconds, taken by each method to compute the optimal regression coefficients for the unconstrained and equality constrained cases, respectively. For the IPO-GRAD method we also report the number of iterations of gradient descent. For each portfolio size, we report the average and 95%-ile range over 100 instances of simulated data. Observe that for problems with 100 or fewer assets, the computation time required to compute the optimal IPO coefficients analytically is comparable to the computation time required to compute the optimal OLS coefficients. In contrast, the IPO-GRAD method typically requires over 100 iterations of gradient descent and is anywhere from 10x - 1000x slower than the corresponding IPO method. We note that for problems of larger scale, the analytical

IPO solution remains tractable and is on average 6x faster than the IPO-GRAD method.

No. Assets	OLS	IPO	IPO-GRAD	Iterations
25	0.029	0.071	4.333	178
	(0.028,0.032)	(0.07,0.08)	(3.966,5.076)	(164,210)
50	0.247	0.429	6.557	186
	(0.209,0.253)	(0.342,0.447)	(6.032,7.278)	(173,207)
100	0.545	1.7	17.642	200
	(0.491,0.638)	(1.495,1.837)	(16.03,21.301)	(183,247)
250	2.89	17.961	123.975	208.5
	(2.75,3.335)	(17.546,18.092)	(114.008,165.094)	(193,279)

Table 5.1: Time in seconds for computing the optimal OLS, IPO and IPO-GRAD coefficients for an unconstrained MVO problem. Results are averaged over 100 instances of simulated data.

No. Assets	OLS	IPO	IPO-GRAD	Iterations
25	0.029	0.088	4.664	176
	(0.028,0.032)	(0.085,0.094)	(4.333,5.587)	(163,211)
50	0.247	0.473	7.389	188
	(0.171,0.259)	(0.383,0.543)	(6.696,8.227)	(172,208)
100	0.549	2.025	19.711	200
	(0.492,0.669)	(1.855,2.161)	(18.034,23.449)	(183,241)
250	2.815	22.378	129.348	208
	(2.71,3.315)	(21.8,22.511)	(119.684,174.607)	(193,280)

Table 5.2: Time in seconds for computing the optimal OLS, IPO and IPO-GRAD coefficients for an equality constrained MVO problem. Results are averaged over 100 instances of simulated data.

5.3.3 Simulation 3: inequality constrained IPO

We now consider the more general case whereby the feasible region of the MVO program is defined by inequality constraints. In general, an analytical solution to the IPO Program (5.8) in the presence of lower-level inequality constraints is not possible. Furthermore, Program (5.8) is not convex in $\boldsymbol{\theta}$. As a result, the current state-of-the-art approach (IPO-GRAD), described in Section 5.2.1, is recommended in order to obtain locally optimal solutions.

The IPO-GRAD solution, however, is challenging for several reasons. First, in contrast to the

traditional OLS approach, estimating the IPO coefficients by iterative methods can be computationally expensive; in particular as the number of assets, d_z, becomes large. Specifically, the IPO-GRAD framework requires solving, at each iteration of gradient descent, at most m constrained quadratic programs, where m is the total number of training observations. The time complexity therefore scales linearly with the number of training observations, m, and the total number of gradient descent iterations, n. Convex quadratic programs, however, are known to be solvable by interior-point methods in polynomial time, with worst-case time complexity on the order of $\mathcal{O}(d_z^3)$ [65]. Therefore, the worst-case time complexity for IPO framework is on the order of $\mathcal{O}(mnd_z^3)$. In practice, however, quadratic programs are typically solved with much fewer iterations than their worst-case bound [20]. Nonetheless, most real-world applications involve on the order of $10,000$ training observations and portfolio sizes on the order of 10 or 100. Therefore, for assets managers that construct portfolios from a very large pool of assets, estimating prediction model parameters by IPO-GRAD can be computationally burdensome. Secondly, because the inequality constrained IPO problem is not convex, we have no guarantee that any particular local solution is globally optimal. Finally, estimating $\text{Var}(\boldsymbol{\theta})$ and computing confidence intervals by standard parametric statistics is not possible.

As a heuristic, we are interested in determining the out-of-sample efficacy of the analytical IPO solutions, presented in Sections 5.2.2 - 5.2.3, applied to the inequality constrained problem. Specifically, we compute the IPO optimal coefficients analytically by dropping the inequality constraints in the lower-level MVO problem. The realized policy, $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$, however, enforces the inequality constraints in the out-of-sample evaluation period.

We generate synthetic asset returns, following the procedure outlined in Section 5.3.1, with $\rho = 0$, SNR $= 0.005$ and $d_z = 10$. Each asset is assumed to have 3 unique features ($d_\theta = 3d_z$). The inequality constraints are standard box-constraints of the form:

$$-\gamma \leq \mathbf{z}_j \leq \gamma, \quad \forall j \in \{1, ..., d_z\},$$

and we consider several values of $\gamma \in \{0.05, 0.10, 0.25, 0.50, 0.75, 1, 2, 5, 10\}$. We also vary the risk aversion parameter $\delta \in \{1, 5, 10, 25\}$. Finally, asset mean returns are generated according to linear and nonlinear polynomial models of the form:

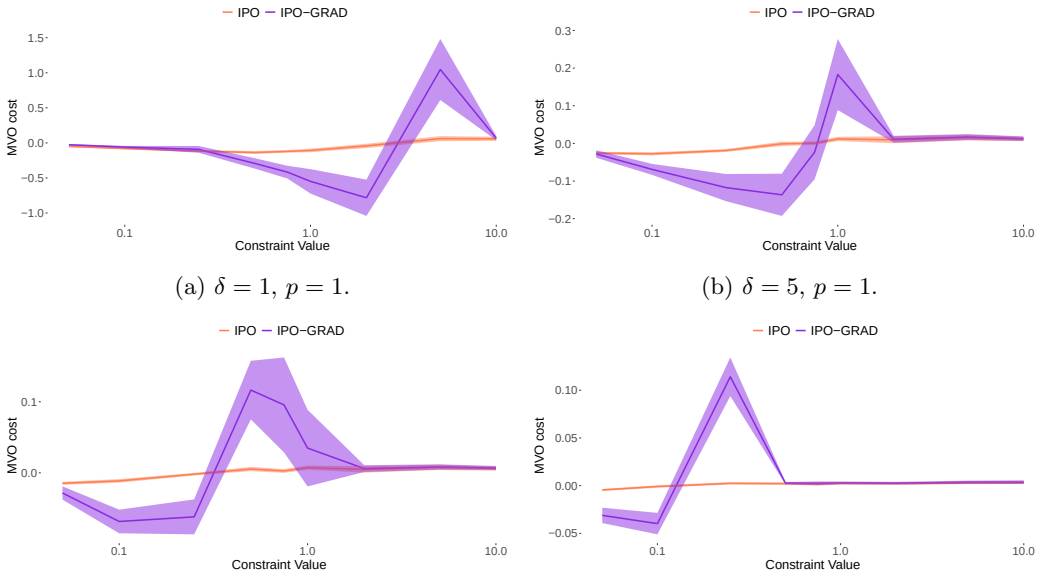
$$\mathbf{y}^{(i)} = \sum_{q=1}^{p} \mathbf{P} \operatorname{diag}(\overset{q}{\mathbf{x}^{(i)}})\, \boldsymbol{\theta}_q + \tau\, \boldsymbol{\epsilon}^{(i)},$$

with $p \in \{1, 2, 4\}$.

Figures 5.8 - 5.10 compare the out-of-sample MVO cost for the IPO and IPO-GRAD methods as of function of the box constraint value, γ, and risk-aversion parameter, δ. For each value of γ, δ and p, we report the mean and 95%-ile range over 30 instances of simulated data. First, we would expect the out-of-sample performance of the IPO and IPO-GRAD methods to converge as γ increases. Furthermore as δ, increases, the point (along γ) at which the two solutions converge will naturally decrease. This effect is purely a consequence of the inequality constraints being non-active when either γ and/or δ are sufficiently large.

In Figure 5.8 asset returns are generated according to a linear ground truth model ($p = 1$). In all cases we observe that the IPO-GRAD does provide improved out-of-sample MVO costs when γ is sufficiently small ($\gamma < 0.5$). However, for moderate and large values of γ, the IPO-GRAD method provides no improvement in out-of-sample MVO costs in comparison to the IPO method. Furthermore, in Figures 5.9 and 5.10, asset returns are generated according to a quadratic ($p = 2$) and quartic ($p = 4$) ground truth model, respectively. We observe that over practically every value of γ and δ, the IPO method provides an equivalent, if not improved, out-of-sample MVO costs in comparison to the IPO-GRAD method. We note that, while not explicitly shown here, the IPO-GRAD method produces lower in-sample (training) MVO costs over every experiment instance, and is potentially overfitting the training data. Moreover, we note that in all experiments, the variance of the out-of-sample MVO costs generated by the IPO-GRAD method is substantially larger than that of the IPO method. The lack of convexity and uniqueness of solution in the IPO-GRAD formulation, along with the likelihood of model overfit, provides a potential explanation for this effect.

Finally, Table 5.3 reports the average time (in seconds) and 95%-ile range, taken by each method to compute the optimal regression coefficients. The results are averaged over all 360 instances of simulated data. For the IPO-GRAD method we also report the number of iterations of gradient descent. We observe that the IPO-GRAD method typically requires around 60 iterations of gradient descent and is on average 100x - 1000x slower than the corresponding IPO method. Note that the computation times reported here are for a relatively small portfolio and, given the computational complexity described above, we would expect the IPO method to provide an even larger computational advantage on medium and large sized portfolios. We therefore conclude that in the presence of inequality constraints, the IPO heuristic is a compelling alternative to the more computationally expensive IPO-GRAD solution.



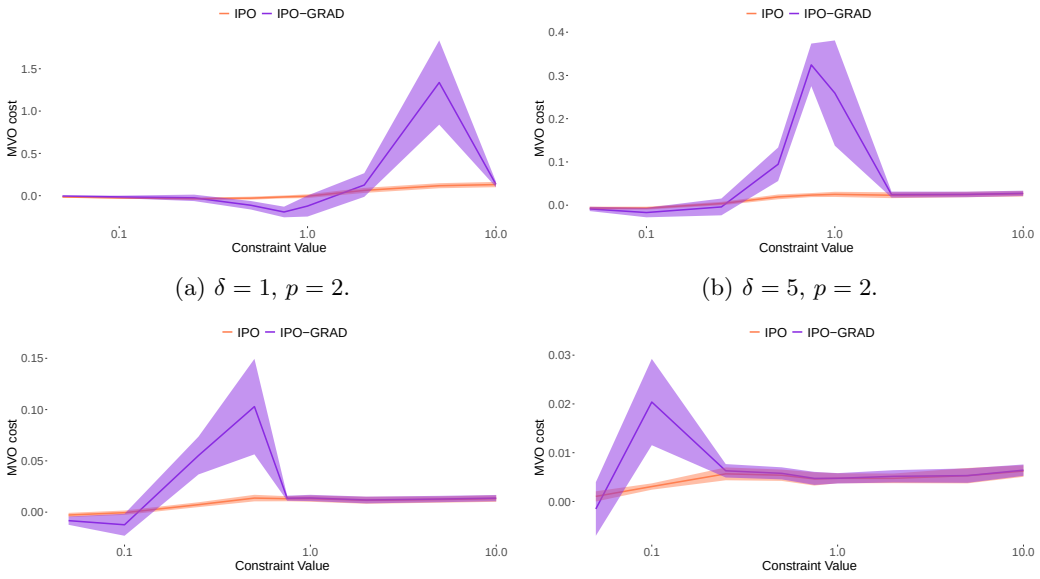
(a) $\delta = 1$, $p = 1$.

(b) $\delta = 5$, $p = 1$.

(c) $\delta = 10$, $p = 1$.

(d) $\delta = 25$, $p = 1$.

Figure 5.8: Out-of-sample MVO costs as of function of box constraint value with $p = 1$.



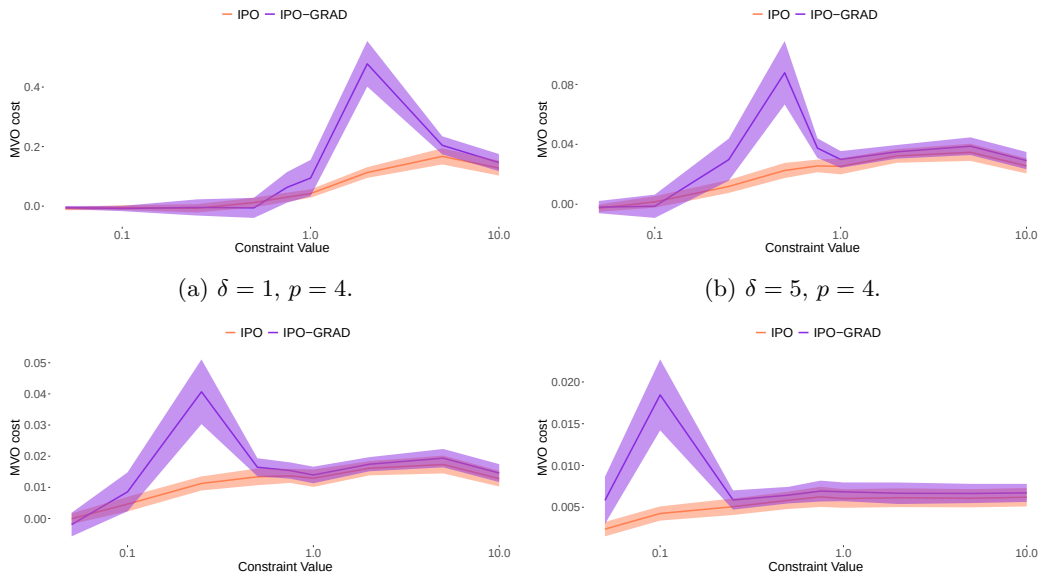
(a) $\delta = 1$, $p = 2$.

(b) $\delta = 5$, $p = 2$.

(c) $\delta = 10$, $p = 2$.

(d) $\delta = 25$, $p = 2$.

Figure 5.9: Out-of-sample MVO costs as of function of box constraint value with $p = 2$.



(a) $\delta = 1$, $p = 4$.

(b) $\delta = 5$, $p = 4$.

(c) $\delta = 10$, $p = 4$.

(d) $\delta = 25$, $p = 4$.

Figure 5.10: Out-of-sample MVO costs as of function of box constraint value with $p = 4$.

IPO	IPO-GRAD	Iterations
0.023	14.389	60
(0.022,0.024)	(6.9417,22.5718)	(29,94)

Table 5.3: Time in seconds for computing the optimal IPO and IPO-GRAD coefficients for an inequality constrained MVO problem. Results are averaged over 360 instances of simulated data.

5.4 Real data experiments

Experiment Setup:

We consider an asset universe of 24 commodity futures markets, described in Table A.1. The daily price data is given from March 1986 through December 2020, and is provided by Commodity Systems Inc. Futures contracts are rolled on, at most, a monthly basis in order to remain invested in the most liquid contract, as measured by open-interest and volume. Arithmetic returns are computed directly from the price data.

In each experiment we follow Zumbach [131] and estimate the covariance matrix using an exponential moving average with a decay rate of 0.94. We consider both univariate and multivariate prediction models. The feature, $\{\mathbf{x}^{(i)}\}$, for univariate models is the 252-day average return, or trend, for each market. The feature therefore represents a measure of the well-documented 'trend' factor, popular to many Commodity Trading Advisors (CTAs) and Hedge Funds (see for example

[8], [24],[100]). The features for multivariate models is the 252-day trend and the carry for each market. We follow Koijen et al. [82] and define the carry as the expected convenience yield, or cost, for holding that commodity, and is estimated by the percent difference in price between the two futures contracts closest to expiry.

As we will see below, the majority of the IPO and OLS regression coefficients are not statistically significant at an individual market level. Indeed this is common and well document in many applications of financial forecasting (see for example [73, 100]). The lack of statistical significance may be indicative of low signal-to-noise levels and/or forecasting model misspecification - conditions that are likely favourable for the IPO model. Furthermore, the absence of statistical significance does not prohibit the development of profitable portfolio level trading strategies and indeed we observe in Table 5.4 that the features are statistically significant at the 95%-ile level when evaluated at an aggregate level across all markets.

Feature	Coefficient	Std. Error	T-Statistic	P-Value
Carry	0.3300	0.1654	1.9953	0.0460
Trend	0.0942	0.0324	2.9101	0.0036

Table 5.4: Univariate regression coefficients and t-statistic summary aggregated across all available markets.

Each day we form the optimal portfolio weight, $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ at the close of day i, and assume execution at the following close, $i+1$. In each experiment, described below, we consider two methods for estimating asset returns:

1. **OLS:** ordinary-least squares method, with prediction coefficients, $\hat{\boldsymbol{\theta}}$.
2. **IPO:** integrated prediction and optimization method, where $\boldsymbol{\theta}^*$ is determined by the IPO optimization framework described in Section 5.2.

We consider 6 experiments:

1. Unconstrained MVO program with univariate regression.
2. Unconstrained MVO program with multivariate regression.
3. Equality constrained MVO program with univariate regression.
4. Equality constrained MVO program with multivariate regression.
5. Inequality constrained MVO program with univariate regression.

6. Inequality constrained MVO program with multivariate regression.

The equality constrained MVO programs are market-neutral: $\mathbb{S} = \{\mathbf{z}^T \mathbf{1} = 0\}$, whereas the inequality constrained MVO programs are both market-neutral and include lower bound and upper bound market constraints:

$$\mathbb{S} = \{\mathbf{z}^T \mathbf{1} = 0, -0.125 \leq \mathbf{z} \leq 0.125\}.$$

In order to provide realistic annualized volatilities in the $10\% - 20\%$ range, we fix the risk-aversion parameter to $\delta = 50$. All experiments start in January 2000 and end in December 2020. For each experiment, the first 14 years (March 1986 through December 1999) is used to perform the initial parameter estimation. Thereafter, we apply a walk-forward training and testing methodology. The optimal regressions coefficients are updated every 2 years using all available data for parameter estimation and the optimal policy, $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$, is then applied to the next out-of-sample 2 year segment. All performance is gross of trading costs and in excess of the risk-free rate.

Each model is evaluated on absolute and relative terms, with a focus on out-of-sample MVO cost and out-of-sample Sharpe ratio cost, provided by Equation (5.34).

$$c_{\text{MVO}}(\mathbf{z}, \mathbf{y}) = -\mu(\mathbf{z}, \mathbf{y}) + \frac{\delta}{2}\sigma^2(\mathbf{z}, \mathbf{y}), \quad \text{and} \quad c_{\text{SR}}(\mathbf{z}, \mathbf{y}) = -\frac{\mu(\mathbf{z}, \mathbf{y})}{\sigma(\mathbf{z}, \mathbf{y})} \tag{5.34}$$

where

$$\mu(\mathbf{z}, \mathbf{y}) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{z}^{T(i)}\, \mathbf{y}^{(i)} \quad \text{and} \quad \sigma^2(\mathbf{z}, \mathbf{y}) = \frac{1}{m}\sum_{i=1}^{m} (\mathbf{z}^{T(i)}\, \mathbf{y}^{(i)} - \mu(\mathbf{z}, \mathbf{y}))^2,$$

denote the mean and variance of realized daily returns. To quantify the magnitude and consistency of observed performance metrics, and to ensure our results are robust to potential outliers in the out-of-sample periods, scatterplots are created by bootstrapping the out-of-sample distribution using 1000 samples as follows:

1. For each $k \in \{1, 2, ..., 1000\}$, sample, without replacement, a batch, B_k, with $|B_k| = 252$ observations (1 year) from the out-of-sample period.

2. For each model, compute the average realized MVO and Sharpe ratio costs over the sample, using Equation (5.34).

Note, in each sample draw we use the same observation dates across both methodologies in order to fairly compare the realized costs over the resulting sample. We report the dominance ratio (DR), which we define as the proportion of samples for which the realized cost of the IPO model is less than that of the OLS model.

Our experiments should be interpreted as a proof-of-concept, rather than a fully comprehensive financial study. That said, we believe that the results presented below provide compelling evidence for using IPO for estimating regression coefficients. In general, the IPO models exhibit lower out-of-sample MVO costs and improved economic outcomes in comparison to the traditional OLS-based 'predict, then optimize' approach.

5.4.1 Experiment 1: $\mathbb{S}$ unconstrained, $f(\mathbf{x}, \boldsymbol{\theta})$ univariate

Economic performance metrics and average out-of-sample MVO costs are provided in Table 5.5 for the time period of 2000-01-01 to 2020-12-31 for the unconstrained MVO portfolios with univariate prediction models. Equity growth charts for the same time period are provided in Figure 5.11. We first observe that the IPO model provides higher absolute and risk-adjusted performance, as measured by the MVO cost and Sharpe ratio. Indeed the IPO model produces an out-of-sample MVO cost that is approximately 50% lower and a Sharpe ratio that is approximately 100% larger than that of the OLS model. Furthermore, the IPO models provide more conservative risk metrics, as measured by portfolio volatility, value-at-risk (VaR), and average drawdown (Avg DD). These results are highly encouraging for the IPO model.

In Figure 5.12 we compare the realized MVO and Sharpe ratio costs across 1000 out-of-sample realizations. In general we observe that the IPO model exhibits consistently lower MVO costs and generally higher Sharpe ratios than the OLS model. In Figure 5.12(a) we report a dominance ratio of 97% meaning that the IPO model realizes a lower MVO cost in 97% of samples in comparison to the OLS model. Figure 5.12(b) reports a dominance ratio of 68%.

In Figure 5.13 we report the estimated univariate regression coefficients and ± 1 standard error bar for the last out-of-sample data fold. As stated earlier, it is clear that the majority of the IPO and OLS regression coefficients are not statistically significant at an individual market basis. Note that for some markets, the IPO model provides very different regression coefficients, in both magnitude and sign, compared to the OLS coefficients. In particular we observe that, with the exception of Cocoa (CC), all IPO regression coefficients are positive. In contrast, 33% of OLS coefficients are negative.

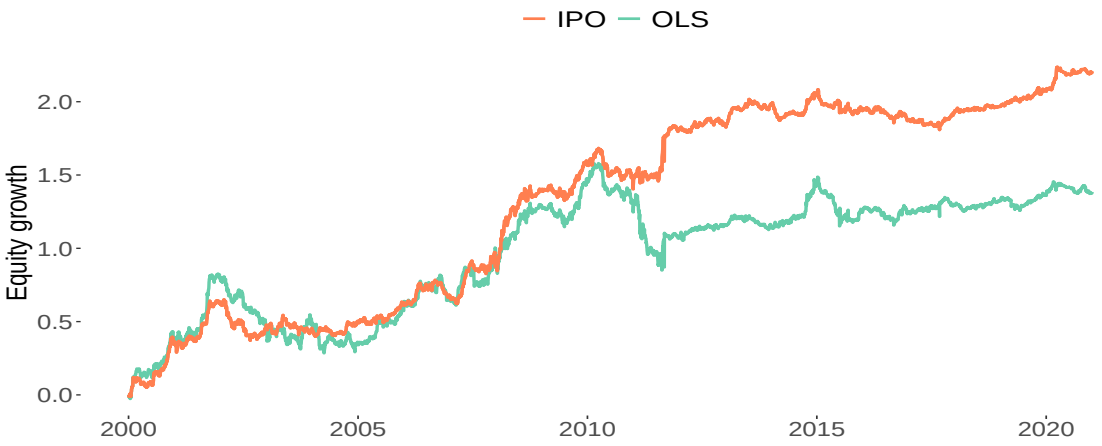
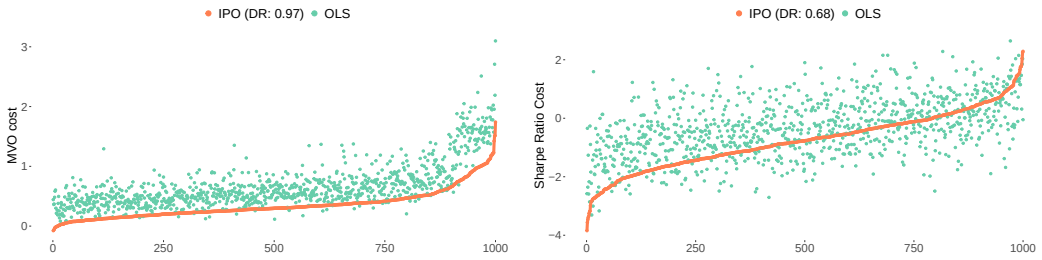


Figure 5.11: Out-of-sample log-equity growth for the unconstrained mean-variance program and univariate IPO and OLS prediction model.

	Annual Return	Sharpe Ratio	Volatility	Avg Drawdown	Value at Risk	MVO Cost
IPO	0.1026	0.7593	0.1352	-0.0275	-0.0107	0.3544
OLS	0.0644	0.3735	0.1725	-0.0426	-0.0142	0.6792

Table 5.5: Out-of-sample MVO costs and economic performance metrics for unconstrained mean-variance portfolios with univariate IPO and OLS prediction models.



(a) Out-of-sample MVO cost.

(b) Out-of-sample Sharpe ratio cost.

Figure 5.12: Realized out-of-sample MVO and Sharpe ratio costs for the unconstrained mean-variance program and univariate IPO and OLS prediction models.

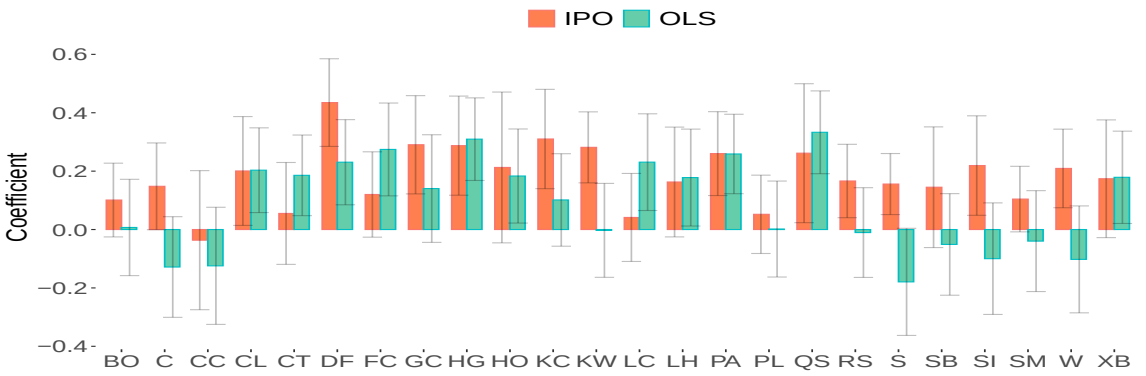


Figure 5.13: Optimal IPO and OLS regression coefficients for the unconstrained mean-variance program and univariate prediction model.

5.4.2 Experiment 2: $\mathbb{S}$ unconstrained, $f(\mathbf{x}, \boldsymbol{\theta})$ multivariate

Economic performance metrics and average out-of-sample MVO costs are provided in Table 5.6 for the time period of 2000-01-01 to 2020-12-31 for the unconstrained MVO portfolios with multivariate prediction models. Equity growth charts for the same time period are provided in Figure 5.14. Again we observe that the IPO model provides higher absolute and risk-adjusted performance and in general more conservative risk metrics. The IPO model produces an out-of-sample MVO cost that is approximately 50% lower and a Sharpe ratio that is approximately 100% larger than that of the OLS model. In Figure 5.15 we compare the realized MVO and Sharpe ratio costs across 1000 out-of-sample realizations. Again we observe that the IPO model exhibits consistently lower MVO costs with a dominance ratio of 99% and generally lower Sharpe ratio costs with a dominance ratio of 65%.

In Figure 5.16 we report the estimated regression coefficients and ± 1 standard error bar for the last out-of-sample data fold. As before, the majority of the IPO and OLS regression coefficients are not statistically significant at an individual market basis. Figures 5.16 (a) and 5.16 (b) report the estimated regression coefficients for the Carry and Trend feature features, respectively. Again we observe that the IPO model provides very different regression coefficients, in both magnitude and sign, compared to the OLS coefficients. Observe that in the multivariate regression model, 50% of the OLS Trend coefficients are negative. In contrast, the IPO model has only 3 (12.5%) negative coefficients: Cocoa (CC), Live Cattle (LC) and Platinum (PL). Furthermore, in many cases such as Feeder Cattle (FC) and Soymeal (SM), the OLS coefficients are relatively large (> 0.30) whereas the corresponding IPO coefficients are effectively zero. Lastly note that the magnitude of the coefficients is approximately 10x larger than the corresponding trend coefficients and is a result of the carry

feature feature values being approximately an order of magnitude smaller.

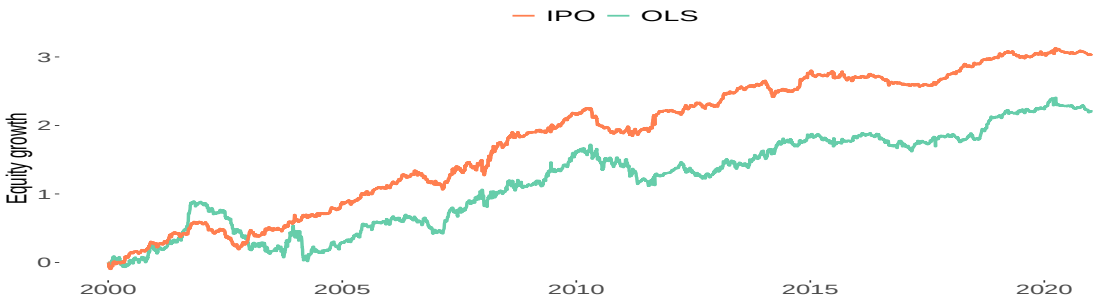
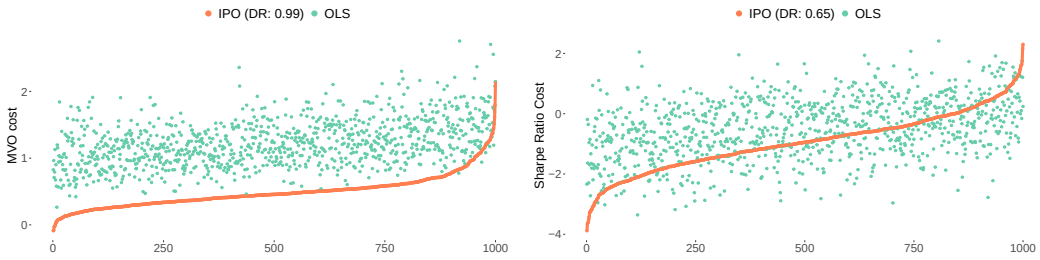


Figure 5.14: Out-of-sample log-equity growth for the unconstrained mean-variance program and multivariate IPO and OLS prediction model.

	Annual Return	Sharpe Ratio	Volatility	Avg Drawdown	Value at Risk	MVO Cost
IPO	0.1416	0.8835	0.1603	-0.0294	-0.0138	0.5004
OLS	0.1034	0.4477	0.2310	-0.0438	-0.0208	1.2308

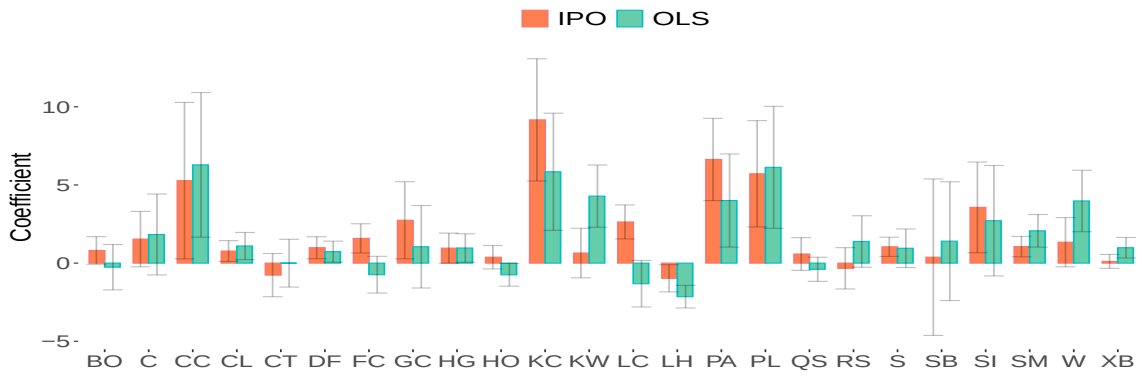
Table 5.6: Out-of-sample MVO costs and economic performance metrics for unconstrained mean-variance portfolios with multivariate IPO and OLS prediction models.



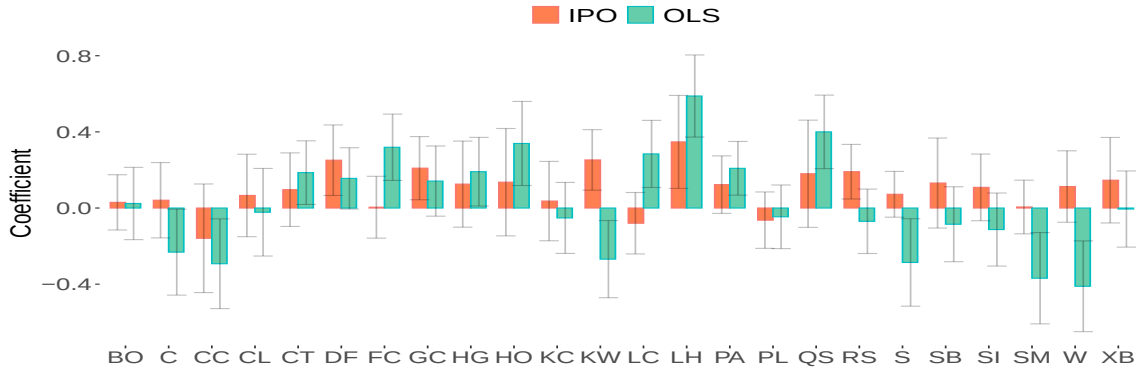
(a) Out-of-sample MVO cost.

(b) Out-of-sample Sharpe ratio cost.

Figure 5.15: Realized out-of-sample MVO and Sharpe ratio costs for the unconstrained mean-variance program and multivariate IPO and OLS prediction models.



(a) Feature: Carry.



(b) Feature: Trend.

Figure 5.16: Optimal IPO and OLS regression coefficients for the unconstrained mean-variance program and multivariate prediction model.

5.4.3 Experiment 3: $\mathbb{S} = \{\mathbf{z} \mid \mathbf{A}\,\mathbf{z} = \mathbf{b}\}$, $f(\mathbf{x}, \boldsymbol{\theta})$ univariate

Economic performance metrics and average out-of-sample MVO costs are provided in Tables 5.7 and 5.8 for the equality constrained MVO portfolios with univariate and multivariate prediction models, respectively. Equity growth charts for the time period of 2000-01-01 to 2020-12-31 are provided in Figures 5.17 and 5.19. As in the unconstrained case, we observe that the IPO model provides higher absolute and risk-adjusted performance, and in general produces more conservative risk metrics. Figures 5.18 and 5.20 demonstrate that the IPO model produces consistently lower out-of-sample MVO costs, with dominance ratios of 93% and 99%, respectively, and generally lower Sharpe ratio costs with dominance ratios of 67% and 66%, respectively.

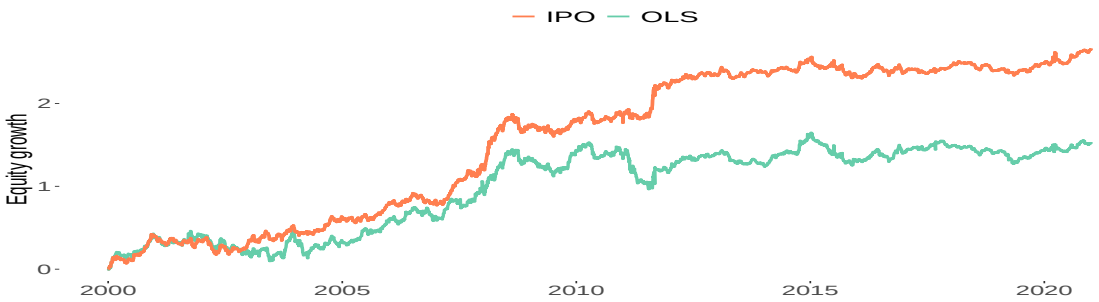
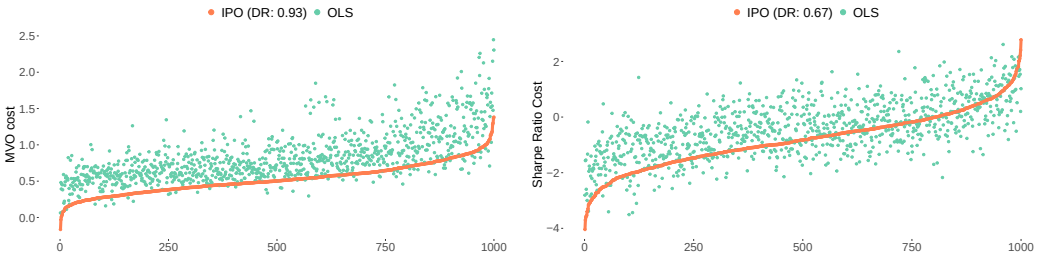


Figure 5.17: Out-of-sample log-equity growth for the equality constrained mean-variance program and univariate IPO and OLS prediction model.

	Annual Return	Sharpe Ratio	Volatility	Avg Drawdown	Value at Risk	MVO Cost
IPO	0.1238	0.7665	0.1616	-0.0290	-0.0142	0.5288
OLS	0.0713	0.3803	0.1876	-0.0471	-0.0170	0.8082

Table 5.7: Out-of-sample MVO costs and economic performance metrics for equality constrained mean-variance portfolios with univariate IPO and OLS prediction models.



(a) Out-of-sample MVO cost.

(b) Out-of-sample Sharpe ratio cost.

Figure 5.18: Realized out-of-sample MVO and Sharpe ratio costs for the equality constrained mean-variance program and univariate IPO and OLS prediction models.

5.4.4 Experiment 4: $\mathbb{S} = \{\mathbf{z} \mid \mathbf{A}\,\mathbf{z} = \mathbf{b}\}$, $f(\mathbf{x}, \boldsymbol{\theta})$ multivariate

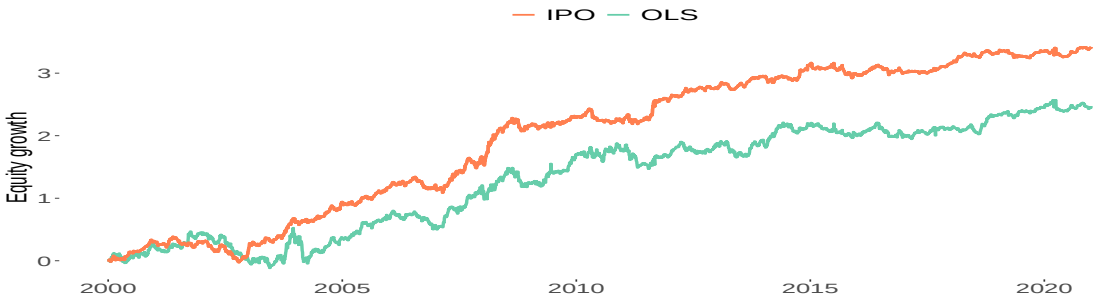
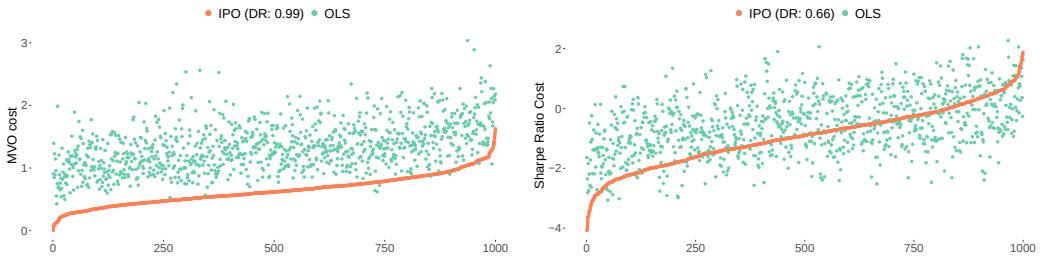


Figure 5.19: Out-of-sample log-equity growth for the equality constrained mean-variance program and multivariate IPO and OLS prediction model.

	Annual Return	Sharpe Ratio	Volatility	Avg Drawdown	Value at Risk	MVO Cost
IPO	0.1590	0.8851	0.1797	-0.0339	-0.0163	0.6482
OLS	0.1151	0.4784	0.2406	-0.0497	-0.0215	1.3315

Table 5.8: Out-of-sample MVO costs and economic performance metrics for equality constrained mean-variance portfolios with multivariate IPO and OLS prediction models.



(a) Out-of-sample MVO cost.

(b) Out-of-sample Sharpe ratio cost.

Figure 5.20: Realized out-of-sample MVO and Sharpe ratio costs for the equality constrained mean-variance program and multivariate IPO and OLS prediction models.

5.4.5 Experiment 5: $\mathbb{S} = \{\mathbf{z} \mid \mathbf{A}\,\mathbf{z} = \mathbf{b}, \mathbf{G}\,\mathbf{z} \leq \mathbf{h}\}$, $f(\mathbf{x}, \boldsymbol{\theta})$ univariate

Economic performance metrics and average out-of-sample MVO costs are provided in Table 5.9 for the time period of 2000-01-01 to 2020-12-31 for the constrained MVO portfolios with univariate prediction models. Equity growth charts for the same time period are provided in Figure 5.21. First, observe that the annual returns, risk and MVO costs are substantially smaller in the presence of portfolio constraints. Indeed this is consistent with the fact that box constraints are themselves a form of portfolio model regularization [75]. Nonetheless, we observe that the IPO model produces an out-of-sample MVO cost that is approximately 50% lower and a Sharpe ratio that is approximately 85% larger than that of the OLS model. In Figure 5.25 we compare the realized MVO and Sharpe ratio costs across 1000 bootstrapped sample realizations. Again we observe that the IPO model produces lower MVO and Sharpe ratio costs on average. Observe, however, that the dominance ratios are more modest, with values in the 60%-70% range. This result is intuitive and we would expect the out-of-sample performance of the two models to converge as the portfolio constraints become more strict. Indeed the IPO and OLS model would yield identical results in the limit where the portfolio constraints define a single weight, irrespective of the mean and covariance estimation. Lastly, in Figure 5.23 we report the estimated univariate regression coefficients and ± 1 standard error bar for the last out-of-sample data fold. Recall that the IPO coefficients are obtained by first dropping the inequality constraints and then solving analytically for $\boldsymbol{\theta}^*$ by Equation (5.29).

The observations and differences between the optimal IPO and OLS coefficients are similar to those discussed in Section 5.4.1.

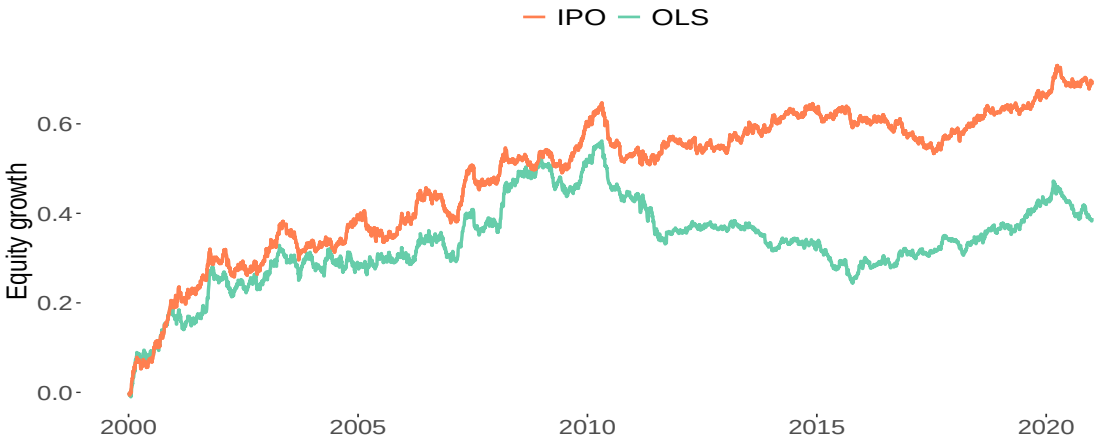
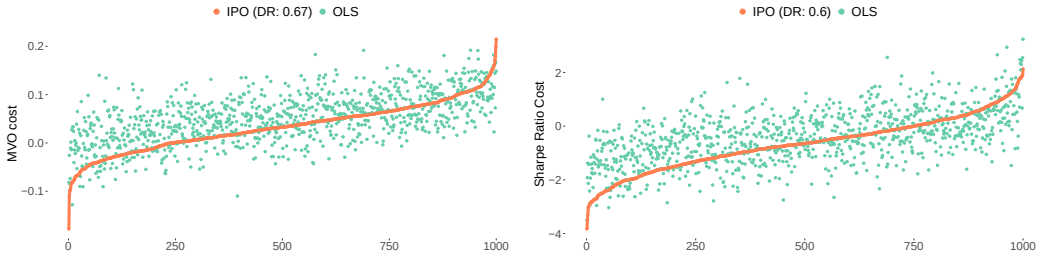


Figure 5.21: Out-of-sample log-equity growth for the inequality constrained mean-variance program and multivariate IPO and OLS prediction model.

	Annual Return	Sharpe Ratio	Volatility	Avg Drawdown	Value at Risk	MVO Cost
IPO	0.0324	0.6310	0.0513	-0.0116	-0.0052	0.0335
OLS	0.0181	0.3421	0.0529	-0.0174	-0.0053	0.0520

Table 5.9: Out-of-sample MVO costs and economic performance metrics for inequality constrained mean-variance portfolios with univariate IPO and OLS prediction models.



(a) Out-of-sample MVO cost.

(b) Out-of-sample Sharpe ratio cost.

Figure 5.22: Realized out-of-sample MVO and Sharpe ratio costs for the inequality constrained mean-variance program and univariate IPO and OLS prediction models.

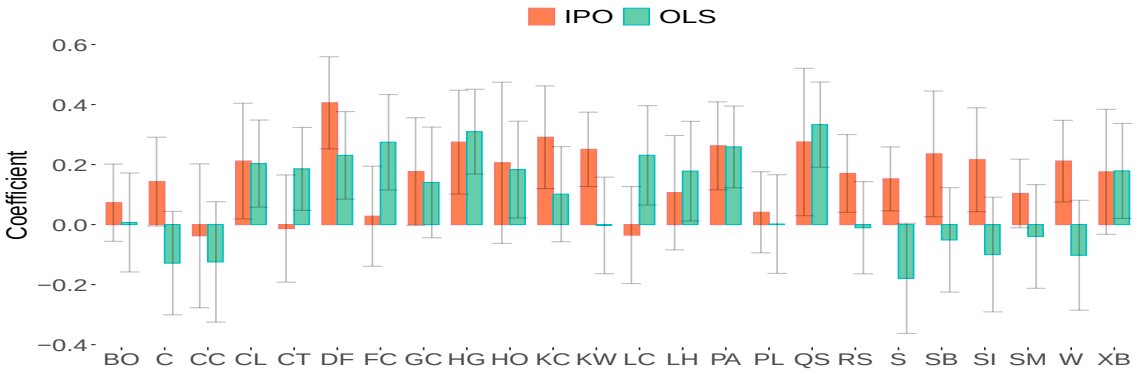


Figure 5.23: Optimal IPO and OLS regression coefficients for the equality constrained mean-variance program and univariate prediction model.

5.4.6 Experiment 6: $\mathbb{S} = \{\mathbf{z} \mid \mathbf{A}\,\mathbf{z} = \mathbf{b}, \mathbf{G}\,\mathbf{z} \leq \mathbf{h}\}$, $f(\mathbf{x}, \boldsymbol{\theta})$ multivariate

Economic performance metrics and average out-of-sample MVO costs are provided in Table 5.10 for the time period of 2000-01-01 to 2020-12-31 for the inequality constrained MVO portfolios with multivariate prediction models. Equity growth charts for the same time period are provided in Figure 5.24. Once again we observe that the IPO model provides modestly higher absolute and risk-adjusted performance and in general more conservative risk metrics. The IPO model produces an out-of-sample MVO cost that is approximately 60% lower and a Sharpe ratio that is approximately 25% larger than that of the OLS model. In Figure 5.25 we compare the realized MVO and Sharpe ratio costs across 1000 out-of-sample realizations. Again we observe more modest dominance ratios with values in the 55%-65% range. We observe that the IPO model provides a modest improvement to performance in comparison to the OLS model; a likely result of lower prediction model misspecification and improved portfolio regularization by virtue of the box constraints. The estimated regression coefficients are provided in Figure 5.26 and the findings are similar to those described in Section 5.4.2.

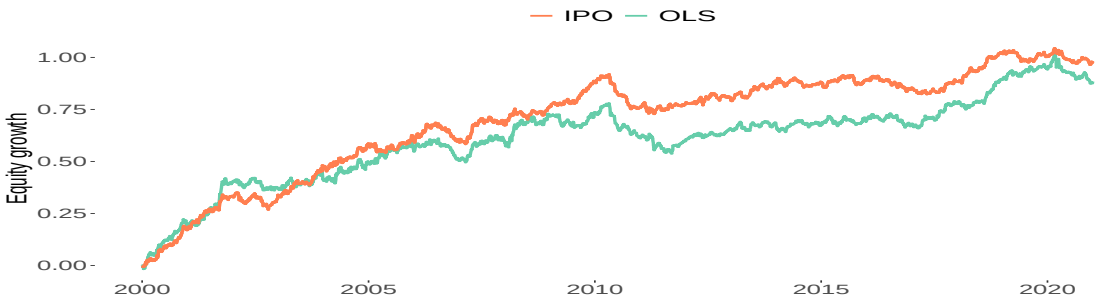
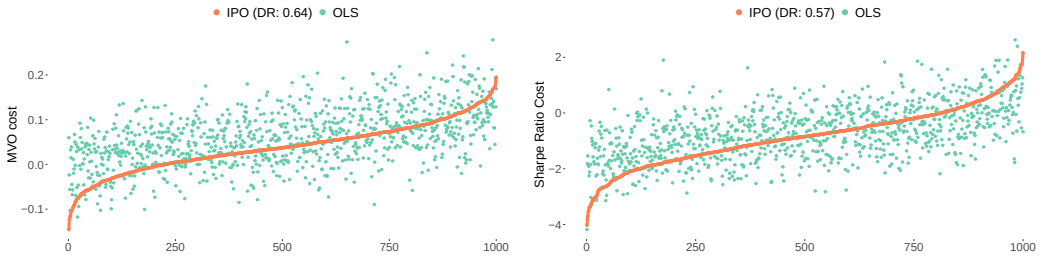


Figure 5.24: Out-of-sample log-equity growth for the inequality constrained mean-variance program and multivariate IPO and OLS prediction model.

	Annual Return	Sharpe Ratio	Volatility	Avg Drawdown	Value at Risk	MVO Cost
IPO	0.0456	0.7937	0.0574	-0.0119	-0.0057	0.0369
OLS	0.0411	0.6488	0.0634	-0.0145	-0.0063	0.0593

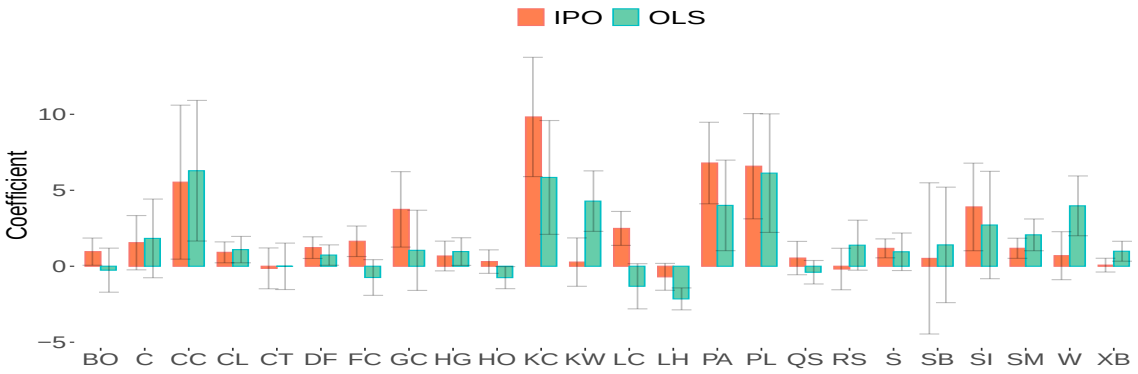
Table 5.10: Out-of-sample MVO costs and economic performance metrics for inequality constrained mean-variance portfolios with multivariate IPO and OLS prediction models.



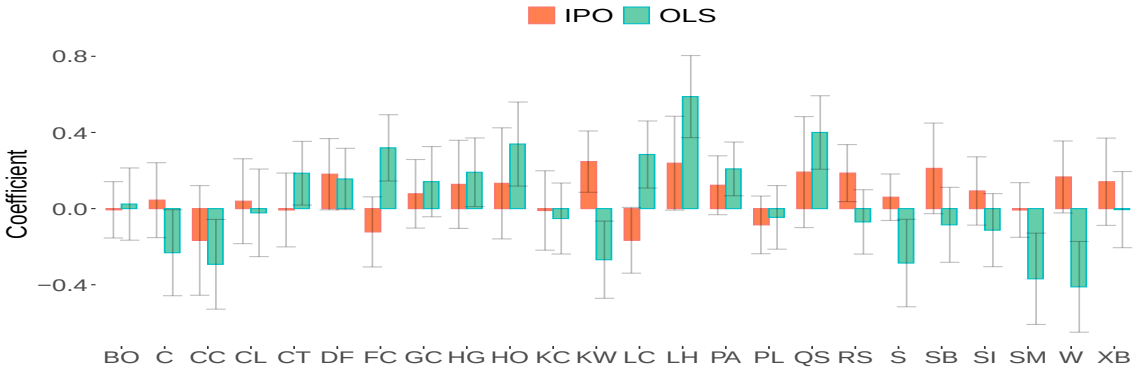
(a) Out-of-sample MVO cost.

(b) Out-of-sample Sharpe ratio cost.

Figure 5.25: Realized out-of-sample MVO and Sharpe ratio costs for the inequality constrained mean-variance program and multivariate IPO and OLS prediction models.



(a) Feature: Carry.



(b) Feature: Trend.

Figure 5.26: Optimal IPO and OLS regression coefficients for the equality constrained mean-variance program and multivariate prediction model.

5.5 Conclusion and future work

In this chapter we proposed an integrated prediction and optimization (IPO) framework for optimizing regression coefficients in the context of a mean-variance portfolio optimization. We investigated the IPO framework under both univariate and multivariate regression settings with a variety of portfolio constraints. In a general setting, we presented the current state-of-the-art approach (IPO-GRAD) and restructured the IPO problem as a neural network with a differentiable quadratic programming layer. Where possible, we provided closed-form analytical solutions for the optimal IPO regression coefficients, $\boldsymbol{\theta}^*$, and the sufficient conditions for uniqueness. We described the sampling distribution properties of $\boldsymbol{\theta}^*$ and provided the conditions for which $\boldsymbol{\theta}^*$ is an unbiased estimator of $\boldsymbol{\theta}$ and provided the expression for the variance.

Extensive numerical simulations demonstrated the computational and performance advantage of the analytical IPO methodology. We demonstrated that, over a wide range of realistic signal-to-noise ratios, the IPO model outperforms the OLS model in terms of minimizing out-of-sample MVO costs. This is true even when the underlying 'ground-truth' return generating process is linear in the feature variables. We demonstrated, for a wide range of portfolio sizes, the computational advantage of computing the IPO coefficients analytically, which is on average 10x-1000x faster than the IPO-GRAD methodology. We briefly discussed the computational complexity of the IPO-GRAD methodology and proposed a heuristic which drops the inequality constraints during parameter estimation and invokes the analytical IPO solution. We hypothesize that in many instances the IPO-GRAD model overfits the training data, whereas the analytical IPO model produces solutions with lower out-of-sample variance, and in many cases, improved out-of-sample MVO costs. We concluded with several experiments using global futures data, under various forms of constraints and prediction model specifications. Out-of-sample results demonstrate that the IPO model provided lower realized MVO costs and superior economic performance in comparison to the traditional OLS 'predict then optimize' approach.

In the presence of general inequality constraints we determined that the current state-of-the-art IPO model is computationally burdensome and has a tendency to overfit the training data. We believe that methods for regularizing both the prediction and the portfolio optimization program, as well as methods for choosing the 'best' feature subsets are an interesting area of future research and may lead to approaches that improve upon our heuristic IPO solution. Future work also includes incorporating other forms of prediction models into the IPO framework as well as exploring methods for performing the more difficult joint prediction of asset returns and covariances.

Chapter 6

Integrated covariance estimation for risk-based portfolio optimization

6.1 Introduction

The aim of this chapter is to present the IPO framework for covariance estimation and to evaluate the effectiveness of a covariance estimation process that is optimized for downstream portfolio objectives. Indeed, there exists well established literature on estimating asset return covariances for use in a downstream portfolio optimization (see for example [17, 40, 85, 86, 88]). As discussed in Chapter 5, the vast majority of the existing literature estimates assets covariances independently from their use in the downstream portfolio optimization. In contrast, in this chapter we present an integrated framework for covariance estimation under a variety of portfolio objectives. Recall that under the integrated setting, the prediction model parameters are estimated in order to yield the 'best' decisions, not to provide the 'best' predictions. In order to isolate the impact of the integrated covariance estimation, we consider purely risk-based portfolio optimizations, namely: minimum-variance [96], maximum-diversification [34] and equal-risk-contribution [93], described in detail in Chapter 2.

The remainder of this chapter is outlined as follows. We begin with a brief literature review in the field of covariance estimation and summarize our primary contributions. In Section 6.2 we describe

the covariance model and review the current approach to parameter estimation. We then state the IPO program and provide the integrated formulations for the risk-based optimizations. We conclude with a simulation study using U.S. industry sector data and U.S. stock data to demonstrate that the integrated framework can provide lower realized costs which, in some cases, translates to improved out-of-sample economic performance in comparison to the 'predict, then optimize' alternative.

6.1.1 Literature review

The majority of risk-based portfolio optimizations require as input a covariance matrix of asset returns. Indeed, volatility and covariance estimation has a long history in the field of quantitative finance and econometrics. The seminal work of Engle [51] and Bollerslev [16] culminated in the now widely accepted univariate generalized autoregressive conditional heteroskedasticity (GARCH) model, which estimates conditional variances as a linear combination of past realizations and variance estimates. With a relatively flexible lag structure, GARCH models have proven capable of capturing long-memory volatility effects and other 'stylized-facts' common to many financial time-series [17, 19].

The first direct extension to multivariate GARCH modelling, VEC-GARCH, was introduced by Bollerslev et al. [18], but was found to be highly impractical for problems with more than three assets. Since then, several alternative multivariate GARCH models have been proposed, and we refer the reader to Bauwens et al. [11] for a comprehensive overview. Of the more practical multivariate GARCH models, the Constant Conditional Correlation (CCC-GARCH) model, presented by Bollerslev [17], overcomes the dimensionality issues of VEC-GARCH by assuming a constant correlation matrix. Fitting CCC-GARCH models is straightforward; first individual univariate GARCH parameters are estimated via maximum-likelihood and then the constant conditional correlation matrix is estimated from the resulting GARCH residuals. Empirical studies demonstrate that the constant correlation assumption is valid for many asset classes [120] and the effectiveness of the estimator is typically evaluated by its ability to minimize realized portfolio variance (see for example [88, 123]).

Under certain circumstances, the assumption of constant conditional correlation may be unrealistic. The Dynamic Conditional Correlation (DCC-GARCH) model, introduced by Engle [49], allows for a time-varying conditional correlation matrix and can handle problems of moderate to large size. As described in Section 6.2, the DCC-GARCH model estimates a correlation matrix proxy process as a linear combination of past standardized realization cross-products and its lagged correlation proxy estimates. Again, parameter fitting is typically performed by maximum-likelihood estimation

using a multi-step approach:

1. For each asset fit a univariate GARCH model by maximum-likelihood.
2. Estimate the unconditional correlation matrix from GARCH residuals.
3. Estimate the conditional correlation dynamics by maximum-likelihood or composite likelihood [106].

For large scale problems, it is common to model asset returns, and subsequently their covariance, according to a linear factor model (see for example [36, 58, 64]). Factor GARCH models, originally introduced by Engle et al. [52], assume that returns are generated by a set of factors that are themselves conditionally heteroscedastic. More recently, De Nard et al. [40] proposed a time-varying factor covariance model for large scale portfolio problems. Factor regression coefficients are fit by ordinary least-squares (OLS) and the GARCH dynamics are fit according to the multi-step maximum-likelihood optimization, described above. They demonstrate that their factor DCC-GARCH model, when combined with nonlinear shrinkage, can yield more efficient portfolio allocation, as measured by realized out-of-sample portfolio objectives.

6.1.2 Main Contributions

It is clear that the literature is rich with sophisticated and highly effective covariance prediction models. The goal of this chapter is not to provide a new covariance model, but rather to present an alternative framework for parameter estimation of existing models, under the assumption that the covariance estimate will ultimately be used as input to a decision-based optimization problem. To our knowledge this is the first empirical study of integrated covariance estimation in a risk-based portfolio optimization setting.

The IPO framework is similar in spirit to the work of Engle and Colacito [50], who propose evaluating covariance prediction models by a custom loss function that quantifies the out-of-sample variance of optimized portfolios induced by the covariance estimate. The realized portfolio variance loss function was later adopted by Ledoit and Wolf [87] and Engle et al. [53], but was never used for parameter estimation itself. Instead the authors continued performing parameter estimation by maximum-likelihood and/or least-squares and the custom loss function was only ever used as an economic measure of out-of-sample prediction accuracy. Indeed, we observe that in the majority of studies discussed above, the effectiveness of the covariance estimate is measured, at least in part, by

its ability to minimize a realized decision-based portfolio objective, despite the fact that this is not the objective under which the model parameters are optimized.

The IPO framework, described in more detail in Section 6.2.7, provides a practical and approachable solution to this problem by directly integrating the prediction model parameter estimation process in the context of the final decision-based objectives and constraints. In this chapter we consider three risk-based portfolio optimizations: minimum-variance (MV), maximum-diversification (MD), and equal-risk-contribution (ERC), under the standard long-only, fully-invested constraint set. The covariance model, described in Section 6.2, follows closely to the time-varying factor covariance model, recently proposed by De Nard et al. [40], and was chosen for its reported improvement in realized performance and ability to handle problems of large dimension. Furthermore, under traditional parameter estimation, fitting the covariance model requires solving two independent prediction optimization problems: OLS for fitting the factor coefficients and maximum-likelihood for fitting the multivariate GARCH dynamics. The IPO framework, on the other hand, optimizes all model parameters jointly, thus demonstrating the flexibility of the integrated approach and its ability to handle prediction models of arbitrary complexity.

Experiments are presented in Section 6.3. From an experimental standpoint, our goal is to analyze the out-of-sample performance of integrated prediction and optimization in a portfolio optimization setting with real asset price data. Our experiments should be interpreted as a proof-of-concept, rather than a fully comprehensive financial study. In summary, our analytical and experimental contributions are as follows:

1. We present the IPO framework for minimum-variance portfolios, a special case of quadratic programs. Out-of-sample numerical results demonstrate that the integrated approach can provide consistent and economically significant lower realized portfolio variance in comparison to the 'predict, then optimize' alternative. Experimentation on the larger stock universe demonstrates that the difference in realized portfolio variance across the two methods increases as the number of assets in the portfolio increases.

2. We present the IPO framework for maximum-diversification portfolios, which again is a special case of general quadratic programs. Out-of-sample numerical results demonstrate that the IPO framework is successful in providing consistently higher realized portfolio diversification ratios. For our particular data set, however, the difference in realized diversification ratios is not statistically significant and therefore does not result in materially different economic outcomes in comparison to the traditional approach.

3. We present the IPO framework for equal-risk-contribution portfolios. The introduction of a log-barrier term results in a convex objective function that is not a standard quadratic program and the relevant gradient equations for first-order optimization are described in chapter 4. Out-of-sample numerical results demonstrate that, in most cases, the integrated approach can yield a consistently lower dispersion in realized risk-contributions, as measured by the Herfindahl index. For our particular data set we find that the magnitude of the difference in realized risk contributions is small and does not result in meaningfully different economic outcomes.

In summary we observe that when the covariance prediction model is more poorly specified, then the IPO minimum-variance framework will often yield consistent and significantly lower out-of-sample portfolio volatility and result in improved economic outcomes. We find this result encouraging from a proof-of-concept standpoint and it further suggests that the IPO framework may be more resilient to model misspecification in comparison to more traditional parameter estimation approach. For the maximum-diversification and equal-risk-contribution portfolios, the improvement in realized costs is marginal and does not result in improved economic performance. These results are discussed in more detail in Section 6.3. Lastly, we acknowledge that the results will vary considerably depending on the choice of data set and covariance estimation model and therefore further experimentation is required.

6.2 Methodology

We begin by describing the covariance prediction model under the traditional 'predict, then optimize' framework. As before, we consider a universe of d_y assets and denote the matrix of (excess) return observations as $\mathbf{Y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(m)}] \in \mathbb{R}^{m \times d_y}$ with $m > d_y$. For simplicity, we decompose the covariance prediction model into three parts:

1. A linear factor model, estimated via ordinary-least-squares (OLS).

2. A univariate GARCH model for volatility estimation, via maximum-likelihood.

3. A multivariate GARCH model for estimating the conditional correlation, again by maximum-likelihood. We consider both constant and dynamic correlation models, described below.

6.2.1 Linear factor model

Asset returns $\mathbf{y}^{(i)}$ are modelled according to a linear factor model of associated feature variables $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x}$. The standard ordinary least-squares regression model is given as:

$$\hat{\mathbf{y}}^{(i)} = \boldsymbol{\theta}_1^T \mathbf{x}^{(i)} + \boldsymbol{\epsilon}^{(i)}, \tag{6.1}$$

where $\boldsymbol{\theta}_1 \in \mathbb{R}^{d_x \times d_y}$ is the matrix of regression coefficients, and $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{F}) \in \mathbb{R}^{d_y}$ is the vector of residual returns. We let $\mathbf{x}^{(i)} \sim \mathcal{N}(\Phi, \mathbf{W}^{(i)})$, and therefore $\mathbf{W}^{(i)} \in \mathbb{R}^{d_x \times d_x}$ denotes the time-varying covariance matrix. Furthermore, we assume that the residual returns are independent, $\text{cov}(\boldsymbol{\epsilon}^{(i)}, \boldsymbol{\epsilon}^{(j)}) = 0 \ \forall i \neq j$. The static matrix $\mathbf{F} \in \mathbb{R}^{d_y \times d_y}$ therefore denotes the diagonal matrix of residual variance.

Recall that traditionally $\boldsymbol{\theta}_1$ is chosen in order to minimize a least-squares loss function. Specifically, given training data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$ we choose $\hat{\boldsymbol{\theta}}_1$ such that:

$$\hat{\boldsymbol{\theta}}_1 = \underset{\boldsymbol{\theta}_1}{\text{argmin}} \, \mathbb{E}_{\mathcal{D}}[\| \boldsymbol{\theta}_1^T \mathbf{x}^{(i)} - \mathbf{y}^{(i)} \|_2], \tag{6.2}$$

where $\mathbb{E}_{\mathcal{D}}$ denotes the expectation operator with respect to the training distribution. The resulting least-squares estimators are given by:

$$\begin{aligned} \hat{\mathbf{y}}^{(i)} &= \hat{\boldsymbol{\theta}}_1^T \mathbf{x}^{(i)} \\ \hat{\mathbf{V}}^{(i)} &= \hat{\boldsymbol{\theta}}_1^T \hat{\mathbf{W}}^{(i)} \hat{\boldsymbol{\theta}}_1 + \hat{\mathbf{F}}, \end{aligned} \tag{6.3}$$

where $\hat{\mathbf{V}}^{(i)}$ denotes the conditional covariance estimate of asset returns.

6.2.2 Univariate GARCH

To model the time-varying dynamics of $\hat{\mathbf{W}}^{(i)}$, we follow De Nard et al. [40] and assume that the feature variables are themselves conditionally heteroscedastic. We model the feature variables volatility as univariate GARCH$(1,1)$ processes and consider two conditional correlation estimates: the constant conditional correlation (CCC-GARCH) model by Bollerslev [17] and the dynamic conditional correlation (DCC-GARCH) model by Engle [49].

Following Bollerslev [16], the univariate mean equation for feature variable j is given by:

$$\begin{aligned} {\mathbf{x}_{\mathrm{j}}}^{(i)} &= \boldsymbol{\phi}_j + {\boldsymbol{\varepsilon}_{\mathrm{j}}}^{(i)} \\ {\boldsymbol{\varepsilon}_{\mathrm{j}}}^{(i)} &= {\boldsymbol{\sigma}_{\mathrm{j}}}^{(i)} \, \boldsymbol{\xi}^{(i)}, \end{aligned} \tag{6.4}$$

where $\boldsymbol{\xi}^{(i)} \sim \mathcal{N}(0,1)$ i.i.d. The single step forward variance is then given by:

$$\begin{aligned} {\boldsymbol{\sigma}_{\mathrm{j}}}^{2^{(i)}} &= \boldsymbol{\omega}_j + \boldsymbol{\alpha}_j \, {\boldsymbol{\varepsilon}_{\mathrm{j}}}^{2^{(i-1)}} + \boldsymbol{\beta}_j \, {\boldsymbol{\sigma}_{\mathrm{j}}}^{2^{(i-1)}} \\ &= \frac{1}{1-\boldsymbol{\beta}_j} \, \boldsymbol{\omega}_j + \boldsymbol{\alpha}_j \sum_{k=1}^{\infty} \boldsymbol{\beta}_j^{k-1} \, {\boldsymbol{\varepsilon}_{\mathrm{j}}}^{2^{(i-k)}}, \end{aligned} \tag{6.5}$$

where $0 \leq \boldsymbol{\omega}_j \leq 1$, $0 \leq \boldsymbol{\alpha}_j \leq 1$ and $0 \leq \boldsymbol{\beta}_j \leq 1 \; \forall j \in \{1, 2, ..., d_x\}$.

Traditionally, the model parameters $\boldsymbol{\theta}_2 = [\boldsymbol{\omega}, \boldsymbol{\alpha}, \boldsymbol{\beta}]$, are optimized by maximum-likelihood. Under a normal distribution, the model parameters are determined by minimizing the constrained negative log-likelihood, given by Program (6.6):

$$\begin{aligned} \underset{\boldsymbol{\theta}_2}{\text{minimize}} \quad & \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{d_x} \left(\ln({\sigma_{\mathrm{j}}}^{2^{(i)}}) + \frac{{\boldsymbol{\varepsilon}_{\mathrm{j}}}^{2^{(i)}}}{{\sigma_{\mathrm{j}}}^{2^{(i)}}} + \ln(2\pi) \right) \\ \text{subject to} \quad & 0 < \boldsymbol{\theta}_2 < 1. \end{aligned} \tag{6.6}$$

6.2.3 Multivariate CCC-GARCH

We let $\mathbf{s}^{(i)} = {\mathbf{D}^{-1}}^{(i)} \, \boldsymbol{\varepsilon}^{(i)}$ denote the standardized GARCH residuals, where $\mathbf{D}^{(i)} = \mathrm{diag}([{\boldsymbol{\sigma}_1}^{(i)}, {\boldsymbol{\sigma}_2}^{(i)}, ..., {\boldsymbol{\sigma}_{\mathrm{d_x}}}^{(i)}])$, determined above. The constant conditional correlation estimate is static and is given by:

$$\mathbf{R}^{(i)} = \bar{\mathbf{R}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{s}^{(i)} \, \mathbf{s}^{T(i)}, \tag{6.7}$$

and the time-varying conditional covariance estimate:

$$\hat{\mathbf{W}}^{(i)} = \mathbf{D}^{(i)} \, \bar{\mathbf{R}} \, \mathbf{D}^{(i)} \, . \tag{6.8}$$

6.2.4 Multivariate DCC-GARCH

Alternatively, under the dynamic conditional correlation model, we follow Engle [49] and model a proxy correlation process, $\mathbf{Q}^{(i)}$ as:

$$\begin{aligned}\mathbf{Q}^{(i)} &= (1-(a_1+a_2))\bar{\mathbf{R}} + a_1\,\mathbf{s}^{(i-1)}\,\mathbf{s}^{T(i-1)} + a_2\,\mathbf{Q}^{(i-1)} \\ &= \frac{(1-(a_1+a_2))}{1-a_2}\bar{\mathbf{R}} + a_1\sum_{k=1}^{\infty} a_2^{k-1}\,\mathbf{s}^{(i-k)}\,\mathbf{s}^{T(i-k)},\end{aligned} \tag{6.9}$$

where $a_1 > 0$, $a_2 > 0$ and $a_1 + a_2 < 1$ to ensure stationarity and positive definiteness. The dynamic conditional correlation estimate is then given by:

$$\mathbf{R}^{(i)} = \operatorname{diag}(\mathbf{Q}^{(i)})^{-\frac{1}{2}}\,\mathbf{Q}^{(i)}\,\operatorname{diag}(\mathbf{Q}^{(i)})^{-\frac{1}{2}}. \tag{6.10}$$

The negative log-likelihood function for the covariance estimate, $\hat{\mathbf{W}}^{(i)}$, in its entirety, is given by Equation (6.12).

$$\begin{aligned}&\frac{1}{2}\sum_{i=1}^{m}\left(\ln(2\pi) + \ln(|\,\hat{\mathbf{W}}^{(i)}\,|) + \boldsymbol{\varepsilon}^{T(i)}\,\hat{\mathbf{W}}^{-1^{(i)}}\,\boldsymbol{\varepsilon}^{(i)}\right) \\ =&\frac{1}{2}\sum_{i=1}^{m}\left(\ln(2\pi) + 2\ln(|\,\mathbf{D}^{(i)}\,|) + \ln(|\,\mathbf{R}^{(i)}\,|) + \mathbf{s}^{T(i)}\,\mathbf{R}^{-1^{(i)}}\,\mathbf{s}^{(i)}\right) \\ =&\frac{1}{2}\sum_{i=1}^{m}\left(\ln(2\pi) + \underbrace{2\ln(|\,\mathbf{D}^{(i)}\,|) + \mathbf{s}^{T(i)}\,\mathbf{s}^{(i)}}_{\text{univariate component}} + \underbrace{\ln(|\,\mathbf{R}^{(i)}\,|) + \mathbf{s}^{T(i)}\,\mathbf{R}^{-1^{(i)}}\,\mathbf{s}^{(i)} - \mathbf{s}^{T(i)}\,\mathbf{s}^{(i)}}_{\text{multivariate component}}\right)\end{aligned} \tag{6.11}$$

Under a Normal distribution assumptions, a multi-step approach is justified. Specifically, we first fit the individual univariate GARCH models by maximum-likelihood estimation. The univariate GARCH residuals are then used to optimize the multivariate parameters, $\boldsymbol{\theta}_3 = (a_1, a_2)$, provided by Program (6.12).

$$\begin{aligned}\underset{\boldsymbol{\theta}_3}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{m}\left(\ln(|\,\mathbf{R}^{(i)}\,|) + \mathbf{s}^{T(i)}\,\mathbf{R}^{-1^{(i)}}\,\mathbf{s}^{(i)} - \mathbf{s}^{T(i)}\,\mathbf{s}^{(i)} + \ln(2\pi)\right) \\ \text{subject to} \quad & 0 < \boldsymbol{\theta}_3 < 1.\end{aligned} \tag{6.12}$$

Lastly, the time-varying covariance estimate is given by:

$$\hat{\mathbf{W}}^{(i)} = \mathbf{D}^{(i)}\,\mathbf{R}^{(i)}\,\mathbf{D}^{(i)}\,. \tag{6.13}$$

6.2.5 Multi-step forward predictions

In practice, it is often the case that the holding period for a portfolio is greater than 1-bar. In all experiments, we rebalance $1/4^{\text{th}}$ of the portfolio on a weekly basis. Therefore, the holding period for any particular portfolio decision is approximately 1 business month. From a common sense standpoint, it is desirable to generate covariance forecasts that are consistent with the holding period.

We follow De Nard et al. [40] and use weekly data for parameter estimation and then project the covariance estimate forward over every forecast horizon from 1 to 4 weeks. Our final covariance estimate is then then average of all 4 covariance estimates. Specifically, we let $\hat{\mathbf{W}}^{(i)}(n)$ denote the n-day forward covariance estimate. Our final feature variable covariance estimate is then given as:

$$\bar{\mathbf{W}}^{(i)} = \frac{1}{4}\sum_{n=1}^{4}\hat{\mathbf{W}}^{(i)}(n). \tag{6.14}$$

The equations for the n-step forward variance and proxy correlation are provided below. From the univariate GARCH model, under the stationarity condition, $\boldsymbol{\alpha}_j + \boldsymbol{\beta}_j < 1$, the long-run variance is given by:

$$\mathbb{E}[{\sigma_{\text{j}}}^{2^{(i)}}] = \mathbb{E}[\boldsymbol{\omega}_j + \boldsymbol{\alpha}_j\,{\boldsymbol{\varepsilon}_{\text{j}}}^{2^{(i-1)}} + \boldsymbol{\beta}_j\,{\boldsymbol{\sigma}_{\text{j}}}^{2^{(i-1)}}] = \frac{\boldsymbol{\omega}_j}{1-\boldsymbol{\alpha}_j-\boldsymbol{\beta}_j}. \tag{6.15}$$

The n-step forward variance is then given by the recursive equation:

$${\sigma_{\text{j}}}^{2^{(i)}}(n) = \boldsymbol{\omega}_j + (\boldsymbol{\alpha}_j+\boldsymbol{\beta}_j)\,{\sigma_{\text{j}}}^{2^{(i)}}(n-1), \tag{6.16}$$

which under stationarity assumptions simplifies to a weighted average of the single-step forward estimate and the long-run average variance:

$${\sigma_{\text{j}}}^{2^{(i)}}(n) = (1-(\boldsymbol{\alpha}_j+\boldsymbol{\beta}_j)^{n-1})\frac{\boldsymbol{\omega}_j}{1-(\boldsymbol{\alpha}_j+\boldsymbol{\beta}_j)} + (\boldsymbol{\alpha}_j+\boldsymbol{\beta}_j)^{n-1}\,{\sigma_{\text{j}}}^{2^{(i)}}(1). \tag{6.17}$$

As in the univariate case, under the stationarity condition, $a_1 + b_1 < 1$, the long-run proxy

correlation is given by:

$$\mathbb{E}[\mathbf{Q}^{(i)}] = \mathbb{E}[(1-(a_1+a_2))\bar{\mathbf{R}} + a_1\,\mathbf{s}^{(i-1)}\,\mathbf{s}^{T(i-1)} + a_2\,\mathbf{Q}^{(i-1)}] = \bar{\mathbf{R}}, \tag{6.18}$$

demonstrating that the long-run expectation of the proxy correlation process is the constant conditional correlation estimate. Similarly, the n-step forward correlation can be expressed as a weighted average of the single-step forward estimate and the long-run average correlation:

$$\mathbf{Q}^{(i)}(n) = (1-(a_1+b_1)^{n-1})\bar{\mathbf{R}} + (a_1+b_1)^{n-1}\,\mathbf{Q}^{(i)}(1). \tag{6.19}$$

6.2.6 Covariance model summary

Note that our particular covariance model exhibits a complicated dependency between the various model parameters. As outlined above, a traditional parameter estimation approach would treat each parameter set independently and can be summarized by the following steps:

1. Fit the regression coefficients, $\boldsymbol{\theta}_1$, by optimizing the OLS Program (6.2).
2. Fit the univariate GARCH parameters, $\boldsymbol{\theta}_2$, by optimizing the maximum-likelihood Program (6.6).
3. Use the GARCH residuals to estimate the constant conditional correlation, provided by Equation (6.7).
4. In the case of DCC-GARCH, fit the multivariate GARCH parameters, $\boldsymbol{\theta}_3$, by optimizing the maximum-likelihood Program (6.12).
5. Combine volatility and correlation estimates to form the time-varying covariance estimate, provided by Equation (6.14).
6. Combine the covariance estimate with the OLS coefficients to form the time-varying asset covariance estimate, provided by Equation (6.3).

Going forward, for compactness, and without loss of generality, we denote the covariance prediction model parameters by $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3]$ and let $f\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_y \times d_y}$ denote the $\boldsymbol{\theta}$-parameterized prediction model:

$$\hat{\mathbf{V}}^{(i)} = f(\mathbf{x}^{(i)}, \boldsymbol{\theta}). \tag{6.20}$$

6.2.7 Integrated prediction and optimization

Incorporating the covariance prediction into a decision-based optimization according to the traditional 'predict, then optimize' framework is straightforward. We assume that we have a risk-based cost function $c\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_z \times d_z} \to \mathbb{R}$, which takes as input the portfolio decision vector, $\mathbf{z}$, constrained to a feasible region $\mathbb{S} \subset \mathbb{R}^{d_z}$, and the estimated covariance matrix $\hat{\mathbf{V}}^{(i)}$. Generating 'optimal decisions' therefore amounts to plugging in the covariance estimate, $\hat{\mathbf{V}}^{(i)}$, and solving the following deterministic optimization problem:

$$\mathbf{z}^*(\hat{\mathbf{V}}^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}}{\operatorname{argmin}}\, c(\mathbf{z}, \hat{\mathbf{V}}^{(i)}). \tag{6.21}$$

Recall that in the integrated framework, we optimize prediction model parameters, $\boldsymbol{\theta}$, to minimize the average realized decision cost of the decisions: $\{\mathbf{z}^*(\hat{\mathbf{V}}^{(i)})\}_{i=1}^m$ induced by this parameterization. The resulting integrated prediction-optimization (IPO) problem is a stochastic optimization problem, presented in discrete from in Program (6.22).

$$\begin{aligned} \underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \quad & \frac{1}{m}\sum_{i=1}^{m} c(\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}), \mathbf{V}^{(i)}) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{V}}^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}}{\operatorname{argmin}}\, c(\mathbf{z}, \hat{\mathbf{V}}^{(i)}) \quad \forall i \in 1, ..., m. \end{aligned} \tag{6.22}$$

We apply the first-order methodology described in chapter 4 and reformulate Program (6.22) as a end-to-end trainable neural network. The IPO equivalent neural network structure is depicted in Figure 6.1. In the forward pass, the input layer takes the feature variables $\mathbf{x}^{(i)}$ and passes them to our prediction model, $f(\mathbf{x}^{(i)}, \boldsymbol{\theta})$, to produce the estimates, $\hat{\mathbf{V}}^{(i)}$. Covariance predictions are then passed to a differentiable optimization layer which, for a given input $\hat{\mathbf{V}}^{(i)}$, solves the portfolio optimization program and returns the optimal portfolio weights $\mathbf{z}^*(\hat{\mathbf{V}}^{(i)})$. Finally, the quality of the portfolio weights are evaluated by the cost function, $c(\mathbf{z}^*(\hat{\mathbf{V}}^{(i)}), \mathbf{V}^{(i)})$ in the context of the realized (true) covariance matrix, $\mathbf{V}^{(i)}$.

The partial derivative of the cost with respect to the covariance estimate, $\hat{\mathbf{V}}^{(i)}$, is computed by implicitly differentiating the solution mapping provided by the KKT conditions at optimality, $\mathbf{z}^*(\hat{\mathbf{V}}^{(i)})$, as described in chapter 4. The backpropagation algorithm then computes the remaining partial derivatives in the regular fashion in order to generate the gradient of the realized cost with respect to the prediction model parameters.

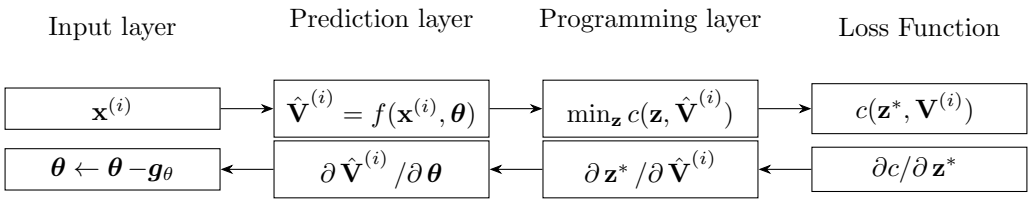


Figure 6.1: IPO program represented as an end-to-end neural network with predictive covariance estimation layer, differentiable convex programming layer and realized portfolio cost loss function.

As before, we search for a locally optimal solution using stochastic gradient descent (SGD), with descent direction, $\boldsymbol{g}_\theta$, given by :

$$\boldsymbol{g}_\theta = \frac{1}{|B|} \sum_{i \in B} \Big(\frac{\partial c}{\partial \boldsymbol{\theta}} \Big)_{|(\mathbf{z}^*(\hat{\mathbf{V}}^{(i)}), \mathbf{V}^{(i)})} \approx \nabla_\theta \bar{c}.$$

6.2.8 IPO: minimum-variance portfolio

In all experiments, we define the constraint set $\mathbb{S} = \{\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}} \mid \mathbf{z}^T \mathbf{1} = 1, \mathbf{z} \geq 0\}$, and thus $\mathbf{z}$ is a long-only, fully-invested portfolio. The minimum-variance portfolio is a special case of the mean-variance program described in detail Section 2.2 with cost function:

$$c_{MV}(\mathbf{z}, \mathbf{V}) = \frac{1}{2} \mathbf{z}^T \mathbf{V} \mathbf{z} \,. \tag{6.23}$$

The full IPO formulation for the minimum-variance portfolio is straightforward and is presented in Program (6.24):

$$\begin{aligned} \underset{\boldsymbol{\theta} \in \Theta}{\text{minimize}} \quad & \frac{1}{m} \sum_{i=1}^{m} c_{MV}(\mathbf{z}^*(\hat{\mathbf{V}}^{(i)}), \mathbf{V}^{(i)}) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{V}}^{(i)}) = \underset{\mathbf{z} \in \mathbb{Z}}{\operatorname{argmin}} \, c_{MV}(\mathbf{z}, \hat{\mathbf{V}}^{(i)}) \quad \forall i \in 1, ..., m. \end{aligned} \tag{6.24}$$

6.2.9 IPO: maximum-diversification portfolio

The maximum-diversification portfolio is described in detail in Section 2.3. Recall, Choueifaty and Coignard [34] define the 'diversification ratio' as the ratio of weighted average asset volatility to portfolio volatility. The *negative* diversification ratio cost is therefore given as:

$$c_{DR}(\mathbf{z}, \mathbf{V}) = -\frac{\mathbf{z}^T \sqrt{\operatorname{diag}(\mathbf{V})}}{\sqrt{\mathbf{z}^T \mathbf{V} \mathbf{z}}}. \tag{6.25}$$

Direct minimization of the negative diversification ratio results in a non-convex optimization program. We follow the methodology described in Section 2.2 and recast the maximum-diversification optimization as a convex quadratic program. The maximum-diversification portfolio is then given by the solution to the following convex quadratic program:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\,\mathbf{z}^T\,\mathbf{V}\,\mathbf{z} \\ \text{subject to} \quad & \mathbf{z}^T\,\sqrt{\text{diag}(\mathbf{V})} = 1 \\ & (\mathbf{z}, k) \in \mathbb{S}_0 \end{aligned} \tag{6.26}$$

The full IPO formulation for maximum-diversification portfolio is presented in Program (6.27):

$$\begin{aligned} \underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \quad & \frac{1}{m}\sum_{i=1}^{m} c_{DR}(\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}), \mathbf{V}^{(i)}) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{V}}^{(i)}) = \underset{\mathbf{z}}{\text{argmin}}\,\frac{1}{2}\,\mathbf{z}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{z} \quad \forall i \in 1, ..., m \\ & \mathbf{z}^T\,\sqrt{\text{diag}(\hat{\mathbf{V}}^{(i)})} = 1 \\ & (\mathbf{z}, k) \in \mathbb{S}_0. \end{aligned} \tag{6.27}$$

6.2.10 IPO: equal-risk-contribution portfolio

The equal-risk-contribution portfolio is described in detail in Section 2.3. Recall that the equal risk contribution, also known as risk-parity, is attained when marginal risk contribution for each asset is equal. As described in Section 2.3, we define the Herfindahl index of risk contributions as follows:

$$c_{ERC}(\mathbf{z}, \mathbf{V}) = \sum_{j=1}^{m}\left(\frac{\mathbf{z}_j \odot (\mathbf{V}\,\mathbf{z})_j}{\mathbf{z}^T\,\mathbf{V}\,\mathbf{z}}\right)^2, \tag{6.28}$$

and recall that the values of $c_{ERC}(\mathbf{z}, \mathbf{V})$ range between $1/d_z$ for equal-risk-contributions, and 1 for a fully concentrated portfolio.

In Section 2.3 we demonstrate that the equal-risk-contribution portfolio can be obtained by solving the following convex optimization program:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\,\mathbf{z}^T\,\mathbf{V}\,\mathbf{z} - \sum_{j=1}^{d_z}\mathbf{r}_j \ln(-\,\mathbf{G}_{jj}\,\mathbf{z}_j) \\ \text{subject to} \quad & \mathbf{G}\,\mathbf{z} \leq 0 \end{aligned} \tag{6.29}$$

where $\mathbf{r} \in \mathbb{R}^{d_z}$ is a vector of risk-contribution weights (i.e. $\mathbf{r}_j = 1/d_z$ for equal risk contribution).

The diagonal matrix $\mathbf{G} \in \mathbb{R}^{d_z \times d_z}$ constrains the optimization to the relevant orthant of interest, with diagonal elements $G_{jj} \in \{-1, 1\}$. The full IPO formulation for ERC portfolio is presented in Program (6.30):

$$\begin{aligned} \underset{\boldsymbol{\theta} \in \Theta}{\text{minimize}} \quad & \frac{1}{m} \sum_{i=1}^{m} -c_{ERC}(\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}), \mathbf{V}^{(i)}) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{V}}^{(i)}) = \underset{\mathbf{z}}{\operatorname{argmin}} \frac{1}{2} \mathbf{z}^T \hat{\mathbf{V}}^{(i)} \mathbf{z} - \sum_{j=1}^{d_z} \mathbf{r}_j \ln(-\mathbf{G}_{jj} \mathbf{z}_j) \quad \forall i \in 1, ..., m \\ & \mathbf{G}\, \mathbf{z} \leq 0. \end{aligned} \tag{6.30}$$

6.3 Experiments

Sections 6.3.1 and 6.3.2 compare the IPO and OLS-ML methodology across two asset universes:

1. 10 U.S. industry sectors with testing from January 1980 to October 2020.

2. 254 U.S. liquid stocks with testing from January 1995 to December 2020.

In both sets of experiments, the minimum-variance portfolios with parameters estimated by IPO exhibit meaningfully lower out-of-samples costs, which in general, result in improved economic performance in comparison to the traditional 'predict, then optimize' approach. On the other-hand, the U.S. industry sector experiments with maximum-diversification (MD) and equal-risk-contribution (ERC) portfolios exhibit only a marginal reduction in the out-of-sample costs. As a result, we find that the realized economic impact of the IPO framework under these portfolio objectives is relatively immaterial. That said, we are encouraged that the results of the MD and ERC portfolios under the IPO framework are no worse than the traditional estimation approach, suggesting, perhaps, that the IPO method converges to prediction model parameterizations of equal integrity. Furthermore, we acknowledge that these findings may be particular to our chosen data sets, or alternatively may speak to the stability and resilience of the MD and ERC portfolios to errors in the covariance estimate [6]. To answer these questions, further testing is required. For completeness, the MD and ERC results are presented in Appendix B.

The minimum-variance results presented below are meant to serve as a proof-of-concept and to illustrate the potential benefit of the IPO framework in comparison to the traditional parameter estimation process, which is based on ordinary least-squares and maximum-likelihood (OLS-ML). All performance is gross of trading costs and in excess of the 13-week T-bill rate, provided by FRED, Federal Reserve Bank of St. Louis.

6.3.1 Experiment 1: U.S. industry data

We consider an asset universe of 10 U.S. industry sectors, described in Table 6.1. The weekly price data is given from January 1964 through October 2020, and is provided by the Kenneth R. French data library. We consider two factor models, based on the Famma-French Three (FF3) factor model and the Famma-French Five (FF5) factor model, respectively. Historical factor returns are also provided from the Kenneth R. French data library. The multivariate factor GARCH dynamics are modelled according to the CCC-GARCH and the DCC-GARCH. We refer the reader to Appendix B for more comprehensive implementation details. For each portfolio optimization we consider 4 covariance model instantiations:

1. FF3-CCC: Famma-French Three Factor with Constant Conditional Correlation GARCH.
2. FF5-CCC: Famma-French Five Factor with Constant Conditional Correlation GARCH.
3. FF3-DCC: Famma-French Three Factor with Dynamic Conditional Correlation GARCH.
4. FF5-DCC: Famma-French Five Factor with Dynamic Conditional Correlation GARCH.

All experiments start in January 1980 and end in October 2020. For each experiment, the first 15 years (January 1964 through December 1979) is used to perform the initial parameter estimation. Thereafter, we apply a walk-forward training and testing methodology. The optimal covariance model coefficients are updated every 2 years using all available data for parameter estimation and the optimal policy, $\mathbf{z}^*(\hat{\mathbf{V}}^{(i)})$, is then applied to the next out-of-sample 2-year segment. Portfolios are rebalanced approximately once a month. In order to minimize the impact of rebalance timing luck, the portfolios are formed at the close of each week, and we rebalance $1/4^{\text{th}}$ of the exposure on a weekly basis [71].

We evaluate the IPO and OLS-ML methodologies based on their respective out-of-sample realized cost values, as defined in Section 6.2. To quantify the magnitude and consistency of this results and to ensure our results are robust to potential outliers in the out-of-sample periods, we bootstrap the out-of-sample distribution using 1000 samples as follows:

1. For each $k \in \{1, 2, ..., 1000\}$, sample, without replacement, a batch, B_k, with $|B_k| = 52$ observations (1 year) from the out-of-sample period.
2. For both the IPO and OLS-ML methods, compute the average realized costs over the sample:

$$\bar{c}^{\text{IPO}}_{B_k} \frac{1}{|B_k|} \sum_{i \in B_k} c(\mathbf{z}^*(\hat{\mathbf{V}}(\boldsymbol{\theta}^*)^{(i)}, \mathbf{V}^{(i)}) \quad \text{and} \quad \bar{c}^{\text{OLS-ML}}_{B_k} \frac{1}{|B_k|} \sum_{i \in B_k} c(\mathbf{z}^*(\hat{\mathbf{V}}(\hat{\boldsymbol{\theta}})^{(i)}, \mathbf{V}^{(i)}),$$

where $\boldsymbol{\theta}^*$ and $\hat{\boldsymbol{\theta}}$ denote the IPO and OLS-ML optimal coefficients, respectively.

Note, in each sample draw we use the same observation dates across both methodologies in order to fairly compare the realized costs over the resulting sample. We report the dominance ratio (DR), which we define as the proportion of samples for which the realized cost of the IPO methodology is less than that of the OLS-ML methodology, presented in Equation (6.31):

$$DR = \frac{1}{N}\sum_{k=1}^{N} \mathbb{I}[\bar{c}_{B_k}^{\text{IPO}} < \bar{c}_{B_k}^{\text{OLS-ML}}], \tag{6.31}$$

where, $\mathbb{I}$ denotes the indicator function and $N = 1000$. We also evaluate performance based on well-documented economic and portfolio risk metrics, summarized in Appendix B.

Industry Short Name	Description
NoDur	**Consumer Nondurables:** food, tobacco, textiles, apparel, leather and toys.
Durbl	**Consumer Durables:** cars, TVs, furniture, and household appliances.
Manuf	**Manufacturing:** machinery, trucks, planes, chemicals, paper, and printing.
Enrgy	**Energy:** oil, gas, coal extraction and products.
HiTec	**Technology:** computers, software, and electronic equipment.
Telcm	**Telecommunications:** telephone and television transmission.
Shops	**Shops:** wholesale, retail, and services (laundries, repair shops).
Hlth	**Health:** healthcare, medical equipment, and pharmaceuticals.
Utils	**Utilities:** water, sewage services, and electricity.
Other	**Other:** mines, construction, transportation, hotels, entertainment, and finance.

Table 6.1: U.S. industry sector data, provided by the Kenneth French data library.

Economic performance metrics are provided in Table 6.2 for the time period of 1980-01-01 to 2020-10-31. We first note that, with the exception of the FF5-DCC model, the IPO method produces a lower long-term realized portfolio volatility. This result is encouraging for the IPO approach as it demonstrates that covariance prediction model parameters can be estimated effectively by a process that seeks to minimize the decision error induced by the estimate. Furthermore, for covariance models using the FF3 factors, the excess mean returns of the IPO and OLS-ML methods are comparable and therefore the IPO approach exhibits an increase in long-term out-of-sample Sharpe ratios. We observe that for the FF3 models, the IPO method provides more conservative portfolio risk metrics, as measured by the value-at-risk (VaR), average drawdown (Avg DD) and conditional drawdown-at-risk (CDaR). This however is not the case for the FF5 models in which

the IPO method exhibits marginally lower Sharpe ratios and larger risk metrics.

Statistic	IPO	OLS-ML
Excess Mean	0.0933	0.0937
Volatility	0.1423	0.1511
Sharpe	0.6556	0.6196
VaR	-0.0440	-0.0468
Avg DD	-0.0313	-0.0327
CDaR	-0.2091	-0.2386

(a) FF3-CCC

Statistic	IPO	OLS-ML
Excess Mean	0.07925	0.08566
Volatility	0.1430	0.1445
Sharpe	0.5541	0.5930
VaR	-0.0448	-0.0445
Avg DD	-0.0327	-0.0295
CDaR	-0.2190	-0.2030

(b) FF5-CCC

Statistic	IPO	OLS-ML
Excess Mean	0.0951	0.0932
Volatility	0.1390	0.1489
Sharpe	0.6840	0.6261
VaR	-0.0432	-0.0462
Avg DD	-0.0300	-0.0310
CDaR	-0.1983	-0.2211

(c) FF3-DCC

Statistic	IPO	OLS-ML
Excess Mean	0.0803	0.0888
Volatility	0.1418	0.1418
Sharpe	0.5661	0.6260
VaR	-0.0444	-0.0438
Avg DD	-0.0338	-0.0287
CDaR	-0.2296	-0.1980

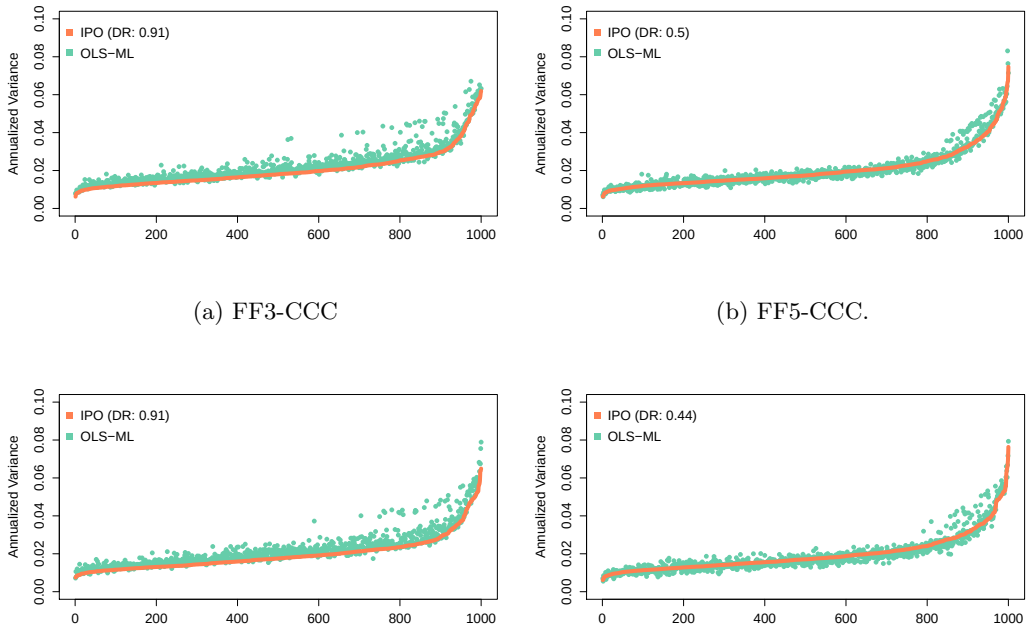
(d) FF5-DCC

Table 6.2: Out-of-sample economic and risk metrics of the IPO and OLS-ML methods for the constrained minimum-variance portfolio.

In Figure 6.2 we compare the realized portfolio variance costs across 1000 bootstrapped out-of-sample realizations, as described above. Observe that for the FF3 models, the IPO method provides, consistently lower realized portfolio variances. For both FF3 GARCH models, we report at dominance ratio 0.91, meaning that in 91% of samples, the IPO method realizes a lower portfolio variance than that of the OLS-ML method. In the case of the FF5 models, however, we do not observe a consistent reduction in realized portfolio variance. In fact, for the FF5-DCC model, the dominance ratio is 0.44, meaning that in 56% of samples, the OLS-ML method in-fact realizes a lower portfolio variance than that of the IPO method.

Figure 6.3 charts the average 52-week rolling difference in realized volatility between the IPO and OLS-ML methods. Note that volatility differences less than zero imply that the IPO method realized a lower average volatility in comparison to the OLS-ML method over that time period. For the FF3 models, we observe that, in normal market conditions, the difference in realized portfolio volatility across the two methods is small, with values oscillating around the zero line. Indeed, almost all of the benefit of the IPO framework is observed during periods of market crisis; notably

the Dot-com bubble and market downturn of the early 2000s (April 1999 through December 2002) and the Global Financial Crisis (January 2008 through December 2009). As we will show in Section 6.3.2, these observations are largely consistent with the experimental results on the individual U.S. stock universe.



(a) FF3-CCC

(b) FF5-CCC.

(c) FF3-DCC

(d) FF5-DCC.

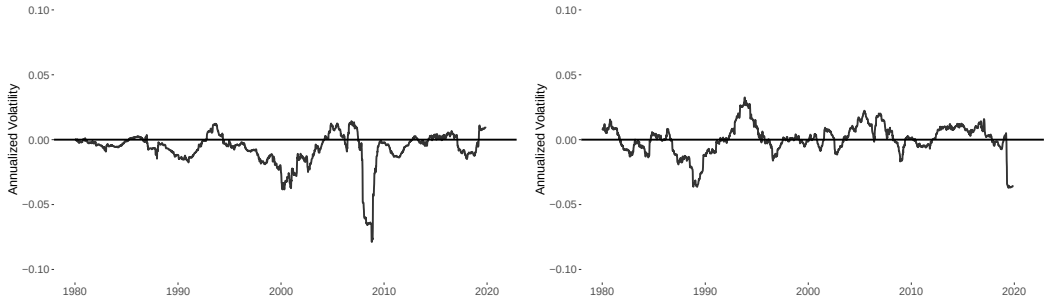
Figure 6.2: Out-of-sample variance cost of the IPO and OLS-ML methods for the constrained minimum-variance portfolio.

In contrast, the out-of-sample results for the FF5 models on the U.S. industry data set suggest no obvious preference for the IPO framework over the OLS-ML method. We hypothesize that the benefit of the IPO framework will be most observable when the chosen prediction model happens to be poorly specified. In our experiments, model misspecification can come from two primary sources:

1. **Factor model misspecification**: errors as a result of improper factor model that does not effectively explain the idiosyncratic variance of the assets. This can occur when using the less explanatory FF3 factor model or, as we will demonstrate in Section 6.3.2, when the number of assets in the portfolio (and thus unexplained variance) is large.

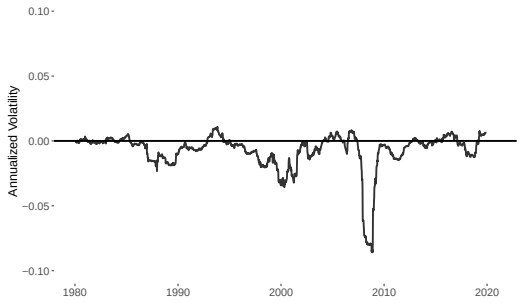
2. **Correlation model misspecification**: errors as a result of improper correlation model assumptions. For example, the CCC GARCH model assumes that asset correlations are constant whereas the DCC GARCH assumes time-varying correlations.

We conjecture that when the number of assets in the portfolio is small, as is the case for 10 U.S. industry universe, the FF5 factor model is well-specified and therefore the resulting decision errors induced by the covariance predictions is small relative to the more poorly specified FF3 models. Again, as we will demonstrate shortly, these observations are consistent with the results in Section 6.3.2 where the IPO framework exhibits a monotonic reduction in relative portfolio variance as the number of assets in the portfolio increases.

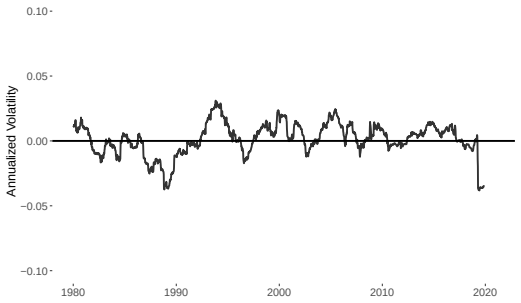


(a) FF3-CCC

(b) FF5-CCC.



(c) FF3-DCC



(d) FF5-DCC.

Figure 6.3: Rolling average 52-week difference in realized portfolio volatility between IPO and OLS-ML methods for the constrained minimum-variance portfolio.

6.3.2 Experiment 2: U.S. stock data

We follow the experimental design as outlined in Costa and Kwon [38]. We consider an asset universe of 254 liquid US stocks traded on major U.S. exchanges (NYSE, NASDAQ, AMEX, ARCA). The universe is summarized in Table B.3 in Appendix B.5, with representative stocks from each of the Global Industry Classification Standard (GICS) sectors. Weekly price data is given from January 1990 through December 2020, and is provided by Quandl.

We use the more explanatory Famma-French Five (FF5) factor model, with multivariate factor GARCH dynamics modelled according both to the CCC-GARCH and the DCC-GARCH. The goal is to observe the behaviour of the IPO framework in a larger scale setting. We consider minimum-

variance portfolios on an n asset universe, for $n \in (25, 50, 100)$. For each covariance model and asset size pair, we conduct a randomized trial experiment. Specifically, each experiment consists of 500 independent trials, where, at the beginning of each trial, a basket of n assets is randomly drawn from the universe of 254 assets. The basket of n assets is held constant throughout the duration of the trial.

All experiments start in January 1995 and end in December 2020, with the first 5 years used to perform the initial parameter estimation. We apply a walk-forward training and testing methodology whereby the optimal covariance model coefficients are updated every 5 years using all available data for parameter estimation. As before, portfolios are formed at the close of each week, and we rebalance $1/4^{\text{th}}$ of the exposure on a weekly basis.

In Table 6.3 we report the mean and standard deviation of excess mean returns, volatility and Sharpe ratios of out-of-sample returns for each covariance model and universe size pair. In all cases we observe that the IPO method produces a lower average out-of-sample volatility in comparison to the traditional OLS-ML approach. Furthermore, the IPO method exhibits larger average excess mean returns, therefore resulting in higher average Sharpe ratios. Additionally, we observe that the difference in realized portfolio variance across the two methods is greatest when the number of assets in the portfolio is largest ($n = 100$). Moreover, the difference in realized portfolio variance across the two methods is more significant for the CCC-GARCH models as opposed to the (perhaps) more well-specified DCC-GARCH models. These two observations are consistent with our prior claim that the benefit of IPO framework is most notable as prediction model misspecification increases. In other words, the IPO framework appears to be more resilient to model misspecification.

Figure 6.4 compares the average out-of-sample portfolio variance in each experiment measured over 500 independently generated trials. As before, we report the dominance ratio: the proportion of trials for which the IPO method realizes an out-of-sample variance smaller than that of the OLS-ML method. We observe that the IPO method provides consistently lower realized portfolio variances, with dominance ratios in the range of 58% to 100%, and increasing in proportion to the number of assets in the portfolio. Furthermore, the dominance ratios are generally larger for the CCC-GARCH models in comparison to the corresponding DCC-GARCH models at each fixed universe size. When the universe size is smallest ($n = 25$) the IPO method reports modest dominance ratios of 79% and 58% for the CCC-GARCH and DCC-GARCH, respectively. In practice, however, portfolio managers typically construct portfolios from a much larger pool of assets ($n \geq 50$). It is under these situations that we observe the greatest benefit of the IPO approach. When $n = 50$ we observe dominance ratios of 95% and 77%, whereas when $n = 100$ we observe dominance ratios of 100% and 91% for

$n = 25$

Statistic	IPO	OLS-ML
Excess Mean	0.1173	0.1146
σ_{Mean}	0.0097	0.0093
Volatility	0.1589	0.1636
σ_{Vol}	0.0105	0.0136
Sharpe	0.7405	0.7042
σ_{Sharpe}	0.0687	0.0687

(a) FF5-CCC

$n = 25$

Statistic	IPO	OLS-ML
Excess Mean	0.1178	0.1119
σ_{Mean}	0.0099	0.0097
Volatility	0.1595	0.1611
σ_{Vol}	0.0101	0.0133
Sharpe	0.7413	0.6983
σ_{Sharpe}	0.0743	0.0762

(b) FF5-DCC

$n = 50$

Statistic	IPO	OLS-ML
Excess Mean	0.1181	0.1160
σ_{Mean}	0.0079	0.0071
Volatility	0.1511	0.1587
σ_{Vol}	0.0067	0.0095
Sharpe	0.7827	0.7333
σ_{Sharpe}	0.0587	0.0551

(c) FF5-CCC

$n = 50$

Statistic	IPO	OLS-ML
Excess Mean	0.1190	0.1132
σ_{Mean}	0.0073	0.0078
Volatility	0.1508	0.1551
σ_{Vol}	0.0064	0.0094
Sharpe	0.7898	0.7318
σ_{Sharpe}	0.0552	0.0626

(d) FF5-DCC

$n = 100$

Statistic	IPO	OLS-ML
Excess Mean	0.1203	0.1182
σ_{Mean}	0.0065	0.0054
Volatility	0.1456	0.1555
σ_{Vol}	0.0045	0.0063
Sharpe	0.8265	0.7615
σ_{Sharpe}	0.0467	0.0428

(e) FF5-CCC

$n = 100$

Statistic	IPO	OLS-ML
Excess Mean	0.1207	0.1155
σ_{Mean}	0.0063	0.0055
Volatility	0.1454	0.1508
σ_{Vol}	0.0046	0.0054
Sharpe	0.8308	0.7668
σ_{Sharpe}	0.0463	0.0427

(f) FF5-DCC

Table 6.3: Out-of-sample economic and risk metrics of the IPO and OLS-ML methods for the constrained minimum-variance portfolio over 500 randomized trials.

the CCC-GARCH and DCC-GARCH models, respectively. Therefore, we posit that asset managers who construct minimum-variance portfolios from a large number of assets would likely benefit from the IPO framework.

Figure 6.6 charts the average 52-week rolling difference in realized volatility between the IPO and OLS-ML methods. The upper and lower shaded regions denote the 97.5%ile and 2.5% outcomes, respectively. Note that volatility differences less than zero imply that the IPO method realized a lower average volatility over that time period. As before, observe that in normal market conditions, the difference in realized portfolio volatility across the two methods is small, with values oscillating around the zero line. We therefore observe negligible difference in realized portfolio volatility by applying the integrated parameter estimation approach during regular market conditions. In contrast, almost all of the benefit of the IPO framework is observed during periods of market crisis. From the charts in Figure 6.6, we can clearly see three periods for which the IPO approach exhibits mean-

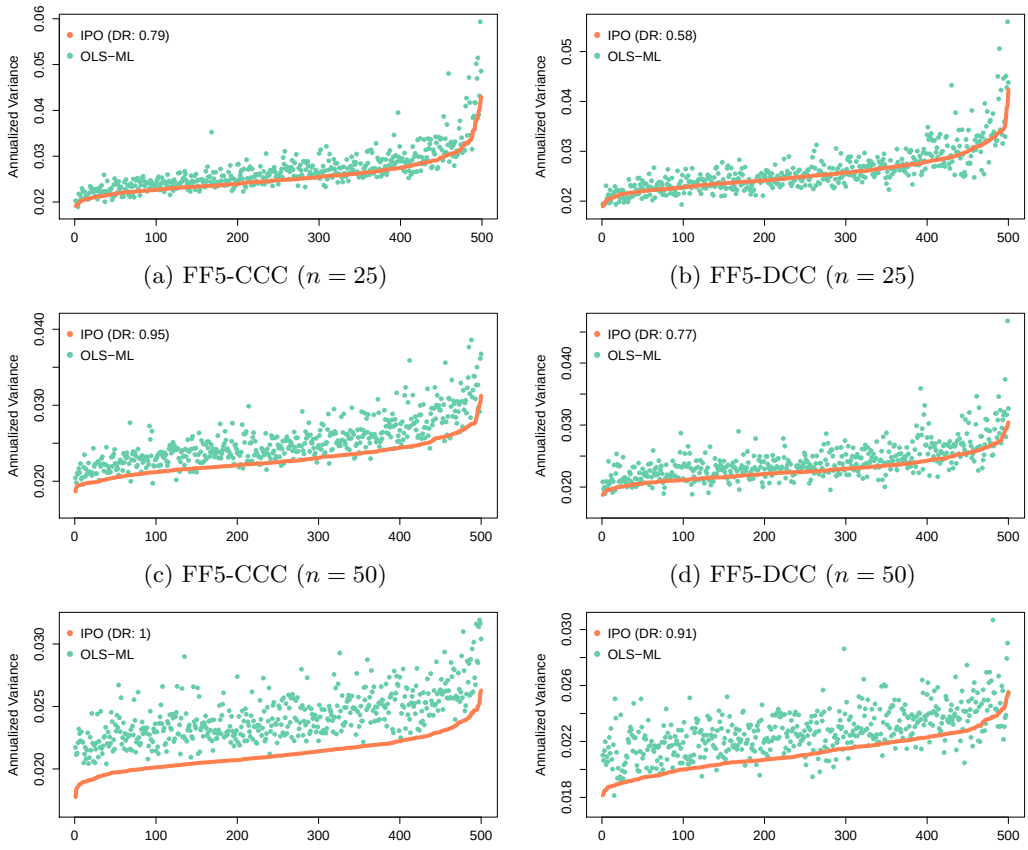
ingful lower portfolio volatility in comparison to the traditional OLS-ML approach. Those periods coincide with the three major U.S. market recessions that occurred during the out-of-sample period, specifically: The Dot-com bubble and market downturn of the early 2000s (April 1999 through December 2002), The Global Financial Crisis (January 2008 through December 2009) and the Global COVID-19 pandemic (March 2020 through December 2020).

Experiment	Dot-com Bubble Apr 1999 to Dec 2002	Global Financial Crisis Jan 2008 to Dec 2009	COVID-19 Pandemic Mar 2020 to Dec 2020
FF5-CCC (n = 25)	-0.0113	-0.0048	-0.0200
FF5-DCC (n = 25)	-0.0106	0.0111	-0.0183
FF5-CCC (n = 50)	-0.0140	-0.0115	-0.0330
FF5-DCC (n = 50)	-0.0141	0.0062	-0.0297
FF5-CCC (n = 100)	-0.0144	-0.0173	-0.0430
FF5-DCC (n = 100)	-0.0170	0.0130	-0.0442

Table 6.4: Average volatility difference between the IPO and OLS-ML methods during U.S. market recessions. Negative values imply that the IPO method realized a lower average volatility during that time period.

In Table 6.4 we provide the average difference in realized volatility over the three time periods of U.S. market recession. Again, values less than zero imply that the IPO method realized a lower average volatility over that time period. In general, we observe that the relative reduction in volatility provided by the IPO method increases as the number of assets in the portfolio increases. Furthermore, we observe that the IPO method exhibits a materially lower portfolio volatility across all experiments during both the Dot-com bubble and the COVID-19 pandemic, with average annualized volatility reductions in the range of 1.06% to 1.70% and 1.83% to 4.42%, respectively. The results during the Global Financial Crisis are mixed, with the IPO CCC-GARCH experiments exhibiting average volatility reduction in the range of 0.48% to 1.73%, whereas the IPO DCC-GARCH experiments actually exhibit larger realized volatility during that time period, with values ranging from 0.62% to 1.30%. Nonetheless, we find these larger scale results to be highly encouraging for the IPO framework and it demonstrates that during most periods of market crisis - when undoubtedly investors are most concerned with portfolio risk - there is a high probability that covariance models that are made aware of their downstream decision based optimizations will result in lower realized portfolio variance.

We conclude this section with a note on computational complexity. In contrast to the traditional approach, estimating prediction model parameters in an integrated setting can be computationally expensive; in particular as the number of problem variables, d_z, becomes large. In Figure 6.5 we compare the average runtime, in seconds, for the IPO and OLS-ML methods as a function of the



(a) FF5-CCC ($n = 25$) (b) FF5-DCC ($n = 25$)

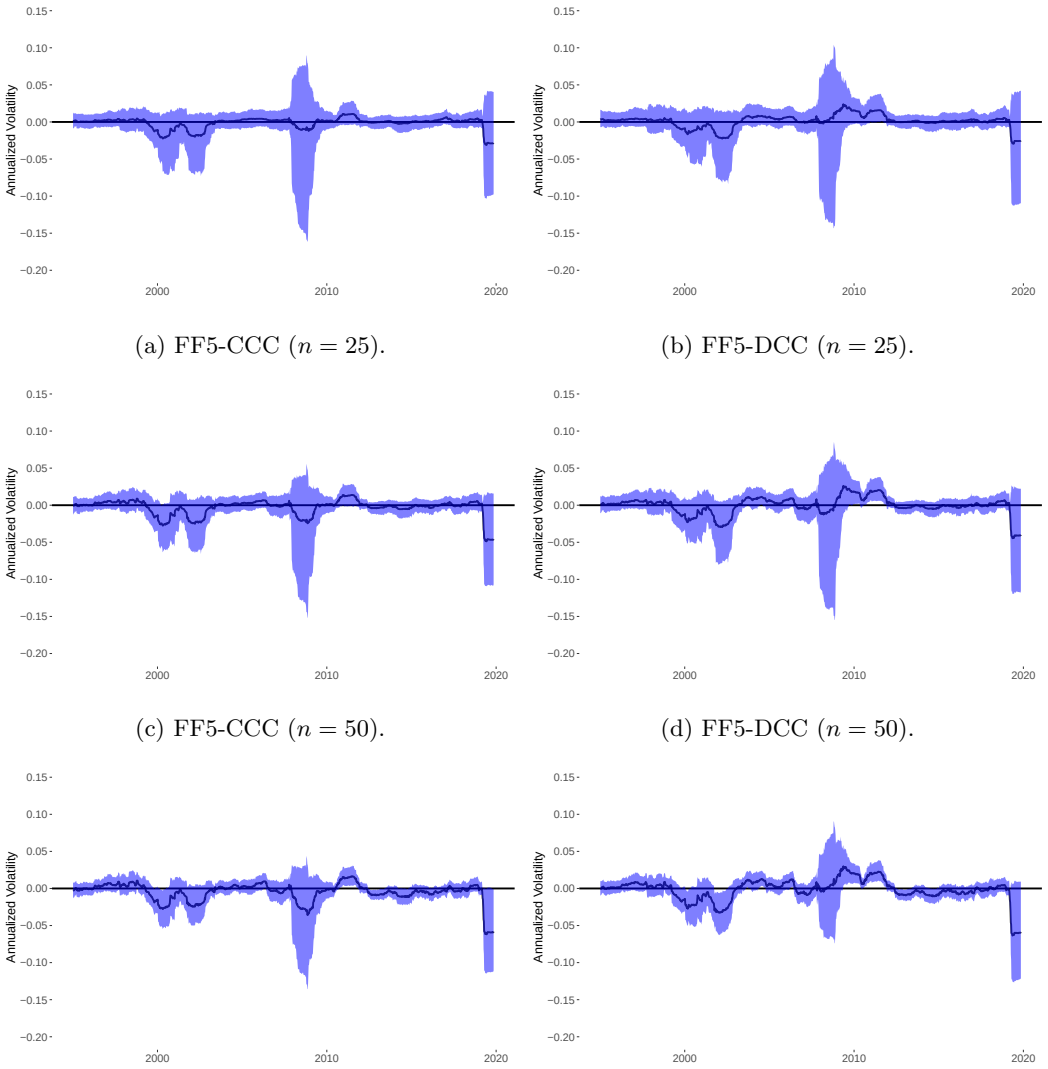
(c) FF5-CCC ($n = 50$) (d) FF5-DCC ($n = 50$)

(e) FF5-CCC ($n = 100$) (f) FF5-DCC ($n = 100$)

Figure 6.4: Out-of-sample variance of the IPO and OLS-ML methods for the constrained minimum-variance portfolio. Each experiment is evaluated over 500 randomized trials.

number of assets, $d_z \in (10, 25, 50, 100, 200)$. We fix the number of training observations to $m = 1000$ and set the maximum number of gradient descent iterations to $n = 25$, and acknowledge that the runtime will scale linearly as a function of these two parameters. We observe that for problems with a small number of assets ($d_z \leq 50$) the IPO runtime is competitive with that of the traditional OLS-ML approach. However, as the number of assets increase, the expected runtime of the IPO method increases considerably, with runtime values of 100 seconds and 800 seconds, for $n = 100$ and $n = 200$ respectively. We note that the observed increase in runtime is much less than the worst-case time-complexity bound. However, when $n = 200$ the expected runtime of the IPO method is a full order of magnitude larger than that of OLS-ML method. Therefore, for assets managers that construct portfolios from a very large pool of assets, estimating prediction model parameters by IPO can be computationally burdensome. Improving the efficiency of the integrated framework is therefore an open and interesting area of future research, explored in more detail in Chapter 7.

Figure 6.5: Average FF5 DCC-GARCH parameter estimation runtime for the IPO and OLS-ML methods, with training observations $m = 1000$ and number of gradient descent iterations $n = 25$. The 95%ile confidence intervals are measured over 30 independent evaluations.



(a) FF5-CCC ($n = 25$).

(b) FF5-DCC ($n = 25$).

(c) FF5-CCC ($n = 50$).

(d) FF5-DCC ($n = 50$).

(e) FF5-CCC ($n = 100$).

(f) FF5-DCC ($n = 100$).

Figure 6.6: Rolling average 52-week difference in realized portfolio volatility between IPO and OLS-ML methods for the constrained minimum-variance portfolio with 95%ile confidence interval over 500 randomized trials.

6.4 Conclusion and future work

In this chapter we proposed an integrated prediction and optimization (IPO) framework for covariance model parameter estimation in the context of several risk-based portfolio optimizations. Specifically, we structured the problem as a stochastic program where, for a fixed instantiation of model parameters, we solve a series of deterministic portfolio optimization programs. We investigated the IPO framework under four covariance model specifications and considered three portfolio optimizations: minimum-variance (MV), maximum-diversification (MD) and equal-risk-contribution (ERC).

Covariance model parameters are optimized locally using the first-order method described in Chapter 4 which restructures the IPO problem as a neural network with a differentiable convex programming layer. We provide the integrated formulation for the MV and MD portfolios, which are special cases of the general quadratic programs. The ERC portfolio is formulated as a convex optimization program and the IPO formulation and relevant gradient equations are provided. We performed several historical simulations using U.S. industry sectors and U.S. stock data and compared the IPO framework against a traditional 'predict, then optimize' framework whereby parameters are optimized by ordinary least-squares and maximum-likelihood. The experiments for minimum-variance portfolios demonstrate the benefits of the IPO framework, and provided a proof-of-concept that the integrated approach can result in lower out-of-sample realized cost values. Furthermore, we observe that prediction models that are more poorly specified exhibit the greatest relative improvement in realized cost values and economic metrics, such as Sharpe ratios. This result is encouraging as it demonstrates the resilience of the IPO framework to model misspecification.

In the case of the MD and ERC portfolios, we observe that the IPO method provides a consistent reduction in realized out-of-sample costs. In both cases, however, the magnitude of the cost reduction is small and does not result in a material improvement in economic outcomes. We hypothesize that these observed results may be specific to our particular choice of data, or alternatively speak to the stability of the MD and ERC portfolio construction process. Further testing with alternative data sets, and under varying prediction model and portfolio constraint assumptions is required in order to better determine the efficacy of the IPO approach under these portfolio decision settings. Future work also includes incorporating other forms of prediction models into the IPO framework as well as exploring the integrated prediction and optimization under a robust portfolio optimization framework.

Chapter 7

Efficient differentiable quadratic programming layers: an ADMM approach

7.1 Introduction

In chapters 5 and 6 we observed that, in many cases, training a prediction model in the context of the downstream optimization problem can be computationally demanding, with computational performance degrading as the number of assets in the portfolio increases. In practice, asset managers typically construct portfolios from a relatively large pool of assets and estimate prediction models using thousands of training examples. We have observed, however, that for medium to large scale problems, with $100 - 1000$ decision variables and thousands of training examples, the IPO estimation process can be several orders of magnitude more computationally expensive than a traditional 'predict, then optimize' approach. In this chapter we present a neural network architecture that, from a computational efficiency standpoint, improves upon the current state-of-the-art solution.

Specifically, the IPO neural network architecture described in chapters 5 and 6 relied on a specialized differentiable optimization layer (OptNet) that uses a primal-dual interior-point method for solving batch quadratic programs [3]. The OptNet layer is computationally efficient and practical for small-scale problems ($d_z < 100$). However, as described in Chapter 5, solving convex quadratic programs exactly by interior-point methods has worst-case time complexity on the order of $\mathcal{O}(d_z^3)$

[65]. Therefore, for medium ($100 < d_z < 1000$) and large ($d_z \geq 1000$) scale problems, embedding an OptNet layer in a larger neural network can be computationally intractable.

In this chapter, we address the computational challenges in the medium to large scale limit and present an alternative differentiable neural network architecture for batch constrained quadratic programs. We consider parametric quadratic programs (PQPs) with linear equality and box inequality constraints:

$$\begin{array}{ll} \underset{\mathbf{z}}{\text{minimize}} & \frac{1}{2}\mathbf{z}^T\,\mathbf{Q}(\boldsymbol{\theta})\,\mathbf{z} + \mathbf{z}^T\,\mathbf{p}(\boldsymbol{\theta}) \\ \text{subject to} & \mathbf{A}(\boldsymbol{\theta})\,\mathbf{z} = \mathbf{b}(\boldsymbol{\theta}), \quad \mathbf{l}(\boldsymbol{\theta}) \leq \mathbf{z} \leq \mathbf{u}(\boldsymbol{\theta}) \end{array} \tag{7.1}$$

Here $\mathbf{z} \in \mathbb{R}^{d_z}$ denotes the decision variable and $\mathbf{Q}(\boldsymbol{\theta})$, $\mathbf{p}(\boldsymbol{\theta})$, $\mathbf{A}(\boldsymbol{\theta})$, $\mathbf{b}(\boldsymbol{\theta})$, $\mathbf{l}(\boldsymbol{\theta})$, $\mathbf{u}(\boldsymbol{\theta})$ are the parameterized problem variables. Program (7.1) occurs in many applications to statistics [118, 119], machine-learning [92, 70], signal-processing [80] and finance [14, 99, 96]. Indeed, the mean-variance, maximum return, minimum variance, maximum Sharpe ratio and maximum diversification ratio portfolios described in Chapter 2 are special cases of Program (7.1).

Our differentiable quadratic programming layer is built on top of the alternating direction method of multipliers (ADMM) algorithm, which recently has become increasingly popular for solving large scale convex optimization problems [21, 103, 112, 115, 109]. Existing ADMM layer architectures [42, 127, 128] perform backpropagation by unrolling the ADMM computational graph, which requires substantially larger networks and, under certain circumstances, can be computationally demanding. Alternatively, many differentiable convex optimization layers perform backpropagation by implicit differentiation of the KKT optimality conditions. However, KKT implicit differentiation requires solving a system of equations of dimension $\mathbb{R}^{3d_z \times 3d_z}$, which for large scale problems can also be computationally impractical. As an alternative, we present a novel and efficient backward-pass routine that implicitly differentiates a customized fixed-point mapping. A primary advantage is that the fixed-point iteration is of dimension d_z and therefore the resulting system of equations is approximately 3 times smaller than the equivalent KKT system. Finally, our differentiable ADMM layer and all algorithmic implementations is made available as an open-source R package, available here:

https://github.com/butl3ra/lqp

The remainder of the chapter is outlined as follows. We begin with a brief discussion of related work in the field of differentiable convex optimization layers. In Section 7.2 we review the ADMM algorithm and present our ADMM-based architecture for forward solving batch PQPs. We review

the KKT based implicit differentiation and then present the fixed-point iteration and derive the expression for the relevant gradients. In Section 7.3 we compare the computational efficiency of our ADMM layer with the OptNet layer and a splitting cone solver (SCS) layer based on the *cvxpylayers* implementation [1, 103]. We demonstrate that for medium scale problems, our ADMM layer implementation is approximately an order of magnitude faster than both the OptNet and SCS layers. Moreover, we compare the computational efficiency of the backpropagation routines based on unrolled differentiation, KKT implicit differentiation and fixed-point implicit differentiation. We demonstrate that the fixed-point implicit differentiation is universally more efficient than the KKT implicit differentiation, and under certain conditions is preferred to the unrolled differentiation. We conclude with real world applications of the ADMM layer to medium scale portfolio optimization problems.

7.1.1 Literature review

Our ADMM layer is motivated by the OptNet layer [3], which implements a primal-dual interior-point method for solving small scale batch constrained quadratic programs and performs backpropagation by implicit differentiation of the KKT optimality conditions. For reasons mentioned earlier, the author's acknowledge that the OptNet layer may be impractical for optimization problems with a moderate number of variables. We refer the reader to Chapter 4 for a more comprehensive overview of state-of-the-art differentiable optimization layers.

Perhaps most closely related to our work, the ADMM-Net and its generalizations (D-LADMM) [128, 127], directly embed the iterative ADMM procedure as a fully learnable network graph. However, the ADMM-Net implementation does not support inequality constraints and moreover performs argmin differentiation by unrolling the 'inner-loop' of the ADMM routine, which necessitates substantially larger and less efficient networks [3, 4].

To our knowledge, our ADMM layer implementation is the first of its kind that can efficiently handle medium to large scale constrained PQPs. The forward pass solves batch PQPs by applying the ADMM algorithm in batch form. Backpropagation is performed by implicit differentiation of a custom fixed-point mapping. A primary advantage of the fixed-point differentiation is that the fixed-point scheme is of dimension d_z and is therefore approximately 3 times smaller than the corresponding KKT system. In the absence of a pre-factorized KKT system, the fixed-point implicit differentiation is preferred to KKT implicit differentiation and is competitive with the efficient KKT factorization caching provided in the OptNet layer. While unrolled differentiation

of the ADMM algorithm is appropriate for small scale problems, for larger scale problems or for problems that require solving the convex optimization problem to a high degree of accuracy, an unrolled differentiation approach can also be computationally demanding. In contrast, the fixed-point implicit differentiation method is shown to be computationally efficient and invariant to the number of 'inner' iterations performed in the ADMM forward-pass. Lastly, our implementation is not without its own limitations and areas for improvement, discussed in detail in Section 7.4.

7.2 Methodology

7.2.1 ADMM algorithm

The alternating direction method of multipliers (ADMM) algorithm, first proposed by Gabay and Mercier [62] and Glowinski and Marroco [63], is well suited to many large-scale and distributed problems common to applications of statistics, machine learning, control and finance. We note that the ADMM algorithm is closely related to algorithms such as dual ascent, the augmented Lagrangian method of multipliers, and operator (Douglas–Rachford) splitting and refer to Boyd et al. [21] for a comprehensive overview.

In general, the ADMM algorithm is applied to problems of the form:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{z} = \mathbf{c} \end{aligned} \tag{7.2}$$

with decision variables $\mathbf{x} \in \mathbb{R}^{d_x}$, $\mathbf{z} \in \mathbb{R}^{d_z}$ and problem variables $\mathbf{A} \in \mathbb{R}^{d_{eq} \times d_x}$, $\mathbf{B} \in \mathbb{R}^{d_{eq} \times d_z}$ and $\mathbf{c} \in \mathbb{R}^{d_{eq}}$. In order to guarantee convergence we assume that $f\colon \mathbb{R}^{d_x} \to \mathbb{R}$ and $g\colon \mathbb{R}^{d_z} \to \mathbb{R}$ are closed, proper convex functions [21]. The augmented Lagrangian of Program (7.2) is given by:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\lambda}^T(\mathbf{r}) + \frac{\rho}{2}\|\mathbf{r}\|_2^2, \tag{7.3}$$

with Lagrange dual variable $\boldsymbol{\lambda}$, residual $\mathbf{r} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{z} - \mathbf{c}$ and user-defined penalty parameter $\rho > 0$. We denote $\boldsymbol{\mu} = \rho^{-1}\,\boldsymbol{\lambda}$ and therefore the well-known scaled ADMM iterations are as follows:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{z}^k - \mathbf{c} + \boldsymbol{\mu}^k\|_2^2 \\
\mathbf{z}^{k+1} &= \underset{\mathbf{z}}{\operatorname{argmin}}\, g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{A}\,\mathbf{x}^{k+1} + \mathbf{B}\,\mathbf{z} - \mathbf{c} + \boldsymbol{\mu}^k\|_2^2 \\
\boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \mathbf{A}\,\mathbf{x}^{k+1} + \mathbf{B}\,\mathbf{z}^{k+1} - \mathbf{c}
\end{aligned} \tag{7.4}$$

where $\mathbf{x}^k$ and $\mathbf{z}^k$ denote the decision variables at iteration k.

We denote $\mathbf{r}^k = \mathbf{A}\,\mathbf{x}^k + \mathbf{B}\,\mathbf{z}^k - \mathbf{c}$ and $\mathbf{s}^k = \rho\,\mathbf{A}^T\,\mathbf{B}(\mathbf{z}^k - \mathbf{z}^{k-1})$ as the primal and dual residual at iteration k. Let $\epsilon^p > 0$ and $\epsilon^d > 0$ be the user defined stopping tolerances for the primal and dual residuals, respectively. Therefore a reasonable stopping criteria would be:

$$\mathbf{r}^k \leq \epsilon^p \quad \text{and} \quad \mathbf{s}^k \leq \epsilon^d. \tag{7.5}$$

7.2.2 ADMM for parametric quadratic programs

We consider convex parametric quadratic programs (PQPs) of the form:

$$\begin{aligned}
\underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\,\mathbf{z}^T\,\mathbf{Q}(\boldsymbol{\theta})\,\mathbf{z} + \mathbf{z}^T\,\mathbf{p}(\boldsymbol{\theta}) \\
\text{subject to} \quad & \mathbf{A}(\boldsymbol{\theta})\,\mathbf{z} = \mathbf{b}(\boldsymbol{\theta}), \quad \mathbf{l}(\boldsymbol{\theta}) \leq \mathbf{z} \leq \mathbf{u}(\boldsymbol{\theta}),
\end{aligned} \tag{7.6}$$

with decision variable $\mathbf{z} \in \mathbb{R}^{d_z}$. The objective function is therefore defined by a vector $\mathbf{p}(\boldsymbol{\theta}) \in \mathbb{R}^{d_z}$ and symmetric positive definite matrix $\mathbf{Q}(\boldsymbol{\theta}) \in \mathbb{R}^{d_z \times d_x}$. Here, $\mathbf{A}(\boldsymbol{\theta}) \in \mathbb{R}^{d_{eq} \times d_z}$, $\mathbf{b}(\boldsymbol{\theta}) \in \mathbb{R}^{d_{eq}}$, $\mathbf{l}(\boldsymbol{\theta}) \in \mathbb{R}^{d_z}$ and $\mathbf{u}(\boldsymbol{\theta}) \in \mathbb{R}^{d_z}$ define the linear equality and box inequality constraints. We assume that all problem variables are parameterized by $\boldsymbol{\theta}$ and are therefore trainable when integrated in an end-to-end neural network; rather than simply being supplied by the user.

7.2.3 ADMM layer: forward-pass

Our ADMM layer solves Program (7.6) in the forward-pass by applying the ADMM algorithm as outlined in Section 7.2.1. For ease of notation we temporarily drop the parameterization, $\boldsymbol{\theta}$.

We define

$$f(\mathbf{x}) = \frac{1}{2}\,\mathbf{x}^T\,\mathbf{Q}\,\mathbf{x} + \mathbf{x}^T\,\mathbf{p}, \tag{7.7}$$

with domain $\{\mathbf{x} \,|\, \mathbf{A}\,\mathbf{x} = \mathbf{b}\}$. Similarly, we define

$$g(\mathbf{z}) = \mathbb{I}_{\mathbf{l} \leq \mathbf{z} \leq \mathbf{u}}(\mathbf{z}) \tag{7.8}$$

where $\mathbb{I}_{\mathbf{l} \leq \mathbf{z} \leq \mathbf{u}}(\mathbf{z})$ denotes the indicator function with respect to the linear inequality constraints. Program (7.6) is then recast to the following convex optimization Program:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x} - \mathbf{z} = 0. \end{aligned} \tag{7.9}$$

Applying the ADMM iterations, as defined by Equations (7.4), to Program (7.9) therefore gives the following iterative optimization algorithm:

$$\mathbf{x}^{k+1} = \underset{\{\mathbf{x} \,|\, \mathbf{A}\,\mathbf{x} = \mathbf{b}\}}{\operatorname{argmin}} \frac{1}{2}\mathbf{x}^T\,\mathbf{Q}\,\mathbf{x} + \mathbf{x}^T\,\mathbf{p} + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}^k + \boldsymbol{\mu}^k\|_2^2 \tag{7.10a}$$

$$\mathbf{z}^{k+1} = \underset{\{\mathbf{l} \leq \mathbf{z} \leq \mathbf{u}\}}{\operatorname{argmin}} \frac{\rho}{2}\|\mathbf{x}^{k+1} - \mathbf{z} + \boldsymbol{\mu}^k\|_2^2 \tag{7.10b}$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1} \tag{7.10c}$$

The ADMM algorithm, as defined by Equations (7.10), allows for efficient optimization of medium and large scale quadratic programs. Firstly, we note that (7.10b) is a least squares problem with box-inequality constraints, and therefore can be solved analytically. Specifically, we define the euclidean projection onto a set of box constraints as:

$$\Pi(\mathbf{x}) = \begin{cases} \mathbf{l}_j & \text{if } \mathbf{x}_j < \mathbf{l}_j \\ \mathbf{x}_j & \text{if } \mathbf{l}_j \leq \mathbf{x}_j \leq \mathbf{l}_j \\ \mathbf{u}_j & \text{if } \mathbf{x}_j > \mathbf{u}_j \end{cases}. \tag{7.11}$$

The analytic solution to Program (7.10b) is therefore given by:

$$\mathbf{z}^{k+1} = \Pi(\mathbf{x}^{k+1} + \boldsymbol{\mu}^k). \tag{7.12}$$

Furthermore, Program (7.10a) is an equality constrained quadratic program, which can also be solved analytically. Specifically, the KKT optimality conditions of Program (7.10a) can be expressed as

the solution to the following linear system of equations:

$$\begin{bmatrix} \mathbf{Q} + \rho \mathbf{I}_{\mathbf{x}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{k+1} \\ \boldsymbol{\eta}^{k+1} \end{bmatrix} = - \begin{bmatrix} \mathbf{p} - \rho(\mathbf{z}^k - \boldsymbol{\mu}^k) \\ -\mathbf{b} \end{bmatrix}, \tag{7.13}$$

with identity matrix $\mathbf{I}_{\mathbf{x}} \in \mathbb{R}^{d_x \times d_x}$. Applying Equations (7.12) and (7.13) allows us to express the ADMM iterations in a simplified form:

$$\begin{bmatrix} \mathbf{x}^{k+1} \\ \boldsymbol{\eta}^{k+1} \end{bmatrix} = - \begin{bmatrix} \mathbf{Q} + \rho \mathbf{I}_{\mathbf{x}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \rho(\mathbf{z}^k - \boldsymbol{\mu}^k) \\ -\mathbf{b} \end{bmatrix} \tag{7.14a}$$

$$\mathbf{z}^{k+1} = \Pi(\mathbf{x}^{k+1} + \boldsymbol{\mu}^k) \tag{7.14b}$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1} \tag{7.14c}$$

Observe that the per-iteration cost of the ADMM algorithm is dominated by solving the system of Equations (7.13). Note, however, that this linear system is in general smaller than the Newton system found in a standard primal-dual interior-point solvers by a factor of approximately 5, and therefore remains tractable for medium and large scale problems. Furthermore, if ρ is static then the left-hand side matrix in Equation (7.13) remains unchanged at each iteration and therefore is factorized only once at the onset of the ADMM algorithm. Furthermore, as we demonstrate below, this matrix factorization can subsequently be cached and invoked during backpropogation to compute the relevant gradients. Lastly, we note that if the matrices $\mathbf{Q}$ and $\mathbf{A}$ remain unchanged at each epoch of gradient descent, then the left-hand-side matrix needs to only be factorized **once** during the entire training process.

7.2.4 ADMM layer: backward-pass

ADMM layer: unrolled differentiation

Note that the standard unrolled differentiation approach, as described in Section 4.2, 'unrolls' the iterations in Equation (7.14) by standard backpropogation and requires that each operation in Equation (7.14) be differentiable. We refer to Domke [43] and Diamond et al. [42] for a more comprehensive overview of unrolled differentiation. Our unrolled differentiation is invoked by the standard auto-differentiation routine in the *torch* library.

ADMM layer: KKT implicit differentiation

As an alternative to unrolled differentiation, we note that the system of equations provided by the KKT conditions at optimality is a fixed-point mapping. It is therefore possible to apply the implicit function theorem to the KKT system of equations at the optimal primal-dual solution to Program (7.6). For completeness, we restate the KKT implicit differentiation equations and refer the reader to Section 4.3 for a more comprehensive overview.

For constrained quadratic programming, let us denote the primal-dual solution at optimality by $\boldsymbol{\nu}^* = (\mathbf{z}^*, \tilde{\boldsymbol{\lambda}}^*, \boldsymbol{\eta}^*)$, where $\mathbf{z}^*$ and $\boldsymbol{\eta}^*$ are defined by Equations (7.13) at optimality. Note that the dual variables associated with the inequality constraints are given by $\tilde{\boldsymbol{\lambda}}^* = (\boldsymbol{\lambda}_-^*, \boldsymbol{\lambda}_+^*)$ with:

$$\boldsymbol{\lambda}_-^* = -\min(\rho\,\boldsymbol{\mu}^*, 0) \quad \text{and} \quad \boldsymbol{\lambda}_+^* = \max(\rho\,\boldsymbol{\mu}^*, 0). \tag{7.15}$$

We define the box inequality constraints as $\mathbf{G}\,\mathbf{z} \leq \mathbf{h}$, where

$$\mathbf{G} = \begin{bmatrix} -\mathbf{I_x} \\ \mathbf{I_x} \end{bmatrix} \quad \text{and} \quad \mathbf{h} = \begin{bmatrix} -\mathbf{l} \\ \mathbf{u} \end{bmatrix}.$$

Note that all constraints are affine and therefore Slater's condition reduces to feasibility. The KKT conditions for stationarity, primal feasibility, and complementary slackness therefore defines a fixed-point at optimality $\boldsymbol{\nu}^*$ given by:

$$G(\boldsymbol{\nu}^*, \boldsymbol{\theta}) = \begin{bmatrix} \mathbf{p} + \mathbf{Q}\,\mathbf{z}^* + \mathbf{G}^T\,\tilde{\boldsymbol{\lambda}}^* + \mathbf{A}^T\,\boldsymbol{\eta}^* \\ \operatorname{diag}(\tilde{\boldsymbol{\lambda}}^*)(\mathbf{G}\,\mathbf{z}^* - \mathbf{h}) \\ \mathbf{A}\,\mathbf{z}^* - \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{7.16}$$

Applying Theorem 1 and simplifying as described in Section 4.3 gives the following system of equations for implicitly computing the gradient of the optimal solution with respect to the problem input variables:

$$\begin{bmatrix} \bar{\mathbf{d}}_{\mathbf{z}} \\ \bar{\mathbf{d}}_{\boldsymbol{\lambda}} \\ \bar{\mathbf{d}}_{\boldsymbol{\eta}} \end{bmatrix} = -\begin{bmatrix} \mathbf{Q} & \mathbf{G}^T \operatorname{diag}(\tilde{\boldsymbol{\lambda}}^*) & \mathbf{A}^T \\ \mathbf{G} & \operatorname{diag}(\mathbf{G}\,\mathbf{z}^* - \mathbf{h}) & 0 \\ \mathbf{A} & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \left(\frac{\partial \ell}{\partial\, \mathbf{z}^*}\right)^T \\ 0 \\ 0 \end{bmatrix}. \tag{7.17}$$

Note that we never explicitly form the right-side Jacobian matrix directly and instead we compute

the left matrix-vector product of the Jacobian with the previous backward-pass gradient, $\partial \ell / \partial \, \mathbf{z}^*$, as outlined below. Equation (7.17) allows for efficient computation of the gradients with respect to any of the relevant problem variables. This is particularly true when using interior-point methods as the required gradients are effectively obtained 'for free' upon factorization of the left matrix when obtaining the solution, $\mathbf{z}^*$, in the forward-pass. In the ADMM algorithm, however, we never explicitly solve the KKT system of equations and therefore we must form and factorize the left-side KKT matrix during the backward pass routine. Observe that the left-side matrix in Equation (7.17) is of dimension $3d_z + d_{eq}$ and therefore for large scale problems this system of equations can be computationally burdensome to solve.

7.2.5 ADMM layer: fixed-point implicit differentiation

In this section we demonstrate that the ADMM iterations in Equation (7.14) can be cast as a fixed-point iteration of dimension $d_z + d_{eq}$. In many applications, d_{eq} is typically much smaller than d_z, and therefore the proposed fixed-point differentiation will almost certainly decrease the computational overhead of the backward-pass routine. We begin with the following proposition. Note that all proofs are available in the Appendix.

Proposition 10. *Let* $\mathbf{v}^k = \mathbf{x}^{k+1} + \boldsymbol{\mu}^k$, $\tilde{\mathbf{v}}^k = (\mathbf{v}^k, \boldsymbol{\eta}^k)$ *and define* $F \colon \mathbb{R}^{d_{\tilde{v}}} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_{\tilde{v}}}$. *Then the ADMM iterations in Equation* (7.14) *can be cast as a fixed-point iteration of the form* $F(\tilde{\mathbf{v}}, \boldsymbol{\theta}) = \tilde{\mathbf{v}}$ *given by:*

$$\begin{bmatrix} \mathbf{v}^{k+1} \\ \boldsymbol{\eta}^{k+2} \end{bmatrix} = - \begin{bmatrix} \mathbf{Q} + \rho \, \mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \rho(2\Pi(\mathbf{v}^k) - \mathbf{v}^k) \\ -\mathbf{b} \end{bmatrix} + \begin{bmatrix} \mathbf{v}^k \\ \boldsymbol{\eta}^{k+1} \end{bmatrix} - \begin{bmatrix} \Pi(\mathbf{v}^k) \\ \boldsymbol{\eta}^{k+1} \end{bmatrix}. \tag{7.18}$$

We follow Busseti et al. [25] and define the derivative of the projection operator, Π as:

$$D\Pi(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x}_j < \mathbf{l}_j \\ 1 & \text{if } \mathbf{l}_j \leq \mathbf{x}_j \leq \mathbf{l}_j \\ 0 & \text{if } \mathbf{x}_j > \mathbf{u}_j \end{cases}. \tag{7.19}$$

Observe that $D\Pi(\mathbf{x})$ is not continuously differentiable when $\mathbf{x}_j = \mathbf{l}_j$ or $\mathbf{x}_j = \mathbf{u}_j$. In practice, we can overcome the non-differentiability of Π by introducing a small perturbation to $\mathbf{x}$, thus moving $\mathbf{x}$ away from the boundaries. Alternatively, smooth sigmoid based approximations to $\Pi(\mathbf{x})$ may also be suitable. In all experiments below, however, we invoke $D\Pi(\mathbf{x})$ directly as defined in Equation

(7.19).

The Jacobian, $\nabla_{\tilde{\mathbf{v}}} F$, is therefore defined as:

$$\nabla_{\tilde{\mathbf{v}}} F = -\begin{bmatrix} \mathbf{Q} + \rho\,\mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} -\rho(2D\Pi(\mathbf{v}) - \mathbf{I}_{\mathbf{v}}) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathbf{I}_{\mathbf{v}} & 0 \\ 0 & \mathbf{I}_{\boldsymbol{\eta}} \end{bmatrix} - \begin{bmatrix} D\Pi(\mathbf{v}) & 0 \\ 0 & \mathbf{I}_{\boldsymbol{\eta}} \end{bmatrix}. \tag{7.20}$$

Corollary 1 therefore gives the desired Jacobian, $\nabla_{\boldsymbol{\theta}} \tilde{\mathbf{v}}(\boldsymbol{\theta})$, with respect to the parameter $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} \tilde{\mathbf{v}}(\boldsymbol{\theta}) = [\mathbf{I}_{\tilde{\mathbf{v}}} - \nabla_{\tilde{\mathbf{v}}} F(\tilde{\mathbf{v}}(\boldsymbol{\theta}), \boldsymbol{\theta})]^{-1} \nabla_{\boldsymbol{\theta}} F(\tilde{\mathbf{v}}(\boldsymbol{\theta}), \boldsymbol{\theta}) \tag{7.21}$$

From the definition of $\mathbf{v}$ we have that the Jacobians $\nabla_{\boldsymbol{\theta}}\, \mathbf{x}(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}}\, \boldsymbol{\eta}(\boldsymbol{\theta})$ are given by:

$$\begin{bmatrix} \nabla_{\boldsymbol{\theta}}\, \mathbf{x}(\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}}\, \boldsymbol{\eta}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} D\Pi(\mathbf{v}) & 0 \\ 0 & \mathbf{I}_{\boldsymbol{\eta}} \end{bmatrix} \Big[\mathbf{I}_{\tilde{\mathbf{v}}} - \nabla_{\tilde{\mathbf{v}}} F(\tilde{\mathbf{v}}(\boldsymbol{\theta}), \boldsymbol{\theta})\Big]^{-1} \nabla_{\boldsymbol{\theta}} F(\tilde{\mathbf{v}}(\boldsymbol{\theta}), \boldsymbol{\theta}) \tag{7.22}$$

As before we never form the Jacobians $\nabla_{\boldsymbol{\theta}}\, \mathbf{x}(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}}\, \boldsymbol{\eta}(\boldsymbol{\theta})$ directly. Instead, we compute the left matrix-vector product of the Jacobian with the previous backward-pass gradient, $\partial \ell / \partial\, \mathbf{z}^*$, as outlined below.

Proposition 11. *Let $\hat{\mathbf{d}}_{\mathbf{x}}$ and $\hat{\mathbf{d}}_{\boldsymbol{\eta}}$ be defined as:*

$$\begin{aligned} \begin{bmatrix} \hat{\mathbf{d}}_{\mathbf{x}} \\ \hat{\mathbf{d}}_{\boldsymbol{\eta}} \end{bmatrix} &= \begin{bmatrix} \mathbf{Q} + \rho\,\mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}^{-1} \Big[\mathbf{I}_{\tilde{\mathbf{v}}} - \nabla_{\tilde{\mathbf{v}}} F(\tilde{\mathbf{v}}(\boldsymbol{\theta}), \boldsymbol{\theta})\Big]^{-T} \begin{bmatrix} D\Pi(\mathbf{v}) & 0 \\ 0 & \mathbf{I}_{\boldsymbol{\eta}} \end{bmatrix} \begin{bmatrix} \big(-\frac{\partial \ell}{\partial\, \mathbf{z}^*}\big)^T \\ 0 \end{bmatrix} \\ &= \left[\begin{bmatrix} D\Pi(\mathbf{v}) & 0 \\ 0 & \mathbf{I}_{\boldsymbol{\eta}} \end{bmatrix} \begin{bmatrix} \mathbf{Q} + \rho\,\mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} + \begin{bmatrix} -\rho(2D\Pi(\mathbf{v}) - \mathbf{I}_{\mathbf{v}}) & 0 \\ 0 & 0 \end{bmatrix} \right]^{-1} \begin{bmatrix} D\Pi(\mathbf{v}) & 0 \\ 0 & \mathbf{I}_{\boldsymbol{\eta}} \end{bmatrix} \begin{bmatrix} \big(-\frac{\partial \ell}{\partial\, \mathbf{z}^*}\big)^T \\ 0 \end{bmatrix}. \end{aligned} \tag{7.23}$$

Then the gradients of the loss function, ℓ, with respect to problem variables $\mathbf{Q}$, $\mathbf{p}$, $\mathbf{A}$ and $\mathbf{b}$ are given by: .

$$\begin{aligned} \frac{\partial \ell}{\partial\, \mathbf{Q}} &= \frac{1}{2}\Big(\hat{\mathbf{d}}_{\mathbf{x}}\, \mathbf{x}^{*T} + \mathbf{x}^*\, \hat{\mathbf{d}}_{\mathbf{x}}^T\Big) & \frac{\partial \ell}{\partial\, \mathbf{p}} &= \hat{\mathbf{d}}_{\mathbf{x}} \\ \frac{\partial \ell}{\partial\, \mathbf{A}} &= \hat{\mathbf{d}}_{\boldsymbol{\eta}}\, \mathbf{x}^{*T} + \boldsymbol{\eta}^*\, \hat{\mathbf{d}}_{\mathbf{x}}^T & \frac{\partial \ell}{\partial\, \mathbf{b}} &= -\hat{\mathbf{d}}_{\boldsymbol{\eta}} \end{aligned} \tag{7.24}$$

Computing the gradients of the loss with respect to the box constraint variables, $\mathbf{l}$ and $\mathbf{u}$ is also straightforward.

Proposition 12. *Define* $\tilde{\boldsymbol{\mu}}^*$ *and* $\hat{\mathbf{d}}_{\boldsymbol{\lambda}}$ *as:*

$$\tilde{\boldsymbol{\mu}}_j^* = \begin{cases} \boldsymbol{\mu}_j^* & \text{if } \boldsymbol{\mu}_j^* \neq 0 \\ 1 & \text{otherwise,} \end{cases} \tag{7.25}$$

and

$$\hat{\mathbf{d}}_{\boldsymbol{\lambda}} = \operatorname{diag}(\rho\tilde{\boldsymbol{\mu}}^*)^{-1}\Big(-\Big(\frac{\partial \ell}{\partial \mathbf{z}^*}\Big)^T - \mathbf{Q}\,\hat{\mathbf{d}}_{\mathbf{x}} - \mathbf{A}^T\,\hat{\mathbf{d}}_{\boldsymbol{\eta}}\Big). \tag{7.26}$$

Then the gradients of the loss function, ℓ*, with respect to problem variables* $\mathbf{l}$ *and* $\mathbf{u}$ *are given by:*

$$\frac{\partial \ell}{\partial \mathbf{l}} = \operatorname{diag}(\boldsymbol{\lambda}_-^*)\hat{\mathbf{d}}_{\boldsymbol{\lambda}} \qquad \frac{\partial \ell}{\partial \mathbf{u}} = -\operatorname{diag}(\boldsymbol{\lambda}_+^*)\hat{\mathbf{d}}_{\boldsymbol{\lambda}}. \tag{7.27}$$

We now have a framework for computing the gradient with respect to all problem variables by implicit differentiation of the fixed-point mapping of the transformed ADMM iterations. We re-iterate that the implicit differentiation of the KKT conditions requires solving a system of equations on the order of $3d_z + d_{eq}$. In contrast, the fixed-point iteration, presented in Equation (7.18) is of dimension $d_z + d_{eq}$. As we will demonstrate shortly, reducing the dimension of the fixed-point mapping results in a considerable improvement in computational efficiency in the backward-pass.

7.3 Experiments

We present several experimental results that highlight the computational efficiency and performance accuracy of the ADMM layer. In all experiments, computational efficiency is measured by the average runtime (in seconds), required to implement the forward-pass and backward-pass algorithms of each model. We compare across 5 methods:

1. **ADMM FP:** ADMM in the forward-pass and fixed-point implicit differentiation in the backward-pass.
2. **ADMM KKT:** ADMM in the forward-pass and KKT implicit differentiation in the backward-pass.
3. **ADMM Unroll:** ADMM in the forward-pass and unrolled differentiation in the backward-pass.
4. **OptNet:** primal-dual interior-point method in the forward-pass and efficient KKT implicit differentiation in the backward-pass.

5. **SCS:** serial version of splitting cone solver (SCS) [103] in the forward-pass and fixed-point implicit differentiation in the backward-pass. See [1] and [102] for more details.

The forward-pass of each solver terminate when the median L_2-norm of the primal and dual residuals are sufficiently small (i.e. less than some user-defined tolerance ϵ_{tol}). In many applications, however, it is not always necessary to solve the batch QPs exactly during training. We therefore consider and compare the computational efficiency of each model over several stopping tolerances: $\epsilon_{\text{tol}} \in \{10^{-1}, 10^{-3}, 10^{-5}\}$. Going forward, the model label 'ADMM FP 3', for example, denotes the ADMM FP model with a stopping tolerance of 10^{-3}.

Lastly, first-order methods are known to be vulnerable to ill-conditioned problems and the resulting convergence rates can vary significantly when the data and algorithm parameters (ρ) are poorly scaled. Many first-order solvers therefore implement a preconditioning and problem scaling initialization step (see for example [103, 109, 115]). In our case, however, the QP problem variables are parameterized and are therefore expected to change at each epoch. Scaling and conditioning the problem variables at each epoch would potentially result in excessive computational overhead. As a result, our ADMM layer implementation does not include a preconditioning step. Instead, in all experiments presented below, we normalize the problem data to have approximately unit standard deviation (on average) and manually scale the problem variables: $\mathbf{p}$, $\mathbf{Q}$, $\mathbf{A}$ and $\mathbf{b}$ where appropriate. We find that for unit-scaled problem data, a value of $\rho \in \{0.10, 1.0\}$ provides a consistent rate of convergence. Indeed, an efficient and dynamic preconditioning and scaling routine is an interesting area of future research.

All experiments are conducted on an Apple Mac Pro computer (2.7 GHz 12-Core Intel Xeon E5,128 GB 1066 MHz DDR3 RAM) running macOS 'Catalina'. The software was written using the R programming language (version 4.0.0) and torch (version 0.6.0).

7.3.1 Experiment 1: ADMM layer performance

We conduct an experiment comparing the computational efficiency of the ADMM, OptNet and SCS methods with various stopping tolerances. We randomly generate problem data of dimension:

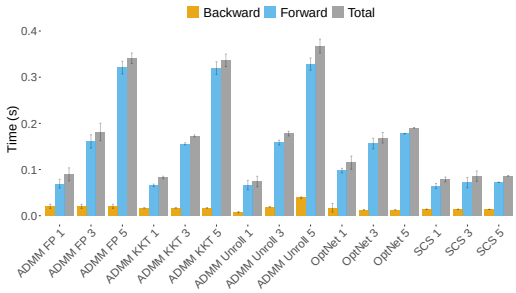
$$d_z \in \{10, 50, 100, 250, 500, 1000\}$$

and for each trial implement the forward and backward algorithms on a mini-batch size of 128. Problem variables are generated as follows. We set $\mathbf{Q} = \frac{1}{2d_z}\mathbf{U}^T\mathbf{U}$ where entries of $\mathbf{U} \in \mathbb{R}^{2d_z \times d_z}$ are sampled from a standard normal distribution. We randomly generate $\mathbf{p}$ by sampling from the

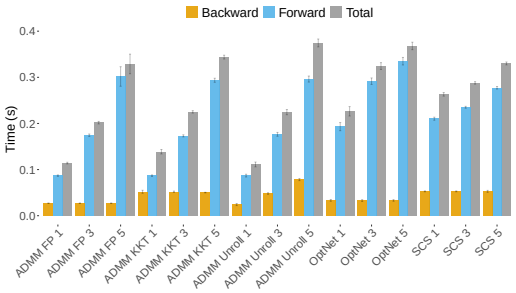
standard normal distribution and randomly generate $\mathbf{l}$ and $\mathbf{u}$ by randomly sampling from the uniform distribution with domain $[-2, -1]$ and $[1, 2]$, respectively . Finally we set $\mathbf{A} = \mathbf{1}$ and $\mathbf{b} = 1$.

Figure 7.1 provides the average runtime and 95%-ile confidence interval, evaluated over 10 trials, of the forward and backward-pass algorithms. We refer the reader to Table C.1 in Appendix C.4.1 for a summary of the relative computational efficiency benchmarked against the ADMM FP method. We make several important observations. First, for small scale problems ($d_z \leq 100$), there is negligible performance differences across all methods of the same stopping tolerance. As expected, the total runtime increases as the required stopping tolerance decreases. For medium to large scale problems ($100 < d_z \leq 1000$) we observe a substantial performance degradation in the ADMM KKT, OptNet and SCS methods, in comparison to the ADMM FP and ADMM Unroll methods. Specifically, the OptNet and SCS methods exhibit an increase in total computation time that is anywhere from 6.8x to 17.6x larger than the corresponding ADMM FP implementation. For example, when $d_z = 1000$ and $\epsilon_{\text{tol}} = 10^{-3}$, the total runtime for the OptNet and SCS methods is 185 seconds and 155 seconds, respectively, whereas the total runtime for the ADMM method is less than 11 seconds; over an order of magnitude faster. This increase in computational performance will ultimately enable training architectures that can practically support substantially larger quadratic optimization problems. Furthermore, the ADMM KKT backward computation time is shown to be 6.9x to 13.5x larger than the corresponding fixed-point method. This result is not surprising as the ADMM KKT backward-pass algorithm must first form and then factorize the KKT system of equations, which is of dimension $3d_z + d_{eq}$ whereas the fixed-point backward-pass routine solves a system of equations of size $d_z + d_{eq}$.

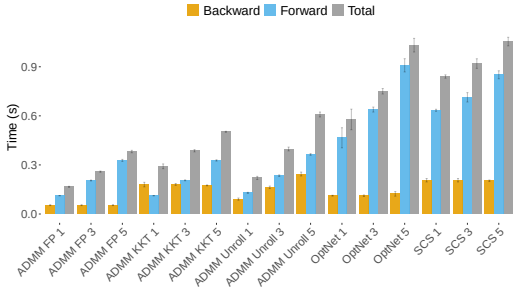
Lastly, we note that the ADMM Unroll method is relatively efficient for small scale problems. Observe, however, that for medium scale problems, as the stopping tolerance decreases the computation time of the unrolled backward-pass increases and is anywhere from 1.1x to 11.3x slower than the fixed-point method. In contrast, ADMM FP is invariant to the number of 'inner' iterations performed in the forward-pass, resulting in a 1.0x to 3.8x improvement in total computation time. Going forward, we choose to work with the ADMM FP method as it is most efficient for medium and large scale problems. Moreover, the OptNet and SCS methods provide very similar performance, particularly when $\epsilon_{\text{tol}} > 10^{-5}$. All subsequent experiments therefore compare the ADMM FP and the OptNet with $\epsilon_{\text{tol}} > 10^{-5}$.



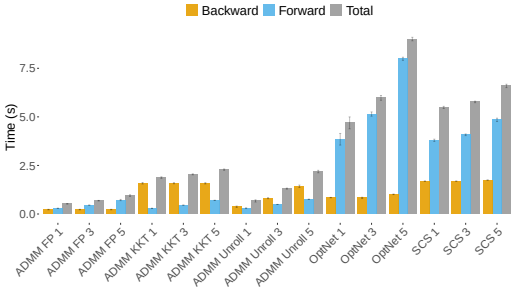
(a) $d_z = 10$.



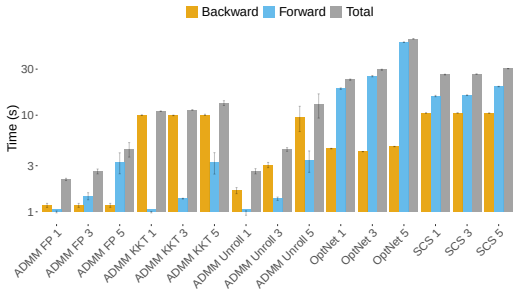
(b) $d_z = 50$.



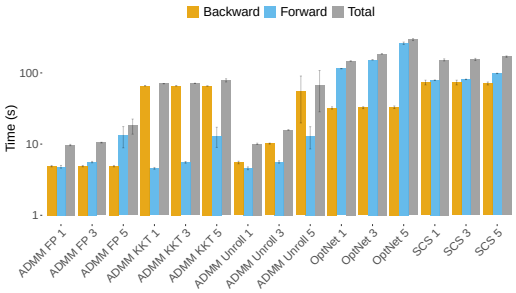
(c) $d_z = 100$.



(d) $d_z = 250$.



(e) $d_z = 500$.



(f) $d_z = 1000$.

Figure 7.1: Computational performance of ADMM-FP, ADMM-KKT, ADMM-Unroll, Optnet and SCS for various problem sizes, d_z, and stopping tolerances. Batch size = 128.

7.3.2 Experiment 2: learning p

We now consider a full training experiment whereby the objective is to learn a parameterized model for the variable $\mathbf{p}$. The resulting integrated predict and optimize (IPO) program is presented below.

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{z}^*(\boldsymbol{\theta})^{T(i)}\,\mathbf{p}^{(i)} + \frac{1}{2}\,\mathbf{z}^*(\boldsymbol{\theta})^{T(i)}\,\mathbf{Q}^{(i)}\,\mathbf{z}^*(\boldsymbol{\theta})^{(i)}\right) \tag{7.28}$$

$$\text{subject to} \quad \mathbf{z}^*(\boldsymbol{\theta})^{(i)} = \underset{\mathbf{z}}{\text{argmin}} - \mathbf{z}^T\,\hat{\mathbf{p}}(\boldsymbol{\theta})^{(i)} + \frac{1}{2}\,\mathbf{z}^T\,\hat{\mathbf{Q}}^{(i)}\,\mathbf{z} \quad \forall i = 1, ..., m \tag{7.29}$$

$$\mathbf{A}\,\mathbf{z}^*(\boldsymbol{\theta})^{(i)} = \mathbf{b} \quad \forall i = 1, ..., m \tag{7.30}$$

$$\mathbf{l} \leq \mathbf{z}^*(\boldsymbol{\theta})^{(i)} \leq \mathbf{u} \quad \forall i = 1, ..., m. \tag{7.31}$$

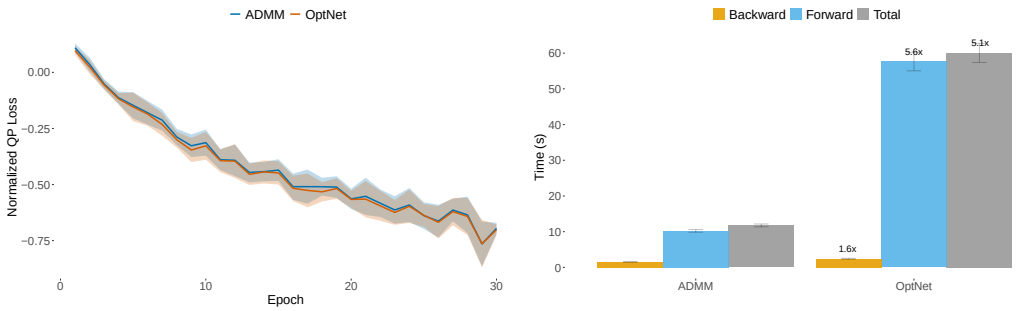
Here $\mathbf{p}^{(i)}$ and $\mathbf{Q}^{(i)}$ denote the ground truth problem data and are generated as follows. We let $\mathbf{p}^{(i)} \sim \mathcal{N}(\mathbf{w}^{T(i)}\,\boldsymbol{\theta}_0 + \tau\,\boldsymbol{\epsilon}^{(i)}, \mathbf{Q})$ where $\mathbf{Q} \in \mathbb{R}^{d_z \times d_z}$ has entry (j,k) equal to $\rho_{\mathbf{p}}^{|j-k|}$. We set $\rho_{\mathbf{p}} = 0.50$ and generate the feature data from the standard normal distribution, $\mathbf{w}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{\mathbf{w}})$. The residuals, $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, preserve the desired correlation structure and the scalar value τ controls the signal-to-noise level. All experiments target a signal-to-noise level of 0.10.

We let $\hat{\mathbf{p}}(\boldsymbol{\theta})^{(i)}$ denote the estimate of $\mathbf{p}^{(i)}$ according to the linear model:

$$\hat{\mathbf{p}}(\boldsymbol{\theta})^{(i)} = \mathbf{w}^{T(i)}\,\boldsymbol{\theta}\,. \tag{7.32}$$

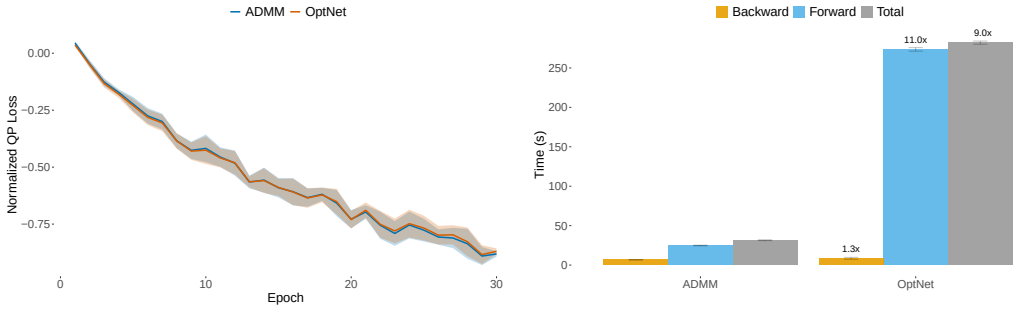
The box constraints, $\mathbf{l}$ and $\mathbf{u}$, are generated by randomly sampling from the uniform distribution with domain $[-1, 0]$ and $[0, 1]$, respectively and we set $\mathbf{A} = \mathbf{1}$ and $\mathbf{b} = 1$. In all experiments we set the stopping tolerance to $\epsilon_{\text{tol}} = 10^{-3}$.

We randomly generate problem data of dimension $d_z \in \{250, 500, 1000\}$. The training process for each trial consists of 30 epochs with a mini-batch size of 32. Figures 7.2(a) - 7.4(a) report the average training loss at each epoch and the 95%-ile confidence interval, evaluated over 10 independent trials. Observe that the loss curves for the ADMM model and OptNet model are almost identical at each epoch, suggesting that the training accuracy of the ADMM model and OptNet model are equivalent. Conversely, Figures 7.2(b) - 7.4(b), compare the average and 95%-ile confidence interval time spent executing the forward and backward pass algorithms. When $d_z = 250$ the ADMM model is shown to be approximately 5 times faster than the OptNet model. Furthermore, when $d_z = 500$ and $d_z = 1000$, the ADMM model is a full order of magnitude faster than the OptNet model. More concretely, when $d_z = 1000$ the entire learning process takes approximately 1600 seconds to train the OptNet model, but less than 130 seconds to train the ADMM model to an equal level of accuracy.



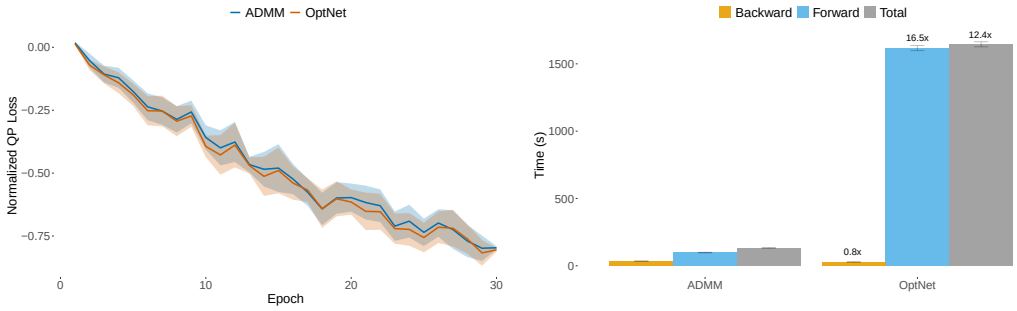
(a) Training Loss. (b) Computational Performance.

Figure 7.2: Training loss and computational performance for learning $\mathbf{p}$. Batch size $= 32$ and $d_z = 250$.



(a) Training Loss (b) Computational Performance

Figure 7.3: Training loss and computational performance for learning $\mathbf{p}$. Batch size $= 32$ and $d_z = 500$.



(a) Training Loss (b) Computational Performance

Figure 7.4: Training loss and computational performance for learning $\mathbf{p}$. Batch size $= 32$ and $d_z = 1000$.

7.3.3 Experiment 3: learning A

We now present a real-world experiment from portfolio optimization whereby the objective is to learn a parameterized model for the variable $\mathbf{A}$. We consider an asset universe of $d_z = 254$ liquid

US stocks traded on major U.S. exchanges (NYSE, NASDAQ, AMEX, ARCA). The universe is summarized in Table B.3, with representative stocks from each of the Global Industry Classification Standard (GICS) sectors. Weekly price data is given from January 1990 through December 2020, and is provided by Quandl.

We denote the matrix of weekly return observations as $\mathbf{A} = [\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, ..., \mathbf{a}^{(m)}] \in \mathbb{R}^{m \times d_z}$ with $m > d_z$. Let $\mathbf{Q}^{(i)} \in \mathbb{R}^{d_z \times d_z}$ denote the symmetric positive definite covariance matrix of asset returns. We define the portfolio $\mathbf{z}^{(i)} \in \mathbb{R}^{d_z}$, where the element, ${\mathbf{z}^{(i)}}_j$, denotes the proportion of total capital invested in the j^{th} asset at time i.

We define the Sharpe ratio at observation i as the ratio of portfolio return to portfolio risk, where risk is measured by the portfolio volatility (standard deviation).

$$\mathrm{S_R}^{(i)} = \frac{\mathbf{a}^{T(i)}\,\mathbf{z}^{(i)}}{\sqrt{\mathbf{z}^{T(i)}\,\mathbf{Q}^{(i)}\,\mathbf{z}^{(i)}}} \tag{7.33}$$

We consider a long-only ($\mathbf{z}^{(i)} \geq 0$), fully invested ($\mathbf{1}^T\,\mathbf{z}^{(i)} = 1$) max-Sharpe portfolio optimization, described in detail in Chapter 2 and presented below:

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}}\; & \frac{1}{2} \quad \mathbf{z}^T\,\mathbf{Q}^{(i)}\,\mathbf{z} \\ \text{subject to} \quad & \mathbf{a}^{T(i)}\,\mathbf{z} = 1, \quad \mathbf{z} \geq 0. \end{aligned} \tag{7.34}$$

Note that the fully-invested constraint can be enforced by normalizing the optimal weights: $\mathbf{z}^* \,/\, \sum_j \mathbf{z}_j^*$.

As before, the learning process can be posed as a bi-level optimization program where the objective is to learn a parameter $\boldsymbol{\theta}$ and the associated constraints, $\hat{\mathbf{a}}(\boldsymbol{\theta})^{(i)}$, in order to maximize the average realized Sharpe ratio induced by the optimal decision policies $\{\mathbf{z}^*(\boldsymbol{\theta})^{(i)}\}_{i=1}^m$.

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & -\frac{1}{m}\sum_{i=1}^{m} \frac{\mathbf{a}^{T(i)}\,\mathbf{z}^*(\boldsymbol{\theta})}{\sqrt{\mathbf{z}^*(\boldsymbol{\theta})^{T(i)}\,\mathbf{Q}^{(i)}\,\mathbf{z}^*(\boldsymbol{\theta})^{(i)}}} \\ \text{subject to} \quad & \mathbf{z}^*(\boldsymbol{\theta}) = \underset{\mathbf{z}}{\text{argmin}}\, \frac{1}{2}\,\mathbf{z}^T\,\mathbf{Q}^{(i)}\,\mathbf{z} \quad \forall i = 1, ..., m \\ & \hat{\mathbf{a}}(\boldsymbol{\theta})^{T(i)}\,\mathbf{z}^*(\boldsymbol{\theta})^{(i)} = 1, \quad \mathbf{1}^T\,\mathbf{z}^*(\boldsymbol{\theta})^{(i)} = 1, \quad \mathbf{z}^*(\boldsymbol{\theta})^{(i)} \geq 0 \quad \forall i = 1, ..., m \end{aligned} \tag{7.35}$$

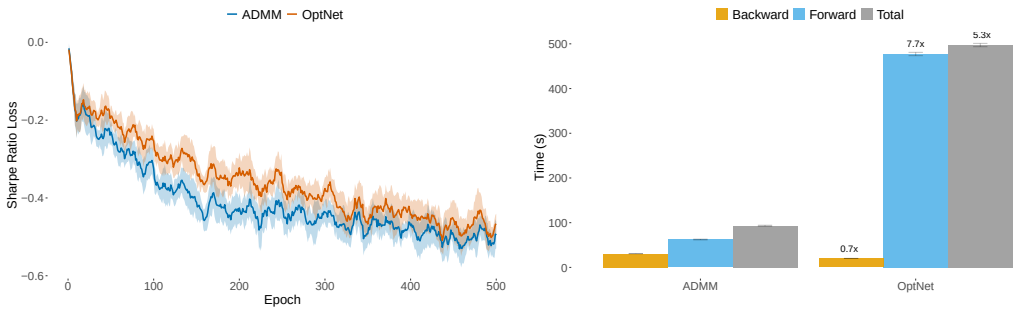
In reality, we do not know the true value of $\mathbf{a}^{(i)}$ at decision time and instead we estimate $\mathbf{a}^{(i)}$ through associated feature variables $\mathbf{w}^{(i)} \in \mathbb{R}^{d_w}$. Again we consider a linear model of the form:

$$\hat{\mathbf{a}}^{(i)} = \boldsymbol{\theta}^T\,\mathbf{w}^{(i)}\,. \tag{7.36}$$

In this experiment, asset returns, $\mathbf{a}^{(i)}$ are modelled using the well-known Famma-French Five (FF5) factor model [57], provided by the Kenneth R. French data library.

The goal is to observe the training and out-of-sample performance of the ADMM model in comparison to the OptNet model. As a benchmark, we include the out-of-sample performance of an equally weighted portfolio, and a max-Sharpe portfolio where $\boldsymbol{\theta}$ is fit by ordinary least-squares (OLS). All experiments are trained on data from January 1990 through December 2014. The out-of-sample period begins in January 2015 and ends in December 2020. Portfolios are formed at the close of each week, and rebalanced on a weekly basis.

The training process for each trial consists of 500 epochs with a mini-batch size of 32. Portfolio models are fit to an accuracy of $\epsilon_{\text{tol}} = 10^{-4}$ in training, and a higher accuracy of $\epsilon_{\text{tol}} = 10^{-6}$ in the out-of-sample period in order to guarantee strict adherence to the constraint set. Figure 7.5(a) reports the average training loss at each epoch and the 95%-ile confidence interval, evaluated over 10 independent trials. Once again, we observe that the loss curves for the ADMM model and OptNet model are very similar. Interestingly, we observe that the ADMM model produces a consistently lower average training loss. Recall that both models use implicit differentiation to compute the relevant gradient, which assumes an exact fixed point at each optimal solution $\mathbf{z}^*(\boldsymbol{\theta})^{(i)}$. In practice, each $\mathbf{z}^*(\boldsymbol{\theta})^{(i)}$ is only approximately optimal, to within a tolerance ϵ_{tol}, and therefore differentiating at a solution that is not an exact fixed point will result in small errors in the gradient that likely explain the observed difference. That said, the training loss profile of the ADMM and OptNet models are very similar, and the final models achieve approximately equal loss after 500 epochs. Figure 7.5(b) compares the average and 95%-ile confidence interval of the total time spent executing the forward and backward pass algorithms during training. Once again we observe that the ADMM model is shown to be approximately 5 times faster than the OptNet model and requires less than 100 seconds to train.



(a) Training Loss (b) Computational Performance

Figure 7.5: Training loss and computational performance for learning $\mathbf{A}$ on US stock data. Batch size $= 32$ and $d_z = 254$.

Figure 7.6 reports the out-of-sample equity growth of the ADMM IPO max-Sharpe portfolio, Equal Weight portfolio, OLS max-Sharpe portfolio and OptNet IPO max-Sharpe portfolio. The out-of-sample economic performance metrics are reported in Table 7.1. First, observe that all max-Sharpe models outperform the Equal Weight benchmark on an absolute and risk-adjusted basis. Furthermore, the ADMM and OptNet IPO max-Sharpe models achieve an out-of-sample Sharpe ratio that is approximately 50% higher than that of the naive 'predict, then optimize' OLS max-Sharpe model, thus highlighting the benefit of the IPO approach. Lastly, the ADMM model achieves a marginally higher out-of-sample Sharpe ratio in comparison to the OptNet model, though the difference is not statistically significant.

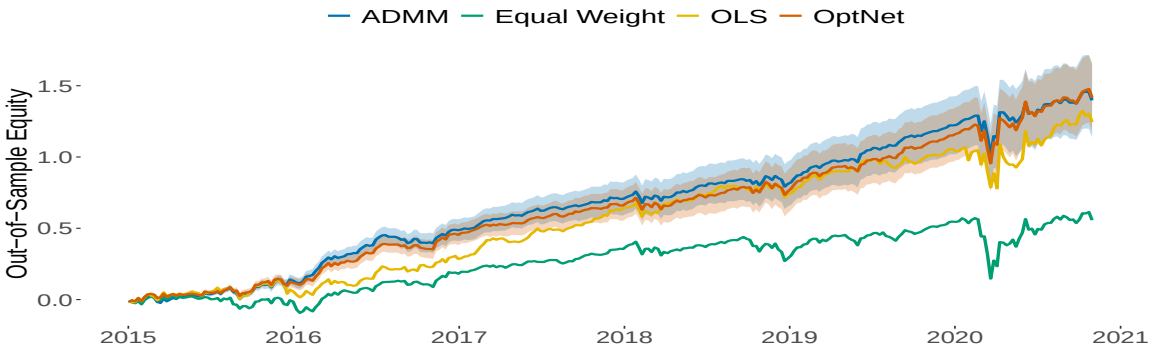


Figure 7.6: Out-of-sample equity growth for ADMM IPO max-Sharpe portfolio, Equal Weight portfolio, OLS max-Sharpe portfolio and OptNet IPO max-Sharpe portfolio.

	ADMM	Equal Weight	OLS	OptNet
Mean	0.2382	0.0950	0.2122	0.2413
Volatility	0.1777	0.1950	0.2435	0.1880
Sharpe Ratio	1.3407	0.4872	0.8747	1.2836

Table 7.1: Out-of-sample economic performance metrics for ADMM IPO max-Sharpe portfolio, Equal Weight portfolio, OLS max-Sharpe portfolio and OptNet IPO max-Sharpe portfolio.

7.3.4 Experiment 4: learning Q

We consider another real-world experiment from portfolio optimization whereby the objective is to learn a parameterized model for the variable $\mathbf{Q}$. We use the same asset universe of $d_z = 254$ liquid US stocks from Experiment 3 and an identical experimental design. Here, we consider the long-only, fully-invested minimum variance portfolio optimization, described in Program (7.37).

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \; & \frac{1}{2} \quad \mathbf{z}^T \, \mathbf{Q}^{(i)} \, \mathbf{z} \\ \text{subject to} \quad & \mathbf{1}^T \, \mathbf{z} = 1, \quad 0 \leq \mathbf{z} \leq 1. \end{aligned} \tag{7.37}$$

As before, the learning process is posed as a bi-level optimization program where the objective is to learn a parameter $\boldsymbol{\theta}$ and the associated covariance matrix, $\hat{\mathbf{Q}}(\boldsymbol{\theta})^{(i)}$, in order to minimize the average realized variance induced by the optimal decision policies $\{\mathbf{z}^*(\boldsymbol{\theta})^{(i)}\}_{i=1}^m$.

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \mathbf{z}^*(\boldsymbol{\theta})^{T(i)} \, \mathbf{Q}^{(i)} \, \mathbf{z}^*(\boldsymbol{\theta})^{(i)} \\ \text{subject to} \quad & \mathbf{z}^*(\boldsymbol{\theta}) = \underset{\mathbf{z}}{\text{argmin}} \, \frac{1}{2} \, \mathbf{z}^T \, \hat{\mathbf{Q}}(\boldsymbol{\theta})^{(i)} \, \mathbf{z} \quad \forall i = 1, ..., m \\ & \mathbf{1}^T \, \mathbf{z}^*(\boldsymbol{\theta})^{(i)} = 1, \quad 0 \leq \mathbf{z}^*(\boldsymbol{\theta})^{(i)} \leq 1 \quad \forall i = 1, ..., m \end{aligned} \tag{7.38}$$

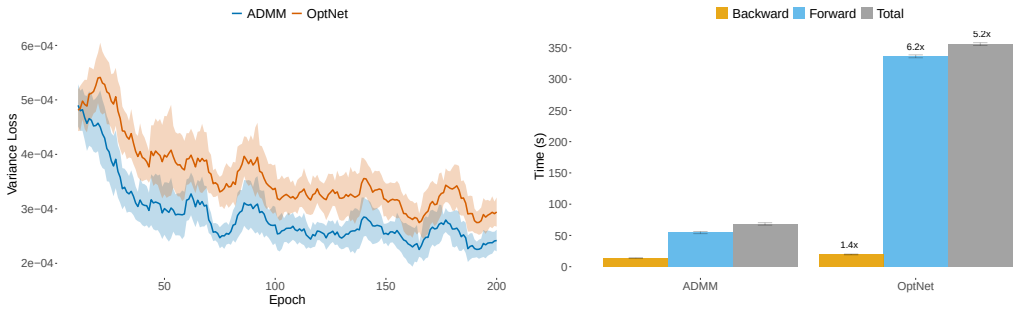
Asset returns, $\mathbf{a}^{(i)}$ are modelled using the Famma-French Five (FF5) factor model. We follow the covariance model described in Chapter 6 and model $\mathbf{w}^{(i)}$ according to a multivariate GARCH$(1, 1)$ process with constant correlation. We let $\hat{\mathbf{W}}^{(i)}$ denote the the time-varying covariance estimate of the feature feature variables. We therefore model the stock covariance matrix as follows:

$$\hat{\mathbf{a}}^{(i)} = \boldsymbol{\theta}^T \, \mathbf{w}^{(i)}, \quad \hat{\mathbf{Q}}^{(i)} = \boldsymbol{\theta}^T \, \hat{\mathbf{W}}^{(i)} \, \boldsymbol{\theta} + \hat{\mathbf{F}}, \tag{7.39}$$

where $\hat{\mathbf{F}}$ denotes the diagonal matrix of residual variances.

Again, the goal is to observe the training and out-of-sample performance of the ADMM model in comparison to the OptNet model. As a benchmark, we include the out-of-sample performance of the equal weight portfolio, and a minimum variance portfolio where $\boldsymbol{\theta}$ is fit by OLS. The training process for each trial consists of 200 epochs with a mini-batch size of 32. Portfolio models are fit to an accuracy of $\epsilon_{\text{tol}} = 10^{-4}$ in training, and a higher accuracy of $\epsilon_{\text{tol}} = 10^{-6}$ in the out-of-sample period.

Figure 7.7(a) reports the average training loss at each epoch and the 95%-ile confidence interval, evaluated over 10 independent trials. We observe that the loss curves for the ADMM model and OptNet model are very similar, thus highlighting the accuracy of the ADMM layer. Once again we observe that the ADMM model produces a consistently lower average training loss and we refer to the discussion in Experiment 3 for a likely explanation. Figure 7.7(b) compares the average and 95%-ile confidence interval of the total time spent executing the forward and backward pass algorithms in training. As before, we observe that the ADMM model requires approximately 60 seconds to train and is approximately 5 times faster than the OptNet model, which requires over 300 seconds.



(a) Training Loss

(b) Computational Performance

Figure 7.7: Training loss and computational performance for learning $\mathbf{Q}$ on US stock data. Batch size $= 32$ and $d_z = 254$.

Finally, Figure 7.8 reports the out-of-sample equity growth of the ADMM IPO minimum variance portfolio, Equal Weight portfolio, OLS minimum variance portfolio and OptNet IPO minimum variance portfolio. The out-of-sample economic performance metrics are reported in Table 7.2. Again, observe that all minimum-variance models outperform the Equal Weight benchmark on an absolute and risk-adjusted basis. Furthermore, the ADMM and OptNet IPO minimum variance models achieve an out-of-sample volatility that is approximately 25% lower and Sharpe ratio that is approximately 35% higher than that of the naive 'predict, then optimize' OLS minimum variance model. These results are broadly consistent with the findings in Chapter 6, which considers an

identical stock universe but with considerably smaller portfolios ($d_z \leq 100$). Our ADMM model, on the other hand, is able to overcome the computational challenges in the medium to large scale limit (described in Butler and Kwon [27]), without any apparent loss in performance accuracy.

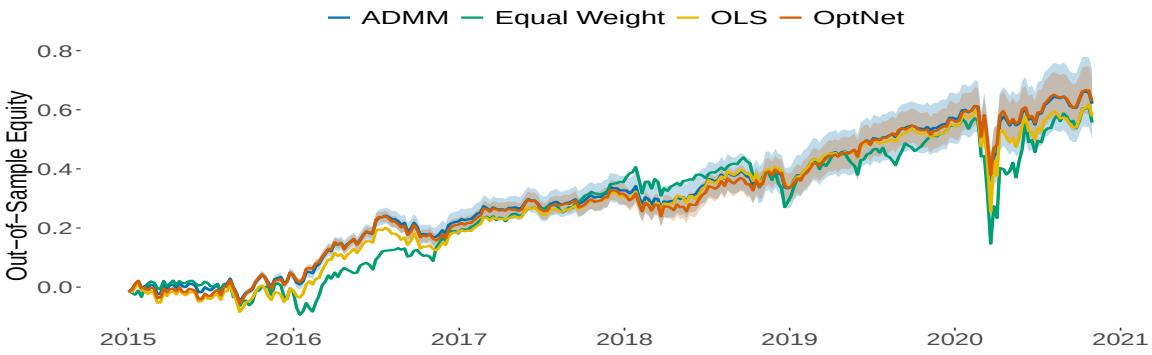


Figure 7.8: Out-of-sample equity growth for ADMM IPO minimum variance portfolio, Equal Weight portfolio, OLS minimum variance portfolio and OptNet IPO minimum variance portfolio.

	ADMM	Equal Weight	OLS	OptNet
Mean	0.1058	0.0950	0.0984	0.1070
Volatility	0.1420	0.1950	0.1786	0.1443
Sharpe Ratio	0.7446	0.4872	0.5510	0.7418

Table 7.2: Out-of-sample economic performance metrics for ADMM IPO minimum variance portfolio, Equal Weight portfolio, OLS minimum variance portfolio and OptNet IPO minimum variance portfolio.

7.4 Conclusion and future work

In this chapter, we provide a novel and efficient framework for differentiable constrained quadratic programming. Our differentiable quadratic programming layer is built on top of the ADMM algorithm, which for medium to large scale problems is shown to be approximately an order of magnitude more efficient than the interior point implementation of the OptNet layer. The backward-pass algorithm computes the relevant problem variable gradients by implicit differentiation of a modified fixed-point iteration, which is computationally favourable to KKT implicit differentiation and memory efficient in comparison to standard unrolled differentiation. Numerical results, using both simulated and real problem data, demonstrates the efficacy of the ADMM layer, which for medium to large scale problems exhibits state-of-the art accuracy and improved computational performance.

Our experimental results should be interpreted as a proof-of-concept and we acknowledge that

further testing on alternative data sets or with different problem variable assumptions is required in order to better determine the efficacy of the ADMM layer as a general purpose solver. Indeed, there is a plethora of areas for active research and algorithm improvement. First, our ADMM layer currently only supports linear equality and box inequality constraints, whereas the OptNet layer supports general linear inequality constraints. Indeed, incorporating more general inequality constraints as well as augmenting the QP with parameterized regularization norms is an active area of research. Secondly, as discussed earlier, the ADMM algorithm is known to be vulnerable to ill-conditioned problems and the resulting convergence rates can vary significantly when the data and algorithm parameter, ρ, are poorly scaled. To overcome this, most first-order solvers implement a preconditioning and scaling initialization step. Currently, our ADMM layer implementation does not support preconditioning and scaling, which is challenged by virtue of the fact that the problem data is expected to change at each epoch of training. Instead, we leave it to the user to select ρ and manually scale the problem data and acknowledge that preconditioning, scaling and automatic parameter selection is an important area of future development.

Furthermore, there are several heuristic methods that could be implemented in order to improve the convergence rates and computational efficiency of our ADMM forward-pass. For example, acceleration methods, such as Andersen acceleration [5], have recently been shown to improve the convergence rates of first-order solvers [112, 125]. Indeed, applying acceleration methods to the modified fixed-point algorithm presented in this chapter provides a numerically efficient scheme for potentially improving the convergence rate of the ADMM algorithm and is an interesting area of future research. Alternatively, methods that hybridize the efficiency of first-order methods with the precision of interior-point methods, such solution polishing and refinement [25], is another area of future exploration. Nonetheless, our proposed ADMM layer is shown to be highly effective and in its current form can be instrumental for efficiently solving real-world medium and large scale learning problems.

Chapter 8

Gradient boosting for convex cone predict and optimize problems

8.1 Introduction

Chapters 5, 6 and 7 focused explicitly on the integration of prediction models with downstream portfolio optimization using the neural network architecture described in Chapter 4. However, the IPO frameworks developed and studied therein are readily generalizable to other forms of prediction and decision optimization modelling. For example, Bertsimas and Kallus [13] present a general framework for conditional decision making whereby the conditional density is estimated through a variety of machine learning methods, including decision trees, locally weighted regression and nonparametric kernel estimation. Similarly, Grigas et al. [67] present an integrated conditional estimation-optimization framework for estimating prediction model parameter distributions in the context of the downstream optimization problem. Moreover, Elmachtoub et al. [48] provide a Smart 'Predict, then Optimize' (SPO) framework for optimizing decision tree prediction models to minimize decision regret.

In this chapter we present a general purpose 'decision boosting' framework, *dboost*, that combines the strength of gradient boosting ensembles with the IPO framework. Previous work [83] considers gradient boosting for integrated prediction and optimization problems but only considers a small subset of optimization problems with linear inequality constraints. In contrast, the *dboost* framework is capable of supporting any optimization problem that can be cast as a convex quadratic cone program (QCP); and thus supports linear, quadratic and second-order cone programming with

general convex cone constraints. As a secondary technical contribution, we present a novel fixed-point implicit differentiation algorithm for efficiently computing the gradient of the decision loss with respect to all of the cone program variables. Thirdly, the *dboost* framework is provided as an open-source R package, available here:

https://github.com/butl3ra/dboost

The remainder of this chapter is outlined as follows. We begin with a brief overview of related work on integrated prediction and optimization. In Section 8.2 we present convex quadratic cone programming in the context of the IPO framework and provide the fixed-point implicit differentiation algorithm. We then present the *dboost* framework as a general extension of the gradient boosting algorithm proposed by Friedman [61]. Experimental results are provided in Section 8.3 and demonstrate that training prediction models with *dboost* can reduce decision regret by anywhere from $15\% - 90\%$ in comparison to existing solutions. We conclude with a summary of results and a discussion of future work.

8.1.1 Literature Review

The *dboost* framework applies gradient boosting, as proposed by Friedman [61] to a quadratic reformulation of the SPO loss function, proposed by Elmachtoub and Grigas [47], and described in more detail in Section 8.2. Recall that optimizing a decision regret loss by gradient descent is challenging as it requires computing the gradient of the optimal solution with respect to predicted costs. To date, several local gradient based methods [94, 117] and convex approximations [28, 47] have been proposed and have proven to be effective in comparison to traditional 'predict, then optimize'.

Most relevant to our work is the work of Elmachtoub et al. [48] who present SPO trees (SPOT) for optimizing classification and regression trees (CART) [23] to minimize downstream decision regret. The authors also consider a random forest [22] implementation, but do not consider a gradient boosting ensemble approach. Konishi and Fukunaga [83] consider gradient boosting problems under the SPO framework for a subset of optimization problems; namely with linear inequality constraints, but do not consider more general convex optimization problems. To our knowledge, *dboost* is the first 'smart' gradient boosting implementation that can support any convex optimization program that can be cast as convex quadratic cone program. A necessary requirement of our framework is the ability to compute gradients of optimal decisions with respect to predicted costs. In the following section we present the *dboost* framework and provide a novel fixed-point implicit differentiation

algorithm for computing said gradients. We first present an example to motivate a prediction estimation process optimized for downstream decision regret.

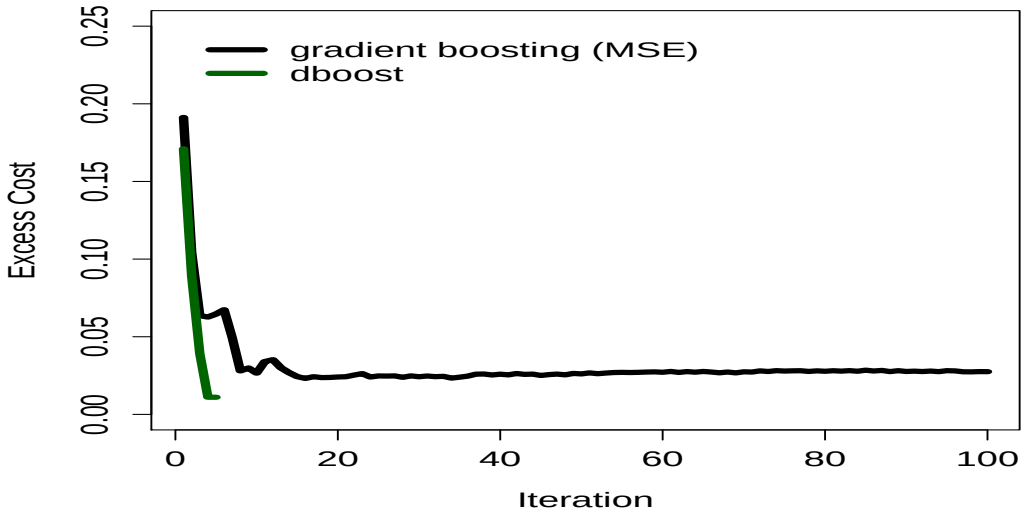
8.1.2 Motivating example

We are motivated by the work of [47, 48] who demonstrate that optimizing prediction model parameters to minimize decision regret produces prediction models with lower complexity and improved out-of-sample performance. Below, we provide a motivating example to help illustrate the behaviour of *dboost* in comparison to a traditional gradient boosting model trained to minimize prediction mean-square error (MSE). We seek to maximize the return (minimize the cost) of a portfolio of two assets subject to linear constraints:

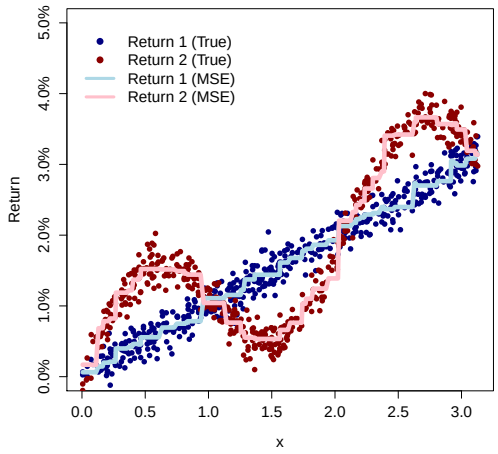
$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & -\mathbf{z}_1\,\mathbf{r}_1 - \mathbf{z}_2\,\mathbf{r}_2 + \frac{1}{2}\|\mathbf{z}\|_2^2 \\ \text{subject to} \quad & \mathbf{z}_1 + \mathbf{z}_2 = 1 \\ & \mathbf{z} \geq 0 \end{aligned} \tag{8.1}$$

where $\mathbf{z}_1$ and $\mathbf{z}_2$ denote the proportion of capital allocated to asset 1 and 2, respectively. Here, the norm penalty $\|\mathbf{z}\|_2^2$ enforces portfolio diversification when differences in returns are small (see for example [29, 41]). We generate a dataset of 500 observations where the feature, $\mathbf{x}$, is drawn from the standard uniform distribution and the return of each asset is given by $\mathbf{r}_1 = \mathbf{x} + \epsilon$ and $\mathbf{r}_2 = \mathbf{x} + \sin(3\,\mathbf{x}) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.10)$.

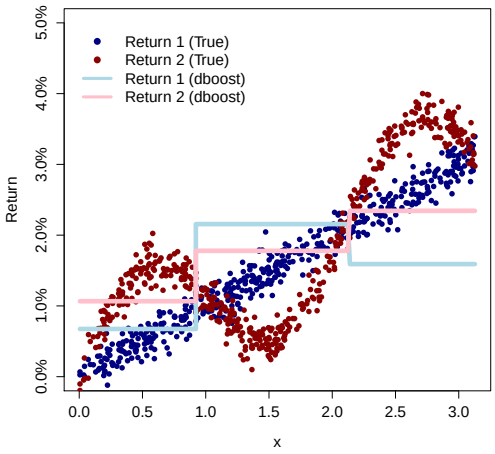
Figure 8.1a plots the total excess decision cost (defined in Equation (8.18)) at each iteration of gradient boosting. Recall, that the traditional model is unaware as to how the predictions will ultimately be used in the context of the final downstream optimization model, and as such continues to add trees to the ensemble in an effort to minimize the prediction MSE. In contrast, the *dboost* model explicitly minimizes decision regret; the boosting algorithm terminates after 5 iterations with *equal* decision accuracy. Figure 8.1b and 8.1c plot the predictions model forecasts of the MSE and *dboost* models as a function of the feature variable $\mathbf{x}$. In this example there are two decision boundaries: $\mathbf{x} \approx 1$ and $\mathbf{x} \approx 2$, where the optimal decision changes from favouring one asset over the other. The *dboost* prediction model correctly identifies these approximate decision boundaries with substantially fewer trees and therefore avoids having to overfit the training dataset.



(a) Excess decision cost at each iteration of gradient boosting.



(b) MSE predictions.

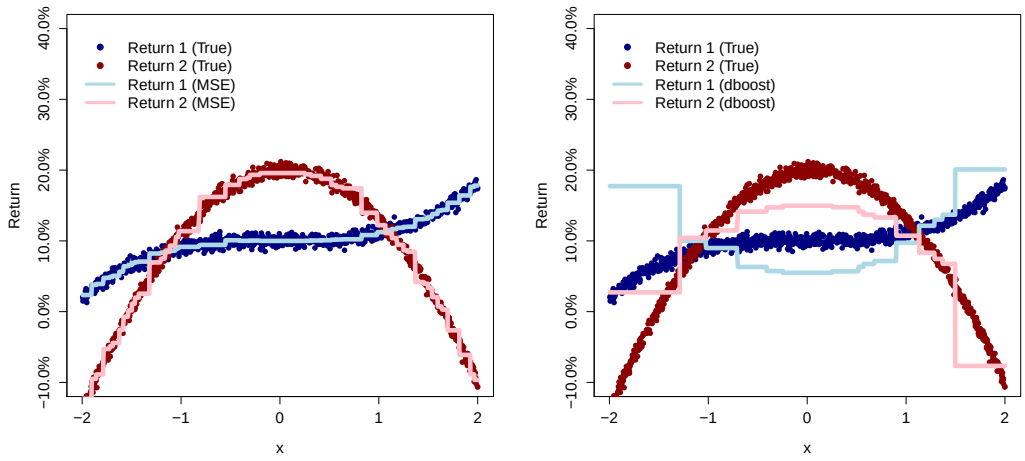


(c) *dboost* predictions.

Figure 8.1: Convergence plot and return forecasts for the MSE gradient boosting and *dboost* prediction models.

We now consider an identical optimization problem but with an alternative dataset of 1000 observations where the feature, $\mathbf{x}$, is drawn from the uniform distribution $\mathcal{U}(-2, 2)$ and the return of each asset is given by $\mathbf{r}_1 = \mathbf{x}^3 + 10 + \epsilon$ and $\mathbf{r}_2 = \mathbf{x} - 8\,\mathbf{x}^2 + 20 + \epsilon$, and $\epsilon \sim \mathcal{N}(0, 0.5)$. As demonstrated in Chapter 3, we observe that the IPO predictions can be counterintuitive when compared with the true response values. Figure 8.2a and 8.2b plot the predictions model forecasts of the MSE and *dboost* models, respectively. As expected, the MSE predicted returns are optimized for prediction accuracy and therefore their predictions closely resemble the true response values. In contrast, the

dboost predictions, reported in Figure 8.2b, are at times vastly different than the true return values. In fact, when $\mathbf{x} = -2$ then *dboost* reports a return prediction of $\mathbf{r}_1 \approx 18\%$, despite the fact that the true return value is $\mathbf{r}_1 \approx 2\%$. Nonetheless, the *dboost* model correctly identifies approximate decision boundaries, resulting in near zero decision error.



(a) MSE predictions.

(b) *dboost* predictions.

Figure 8.2: Return forecasts for the MSE gradient boosting and *dboost* prediction models.

8.2 Methodology

We consider convex quadratic cone programs (QCP) with the following primal-dual form [102]:

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}\mathbf{z}^T\mathbf{P}\mathbf{z} + \mathbf{c}^T\mathbf{z} & \\ \text{subject to} & \mathbf{A}\mathbf{z} + \mathbf{s} = \mathbf{b} & \\ & \mathbf{s} \in \mathcal{K} & \end{array} \qquad \begin{array}{lll} \text{maximize} & -\frac{1}{2}\mathbf{z}^T\mathbf{P}\mathbf{z} - \mathbf{b}^T\mathbf{y} & \\ \text{subject to} & \mathbf{P}\mathbf{z} + \mathbf{A}^T\mathbf{y} + \mathbf{c} = 0 & \\ & \mathbf{y} \in \mathcal{K}^* & \end{array} \tag{8.2}$$

where $\mathbf{z} \in \mathbb{R}^{d_z}$, $\mathbf{y} \in \mathbb{R}^{d_y}$ and $\mathbf{s} \in \mathbb{R}^{d_y}$ denote the primal, dual and slack variables, respectively. The objective input variables are a symmetric positive semi-definite matrix $\mathbf{P} \in \mathbb{R}^{d_z \times d_z}$ and cost vector $\mathbf{c} \in \mathbb{R}^{d_z}$. The feasible region is defined by the matrix $\mathbf{A} \in \mathbb{R}^{d_y \times d_z}$, the vector $\mathbf{b} \in \mathbb{R}^{d_y}$ and the nonempty convex cone $\mathcal{K}$ with associated dual cone $\mathcal{K}^*$. We denote the optimal solution to Program (8.2) as $\boldsymbol{\zeta}^* = (\mathbf{z}^*, \mathbf{y}^*, \mathbf{s}^*) \in \mathbb{S} \subset \mathbb{R}^{d_z} \times \mathcal{K} \times \mathcal{K}^*$ with feasible region:

$$\mathbb{S} = \{\boldsymbol{\zeta} \in \mathbb{R}^{d_z} \times \mathcal{K} \times \mathcal{K}^* \mid \mathbf{A}\,\mathbf{z} + \mathbf{s} = \mathbf{b}, \mathbf{P}\,\mathbf{z} + \mathbf{A}^T\,\mathbf{y} + \mathbf{c} = 0\}. \tag{8.3}$$

In practice, the true cost $\mathbf{c}$ is not directly observable at decision time, and rather an associated feature vector $\mathbf{x} \in \mathbb{R}^{d_x}$ is observed. Given a training data sets $D = \{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^m$ we seek to estimate a prediction model $f\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z}$ such that:

$$\hat{\mathbf{c}}^{(i)} = f(\mathbf{x}^{(i)}, \boldsymbol{\theta}^*) \tag{8.4}$$

with optimal prediction model parameter $\boldsymbol{\theta}^*$ given by:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_D[\ell(\hat{\mathbf{c}}^{(i)}, \mathbf{c}^{(i)})], \tag{8.5}$$

In this chapter we consider 'additive' prediction models of the form:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{n=0}^{N} \beta_n h(\mathbf{x}, \boldsymbol{\alpha}_n), \tag{8.6}$$

with parameter $\boldsymbol{\theta} = \{(\beta_n, \boldsymbol{\alpha}_n)\}_{n=1}^N$ and non-negative weight $\beta_n \geq 0 \quad \forall n$. In particular, we focus on the case where each of the functions $h(\mathbf{x}, \boldsymbol{\alpha}_m)$ is a regression tree and therefore the parameters $\boldsymbol{\alpha}_n$ encode the feature component and splitting information.

Recall that a traditional 'predict, then optimize' approach would first estimate $\boldsymbol{\theta}^*$ by minimizing a prediction loss (such as least-squares) and then simply 'plug-in' the estimate $\hat{\mathbf{c}}^{(i)}$ to Program (8.2) in order to retrieve the 'optimal' decision. In contrast, an integrated prediction and optimization (IPO) approach estimates the prediction model parameters by minimizing the decision regret. Here, we choose to minimize a quadratic reformulation of the SPO loss function:

$$\ell_{\text{QSPO}}(\hat{\mathbf{c}}, \mathbf{c}) = \frac{1}{2}\,\mathbf{z}^*(\hat{\mathbf{c}})^T\,\mathbf{P}\,\mathbf{z}^*(\hat{\mathbf{c}}) + \mathbf{c}^T\,\mathbf{z}^*(\hat{\mathbf{c}}) - \frac{1}{2}\,\mathbf{z}^*(\mathbf{c})^T\,\mathbf{P}\,\mathbf{z}^*(\mathbf{c}) - \mathbf{c}^T\,\mathbf{z}^*(\mathbf{c}) \tag{8.7}$$

where

$$\mathbf{z}^*(\hat{\mathbf{c}}) = \operatorname*{argmin}_{\mathbf{z} \in \mathbb{S}_z} \frac{1}{2}\,\mathbf{z}^T\,\mathbf{P}\,\mathbf{z} + \hat{\mathbf{c}}^T\,\mathbf{z}\,. \tag{8.8}$$

As before, the IPO estimation process can be posed as a bi-level optimization:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{m}\sum_{i=1}^{m} \ell_{\text{QSPO}}(\hat{\mathbf{c}}^{(i)}, \mathbf{c}^{(i)}) \\ \text{subject to} \quad & \mathbf{z}^*(\hat{\mathbf{c}}^{(i)}) = \underset{\mathbf{z}\in\mathbb{S}_z}{\operatorname{argmin}} \frac{1}{2}\mathbf{z}^T\mathbf{P}\,\mathbf{z} + \hat{\mathbf{c}}^{(i)^T}\mathbf{w}\,. \end{aligned} \tag{8.9}$$

In this chapter, we approximate a local solution to Program (8.9) by applying functional gradient descent [61]. Specifically, at each iteration of gradient descent we must first solve m QCPs to optimality. The gradient, $\nabla_{\hat{\mathbf{c}}^{(i)}}\ell_{\text{QSPO}}$, is then obtained by specialized argmin differentiation described in more detail below.

8.2.1 Fixed-point argmin differentiation

Program (8.2) is solved by applying a Douglas-Rachford splitting to a homogeneous embedding of the QCP as described in O'Donoghue [102]. Specifically, we define the convex cone $\mathcal{C} = \mathbb{R}^{d_z} \times \mathcal{K}^*$ and denote:

$$\mathbf{u} = \begin{bmatrix} \mathbf{z} \\ \mathbf{y} \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} \mathbf{P}+\mathbf{I_z} & \mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} \mathbf{c} \\ \mathbf{b} \end{bmatrix}, \tag{8.10}$$

where $\mathbf{I_z}$ denotes the identity matrix of same dimension as $\mathbf{z}$. O'Donoghue [102] demonstratea that a direct application of operator splitting produces the following procedure; from any initial $\mathbf{u}^0$ and $\mathbf{v}^0$ then the following iterations converge to the optimal $\boldsymbol{\zeta}^*$ (if it exists):

$$\tilde{\mathbf{u}}^{k+1} = (\mathbf{I_u} + \mathbf{M})^{-1}(\mathbf{u}^k + \mathbf{v}^k - \mathbf{q}) \tag{8.11a}$$

$$\mathbf{u}^{k+1} = \Pi_{\mathcal{C}}(\tilde{\mathbf{u}}^{k+1} - \mathbf{v}^k) \tag{8.11b}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k + \mathbf{u}^{k+1} - \tilde{\mathbf{u}}^{k+1}, \tag{8.11c}$$

where $\Pi_{\mathcal{C}}$ denotes the euclidean projection operator onto the set $\mathcal{C}$.

We follow Butler and Kwon [26] and recast the iterative procedure (8.11) as a fixed-point iteration over variable $\mathbf{w}^k = \tilde{\mathbf{u}}^{k+1} - \mathbf{v}^k$. We then apply the *implicit function theorem* [44] to the fixed-point iteration in order to derive an efficient scheme for computing the required gradients. We begin with the following proposition. Note that all proofs are available in the Appendix.

Proposition 13. *Let* $\mathbf{w}^k = \tilde{\mathbf{u}}^{k+1} - \mathbf{v}^k$ *and define* $F\colon \mathbb{R}^{d_z+d_y} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z+d_y}$. *Then the iterations*

in Equation (8.11) *can be cast as a fixed-point iteration of the form* $F(\mathbf{w}, \boldsymbol{\theta}) = \mathbf{w}$ *given by:*

$$\mathbf{w}^{k+1} = (\mathbf{I_w} + \mathbf{M})^{-1}(2\Pi_{\mathcal{C}}(\mathbf{w}^k) - \mathbf{w}^k - \mathbf{q}) + \mathbf{w}^k - \Pi_{\mathcal{C}}(\mathbf{w}^k). \tag{8.12}$$

We follow Busseti et al. [25] and denote the derivative of the projection operator as $D\Pi_{\mathcal{C}}$. Note for ease of notation we drop the index k. The Jacobian, $\nabla_{\mathbf{w}} F$, is therefore defined as:

$$\nabla_{\mathbf{w}} F = (\mathbf{I_w} + \mathbf{M})^{-1}(2D\Pi_{\mathcal{C}}(\mathbf{w}) - \mathbf{I_w}) + \mathbf{I_w} - D\Pi_{\mathcal{C}}(\mathbf{w}). \tag{8.13}$$

Direct application of the implicit function theorem gives the desired Jacboian $\nabla_{\boldsymbol{\theta}} \mathbf{w}^*(\boldsymbol{\theta})$, with respect to the parameter $\boldsymbol{\theta}$ at optimality $\mathbf{w}^*$:

$$\nabla_{\boldsymbol{\theta}} \mathbf{w}^*(\boldsymbol{\theta}) = [\mathbf{I_w} - \nabla_{\mathbf{w}} F(\mathbf{w}^*(\boldsymbol{\theta}), \boldsymbol{\theta})]^{-1} \nabla_{\boldsymbol{\theta}} F(\mathbf{w}^*(\boldsymbol{\theta}), \boldsymbol{\theta}). \tag{8.14}$$

From the definition of $\mathbf{w}^*$ we have that $\mathbf{u}^* = \Pi_{\mathcal{C}}(\mathbf{w}^*)$ and therefore $\nabla_{\mathbf{w}^*} \mathbf{u}^* = D\Pi_{\mathcal{C}}(\mathbf{w}^*)$.

In practice, it is inefficient to form the Jacobian $\nabla_{\boldsymbol{\theta}} \mathbf{u}^*(\boldsymbol{\theta})$ directly and instead we compute the left matrix-vector product of the Jacobian with respect to the relevant gradient, $\partial \ell / \partial \mathbf{z}^*$, as outlined below.

Proposition 14. *Denote the matrix* $\mathbf{G}$ *as:*

$$\mathbf{G} = (\mathbf{I_w} + \mathbf{M}) D\Pi_{\mathcal{C}}(\mathbf{w}^*) + \mathbf{I_w} - 2D\Pi_{\mathcal{C}}(\mathbf{w}^*) \tag{8.15}$$

Let $\hat{\mathbf{d}}_{\mathbf{z}}$ *and* $\hat{\mathbf{d}}_{\mathbf{y}}$ *be defined as:*

$$\begin{bmatrix} \hat{\mathbf{d}}_{\mathbf{z}} \\ \hat{\mathbf{d}}_{\mathbf{y}} \end{bmatrix} = \mathbf{G}^{-T} D\Pi_{\mathcal{C}}(\mathbf{w}^*)^T \begin{bmatrix} (\frac{\partial \ell}{\partial \mathbf{z}^*})^T \\ \mathbf{0} \end{bmatrix} \tag{8.16}$$

Then the gradients of the loss function, ℓ*, with respect to problem variables* $\mathbf{P}$*,* $\mathbf{c}$*,* $\mathbf{A}$ *and* $\mathbf{b}$ *are given by:*

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{P}} &= \frac{1}{2}\left(\hat{\mathbf{d}}_{\mathbf{z}} \mathbf{z}^{*T} + \mathbf{z}^* \hat{\mathbf{d}}_{\mathbf{z}}^T\right) & \frac{\partial \ell}{\partial \mathbf{c}} &= \hat{\mathbf{d}}_{\mathbf{z}} \\ \frac{\partial \ell}{\partial \mathbf{A}} &= \mathbf{y}^* \hat{\mathbf{d}}_{\mathbf{z}}^T - \hat{\mathbf{d}}_{\mathbf{y}} \mathbf{z}^{*T} & \frac{\partial \ell}{\partial \mathbf{b}} &= \hat{\mathbf{d}}_{\mathbf{y}} \end{aligned} \tag{8.17}$$

Proposition 14 provides a framework for computing the gradient of any arbitrary loss function with respect to the cost variable $\mathbf{c}$ by implicit differentiation of the fixed-point mapping of the transformed operator splitting iterations. We now present the *dboost* algorithm which applies gra-

dient boosting to minimize the downstream decision regret. We highlight two important differences between Algorithm 1 and the standard gradient boosting algorithm. First, on line 8 we compute the gradient of the decision regret, ℓ_{QSPO}, with respect to the estimated costs $\{\hat{\mathbf{c}}^{(i)}\}_{i=1}^m$. Computing the gradient requires that we first solve the corresponding optimal decisions, $\{\mathbf{z}^*(\hat{\mathbf{c}}^{(i)})\}_{i=1}^m$, and then perform implicit differentiation as described above. Similarly, the line search on line 10 is with respect to the decision regret, ℓ_{QSPO}, and therefore we must solve $\{\mathbf{z}^*(\hat{\mathbf{c}}^{(i)} + \beta h(\mathbf{x}^{(i)}, \boldsymbol{\alpha}_n))\}_{i=1}^m$ at each candidate value β. These two important differences make Algorithm 1 significantly more computationally demanding than the typical gradient boosting algorithm.

Algorithm 1 gradient boosting for min ℓ_{QSPO} :

1: **procedure** DBOOST
2: $f_0(\mathbf{x}, \boldsymbol{\theta}) = \text{argmin}_{\boldsymbol{\theta}}$ Program(8.9)
3: Set $n = 0$
4: Set $0 < \epsilon_\beta < 1$ and $0 < \epsilon_\ell < 1$
5: **while** run = TRUE **do**
6: $n = n + 1$
7: $\hat{\mathbf{c}}^{(i)} = f_{n-1}(\mathbf{x}^{(i)}, \boldsymbol{\theta}), i = 1, 2, ..., m$
8: $\tilde{\mathbf{c}}^{(i)} = -\frac{\partial \ell_{\text{QSPO}}}{\partial \hat{\mathbf{c}}^{(i)}}, i = 1, 2, ..., m$
9: $\boldsymbol{\alpha}_n = \text{argmin}_{\boldsymbol{\alpha}} \sum_{i=1}^m (\tilde{\mathbf{c}}^{(i)} - h(\mathbf{x}^{(i)}, \boldsymbol{\alpha}))^2$
10: $\beta_n = \text{argmin}_{\beta} \sum_{i=1}^m \ell_{\text{QSPO}}(\hat{\mathbf{c}}^{(i)} + \beta h(\mathbf{x}^{(i)}, \boldsymbol{\alpha}_n), \mathbf{c}^{(i)})$
11: $f_n(\mathbf{x}, \boldsymbol{\theta}) = f_{n-1}(\mathbf{x}^{(i)}, \boldsymbol{\theta}) + \beta_n h(\mathbf{x}^{(i)}, \boldsymbol{\alpha}_n)$
12: $\ell_n = \sum_{i=1}^m \ell_{\text{QSPO}}(f_n(\mathbf{x}, \boldsymbol{\theta})^{(i)}, \mathbf{c}^{(i)})$
13: $\Delta\ell = (\ell_n - \ell_{n-1})/\ell_{n-1}$
14: **if** $\beta_n < \epsilon_\beta$ or $\Delta\ell < \epsilon_\ell$ or n = N **then**
15: run = FALSE
16: **end**
17: **return** $f_n(\mathbf{x}, \boldsymbol{\theta})$

8.3 Experiments

We present several experiments that compare the out-of-sample decision cost of *dboost* to 5 alternative methods:

1. **CART:** traditional 'predict, then optimize' with prediction model given by the classification

and regression tree as proposed by Breiman et al. [23]. Trees are optimized to minimize prediction MSE.

2. **CART Forest:** A random forest [22] implementation of CART.

3. **SPOT:** smart 'predict, then optimize' trees as proposed by Elmachtoub et al. [48]. Trees are optimized to minimize the SPO loss (8.8).

4. **SPOT Forest:** random forest implementation of SPOT.

5. **MSE Boosting:** traditional 'predict, then optimize' with prediction model given by gradient boosting as proposed by Friedman [61].

Experiments are conducted on an Apple Mac Pro computer (2.7 GHz 12-Core Intel Xeon E5,128 GB 1066 MHz DDR3 RAM) running macOS 'Catalina'. All computations are run on an unloaded, single-threaded CPU. The software was written using the R programming language (version 4.0.0). All ensemble based approaches use a maximum of 100 trees. Random forest implementations perform bagging across feature variables and training data observations with a 50% sampling rate. All models consider maximum tree depths: $\{0, 1, 2\}$.

All experiments are evaluated over 10 independent trials. Each trial consists of a randomly generated training dataset $D = \{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^{m}$ with $m = 1000$ observations and an out-of-sample training dataset with $m = 1000$. Performance is evaluated with respect to the total excess decision cost, given by:

$$\frac{\sum_{i=1}^{m} \ell_{\text{QSPO}}(\hat{\mathbf{c}}^{(i)}, \mathbf{c}^{(i)})}{|\sum_{i=1}^{m} \frac{1}{2} \mathbf{z}^*(\mathbf{c}^{(i)})^T \mathbf{P} \mathbf{z}^*(\mathbf{c}^{(i)}) + {\mathbf{c}^{(i)}}^T \mathbf{z}^*(\mathbf{c}^{(i)})|} \tag{8.18}$$

Synthetic data is generated as follows. Feature variables, $\mathbf{x}$, are generated by randomly drawing from the Uniform distribution $\mathcal{U}(-1, 1)$. The cost values, $\mathbf{c}$, are generated according to the polynomial model:

$$\mathbf{c} = \mathbf{H}_0 + \sum_{j=1}^{p} \mathbf{H}_j \mathbf{x}^j + \tau\epsilon, \tag{8.19}$$

with intercept $\mathbf{H}_0$ and regression coefficients $\mathbf{H}_j \in \mathbb{R}^{d_z \times d_x}$. The parameter p controls the polynomial degree. Regression coefficients are sparse with each element of $\mathbf{H}_j$ having a 50% probability of being 0 and 50% probability of being nonzero $\mathcal{U}(-1, 1)$. We let $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and the scalar value τ controls the amount of noise in the data. All experiments consider three noise levels: $\tau \in \{0.0, 0.5, 1.0\}$.We present three optimization problems:

1. Noisy network-flow. We consider a continuous network flow problem over a directed graph with 5 nodes. Network edges are randomly generated with the probability that node i flows to node

j given by $Pr(i \to j) = 0.75^{|i-j-1|}$. The number of decision variables, d_z, is determined by the number of edges. Cost values are generated according to (8.19) with $\mathbf{H}_0 \sim \mathcal{N}(-\mathbf{1}, \mathbf{1})$, $p = 3$ and $d_x = 5$. We augment the standard network-flow linear program with L_2 norm regularization to regulate flow across the network. The optimization problem is given by:

$$\begin{aligned} \text{minimize} \quad & \hat{\mathbf{c}}^T \mathbf{z} + \frac{1}{2}\|\mathbf{z}\|_2^2 \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b}, \mathbf{0} \leq \mathbf{z} \leq \mathbf{1}\,. \end{aligned} \tag{8.20}$$

2. Noisy quadratic program. We consider an equality constrained quadratic program of the form:

$$\begin{aligned} \text{minimize} \quad & \hat{\mathbf{c}}^T \mathbf{z} + \frac{1}{2}\,\mathbf{z}^T \hat{\mathbf{P}}\,\mathbf{z} \\ \text{subject to} \quad & \mathbf{A}\,\mathbf{z} = \mathbf{b} \end{aligned} \tag{8.21}$$

with number of decision variables $d_z = 25$ and randomly generated constraint matrix $\mathbf{A} \in \mathbb{R}^{3 \times d_z}$ where $Pr(A_{jk} = 0) = Pr(A_{jk} = 1) = 0.50$. The vector $\mathbf{b}$ is chosen to guarantee that the problem is non-empty. Cost values are generated according to (8.19) with $\mathbf{H}_0 = \mathbf{0}$, $p = 3$ and $d_x = 5$. The positive definite matrix, $\mathbf{P}$, is subject to estimation error: $\hat{\mathbf{P}} = \mathbf{P} + 0.1\,\Xi$ where $\Xi = \frac{1}{n}\,\mathbf{L}^T\,\mathbf{L}$, $\mathbf{L} \in \mathbb{R}^{10 d_z \times d_z}$ and entries of $\mathbf{L} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. Program (8.22) occurs in many applications to statistics [118, 119], machine-learning [92, 70], signal-processing [80] and finance [14, 99].

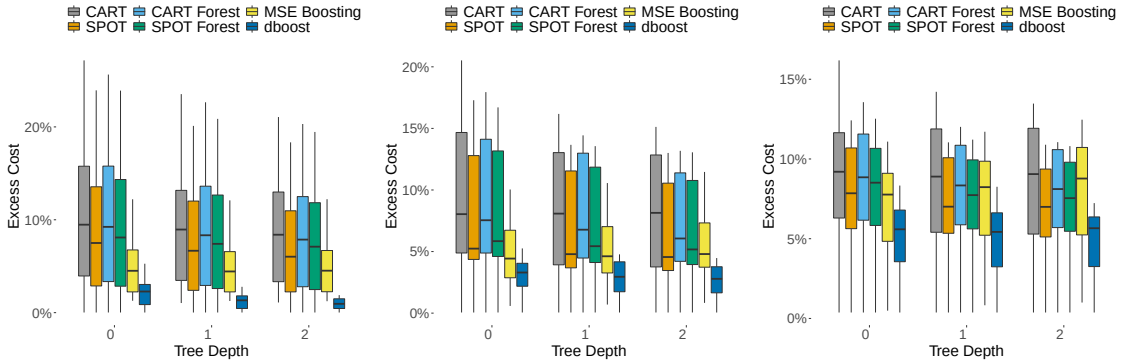
3. Noisy portfolio optimization. We consider the Markowitz [96] mean-variance optimization program:

$$\begin{aligned} \text{minimize} \quad & \hat{\mathbf{c}}^T \mathbf{z} \\ \text{subject to} \quad & \mathbf{1}^T \mathbf{z} = 1, \mathbf{z} \geq 0 \\ & \sqrt{\mathbf{z}^T\,\mathbf{V}\,\mathbf{z}} \leq \sigma, \end{aligned} \tag{8.22}$$

where $\mathbf{z}_j$ denotes the proportion of capital allocated to asset j. The cost values, $\mathbf{c}$, are the negative asset returns and are generated according to (8.19) with $\mathbf{H}_0 = \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, $p = 3$ and $d_x = 5$. The covariance matrix, $\mathbf{V}$, is generated according to a linear factor model $\mathbf{V} = \mathbf{L}^T\,\mathbf{L} + \varepsilon\,\mathbf{I}_\mathbf{z}$, with $\varepsilon = 0.01$ and factor matrix $\mathbf{L} \in \mathbb{R}^{4 \times d_z}$ with entries $\mathbf{L} \sim \mathcal{U}(-1, 1)$. The second-order cone constraint limits the maximum risk of the portfolio. In each trial we set $\sigma = d_z^{-1}\sqrt{\mathbf{1}^T\,\mathbf{V}\,\mathbf{1}}$.

8.3.1 Results

Figure 8.3 reports the out-of-sample excess cost for the noisy network flow problem with noise level $\tau \in \{0.0, 0.5, 1.0\}$. First, observe that for all noise levels, the SPOT models produce a smaller excess cost in comparison to the CART counterparts. In general, the random forest implementations provide a further marginal reduction in excess cost. Gradient boosting models produce excess costs that are on average 50% lower than the corresponding CART and SPOT costs. In all cases, however, the *dboost* model most effectively minimizes the decision cost, further reducing the excess cost of the MSE boosting model by 30%-60%, on average. We observe that the reduction in excess cost is greatest when the noise level is low and in general, deeper tree models tend to produce smaller excess costs.

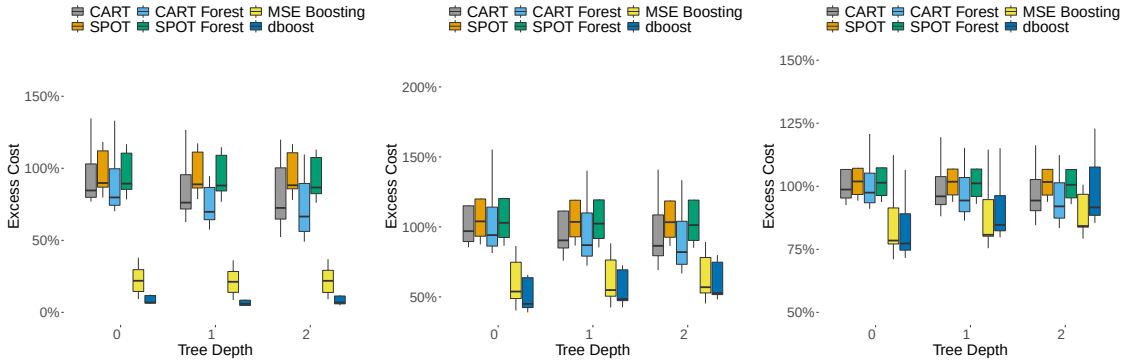


(a) $\tau = 0.0$. (b) $\tau = 0.5$. (c) $\tau = 1.0$.

Figure 8.3: Out-of-sample excess cost for network flow problem with noise level $\tau \in \{0.0, 0.5, 1.0\}$.

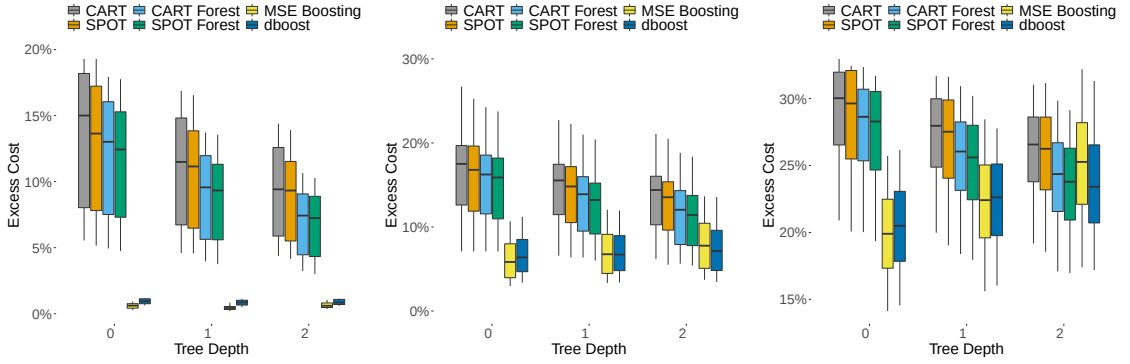
Figure 8.4 reports the out-of-sample excess cost for the noisy quadratic program problem with noise level $\tau \in \{0.0, 0.5, 1.0\}$. Here we note that the CART models produce smaller excess cost in comparison to the SPOT counterparts. Among the CART and SPOT models, the CART Forest produces the smallest excess cost across all noise levels and tree depths. Again, we observe that both gradient boosted models produce the smallest out-of-sample costs, and in most cases, the excess costs are 25%-75% smaller than the corresponding CART and SPOT costs. When $\tau = 0$, the average excess cost of the MSE Boosting model is approximately 22.2% whereas the *dboost* model produces an average excess cost of approximately 5.8%; a further 75% reduction in excess cost over the traditional gradient boosting model. In the majority of cases, the *dboost* model produces the smallest excess cost, however, as the noise level increases the difference in excess costs in comparison to MSE Boosting is less substantial. In fact, when $\tau = 1$ and tree depth is greater than zero, the MSE Boosting model produces marginally lower excess costs. Finally, observe that when the noise

level is high ($\tau = 1$) both gradient boosted models with tree depth 0 produce the smallest excess cost; an argument in favour of low model complexity when the training dataset is noisy.



(a) $\tau = 0$. (b) $\tau = 0.50$. (c) $\tau = 1.0$.

Figure 8.4: Out-of-sample excess cost for quadratic program problem with noise level $\tau \in \{0.0, 0.5, 1.0\}$.



(a) $\tau = 0$. (b) $\tau = 0.50$. (c) $\tau = 1.0$.

Figure 8.5: Out-of-sample excess cost for portfolio optimization problem with noise level $\tau \in \{0.0, 0.5, 1.0\}$.

Lastly, Figure 8.5 reports the out-of-sample excess cost for the noisy portfolio optimization program problem with noise level $\tau \in \{0.0, 0.5, 1.0\}$. As in the network flow problem, we note that the SPOT models produce smaller excess cost in comparison to the CART counterparts. Among the CART and SPOT models, the SPOT Forest produces the smallest excess cost across all noise levels and tree depths. Once again, we observe that both gradient boosted models produce excess costs that, in the most cases, are anywhere from 15%-90% smaller than the corresponding CART and SPOT costs. When $\tau = 0$, we observe that the MSE Boosting model produces almost 0% excess cost; the smallest excess cost across all methods. In contrast to the prior experiments, in this experiment, *dboost* provides no marginal benefit over traditional gradient boosting and we observe

that as the noise level increases, the MSE Boosting and *dboost* produce very similar excess costs. However, across all noise levels, the MSE Boosting with tree depth 0 produces the smallest excess cost; again an argument in favour of prediction models with low model complexity.

8.4 Conclusion and future work

In this chapter, we presented *dboost*, a general purpose framework for building gradient boosting prediction models within the IPO framework. In contrast to prior work, *dboost* is capable of supporting any optimization problem that can be cast as a convex quadratic cone program (QCP). Prediction models are estimated by functional gradient descent, which ultimately requires computing the action of the Jacobian of optimal QCP solutions with respect to the prediction model parameters. We present a novel fixed-point implicit differentiation algorithm for computing the gradient of the decision regret with respect to all of the QCP program variables.

Our experimental results should be interpreted as a proof-of-concept and we acknowledge that further testing on alternative datasets or with different problem variable assumptions is required in order to better determine the efficacy of the integrated gradient boosting approach. In this chapter we considered three experiments: a noisy network flow, a noisy quadratic program and noisy portfolio optimization problem, with a variety of problem specifications. We compared *dboost* with CART and SPOT tree models, as well as a traditional gradient boosting model optimized for prediction MSE. In the majority of cases, both gradient boosting prediction models produce anywhere from a $15\% - 90\%$ reduction in the out-of-sample excess costs in comparison to CART and SPOT models. Moreover, for the network flow and quadratic programming experiment, *dboost* produces the smallest decision costs, with excess costs that are $30\% - 75\%$ smaller than the those produced by the traditional MSE boosting model. In the portfolio optimization experiment, *dboost* provides no marginal benefit over traditional gradient boosting and in fact across all noise levels, the MSE Boosting with tree depth 0 produces the small excess cost. Nonetheless, in this experiment *dboost* remains competitive with MSE Boosting and outperforms both the CART and SPOT models.

We note that training gradient boosting models to minimize downstream decision regret in general is computationally demanding. Indeed, at each iteration of gradient boosting, we must solve m convex quadratic cone programs to obtain optimal decisions: $\{\mathbf{z}^*(\hat{\mathbf{c}}^{(i)})\}_{i=1}^m$, and then compute the resulting gradient of the SPO loss at optimality, both of which can present computational bottlenecks. The gradient, $\nabla_{\hat{\mathbf{c}}^{(i)}} \ell_{\text{QSPO}}$, is computed by implicit argmin differentiation and therefore requires solving m linear systems of dimension $d_z + d_y$, which for large scale problems can be com-

putational expensive. All experiments herein were performed with relatively small training datasets ($m = 1000$) and small decision optimization problems ($d_z \leq 25$). Improving the performance and scalability of *dboost* to support large scale problems is therefore an important area of future research. Further experimentation on alternative optimization problems and datasets to better understand the character of IPO gradient boosting is another area of active research.

Chapter 9

Conclusion

A traditional 'predict, then optimize' approach considers the prediction estimation problem independently from its use in downstream decision-based optimization. In contrast, an integrated prediction and optimization (IPO) approach optimizes prediction model parameters in order to minimize the downstream decision error. This thesis studied the integration of prediction models with downstream portfolio optimization problems and presented several fundamental contributions.

We considered a variety of prediction model and portfolio optimization formulations. Our first contribution examined the integration of linear regression prediction models with the well-known mean-variance optimization (MVO). We identified several special cases where a globally optimal IPO parameterization is possible and derived the relevant theory. In the general case, we formulated a first-order gradient based framework for determining approximate locally optimal IPO regression coefficients. We presented several simulation studies, based on synthetically generated data, and compared the out-of-sample performance of the IPO framework with a traditional 'predict, then optimize' approach based on ordinary least-squares (OLS) regression. Empirical results based on real asset price data demonstrated that the IPO framework can provide lower realized costs and improved out-of-sample economic performance.

We then examined an IPO framework for covariance estimation and evaluated the effectiveness of a covariance estimation process that is optimized for downstream decision objectives. We considered several risk-based portfolio optimization problems and compared the IPO covariance model with a traditional approach based on OLS and maximum-likelihood estimation. In general, our findings are varied; for minimum variance portfolios we demonstrated, across a variety of experiments, that optimizing the covariance model to minimize downstream portfolio variance can result in reduced

out-of-sample variance and improved economic performance. Moreover, we observed that the benefit of the integrated approach is most noticeable when the prediction model is poorly specified; IPO models are shown to be resilient to both factor and correlation model misspecification. However, these results do not hold true universally. Specifically, the IPO framework does not produce materially different economic outcomes for the maximum-diversification and equal-risk-contribution portfolios. We hypothesize that these observed results may speak to the stability of these portfolio optimization processes and acknowledge that further testing on alternative data and with different portfolio objectives and constraints is required.

Thirdly, in contrast to a traditional 'predict, then optimize' approach, training an IPO model by iterative methods can be computationally expensive; in particular as the number of assets in the portfolio is large. Specifically, the frameworks presented herein require solving, at each iteration of gradient descent, hundreds, if not thousands of convex optimization problems. Many real-world portfolio optimization problems involve on the order of $10,000$ training observations and consider portfolios with $10 - 1000$ assets. Therefore, for assets managers that construct portfolios from a very large pool of assets, estimating prediction model parameters by IPO can be computationally impractical. To that end, we addressed the computational challenges in the medium to large scale limit and proposed an alternative differentiable neural network architecture for batch constrained quadratic programs. For medium to large scale problems, our ADMM layer is shown to be approximately an order of magnitude more efficient than the current state-of-the-art. Our software is made available as an open-source R package with the hope that the improved efficiency and accessibility of our framework will help researchers and practitioners, alike, to explore, implement and advance the existing IPO solution space.

Finally, we note that the majority of this thesis focused explicitly on integrating prediction models with downstream portfolio optimization by using a fully integrated neural network approach. That said, the IPO frameworks developed and studied herein are readily generalizable to other forms of prediction and decision optimization modelling. We presented a general purpose 'decision boosting' framework, *dboost*, which optimizes general additive prediction models to minimize downstream decision regret. We considered several optimization problems from a variety of fields and demonstrated that *dboost* is competitive with existing solutions and in many cases can further reduce the out-of-sample decision error. The *dboost* software is made available as an open-source R package with the goal of advancing the accessibility and development of general integrated prediction and optimization solutions.

This thesis explored the fundamental question: 'do optimal predictions result in optimal portfolio

decision-making?'. We presented several examples demonstrating that prediction models that are optimized to maximize prediction accuracy do not necessarily result in the lowest possible decision error. Indeed, the problem depends on several factors, including (1) the prediction model hypothesis class, (2) the decision optimization structure and (3) the training data distribution. Empirical evidence suggests that optimizing prediction models to minimize downstream decision objectives can result in lower decision regret in comparison to the traditional 'predict, then optimize' approach. However, the effect is not universal, and in many instances throughout this thesis, the IPO models provided no additional value, and in some cases produced higher decision regret. Moreover, IPO prediction models are considerably more computationally expensive to train, and can lead to predictions that seem counter intuitive in the context of the training data. Nonetheless, the theoretical foundations and empirical evidence provided in this thesis demonstrate the advantages of a prediction modelling process that is optimized for its use in downstream portfolio optimization decision-making.

Throughout this thesis we acknowledged many possible directions for future research. Below is a consolidation of potential future research directions, aggregated along three broad categories: (1) theory, (2) applications and (3) computational improvements.

1. **Theory:**

    - **Global optimization:** the IPO methods presented in this thesis are local optimization methods and therefore a natural question arises: 'what is the optimality gap between a local solution and a globally optimal solution?'. Recently, Jeong et al. [76] presented an exact mixed-integer linear programming reformulation of the bi-level IPO problem for linear prediction models with a downstream linear decision optimization program. They demonstrated that, under certain conditions, the global solution produces a decision regret that is several orders of magnitude smaller than the corresponding local solutions. In many domains it is important to quantify the quality of a locally optimal IPO solution and in turn, finding a globally optimal solution can potentially lead to improved out-of-sample decision regret. The development of theory and methods for global optimization of more general IPO problems is therefore an important direction for future research.
    - **Inverse optimization:** the IPO framework presented in this thesis is related to *inverse optimization*, which takes as input optimal decisions and determines the optimization program variables that renders these decisions optimal [30]. The IPO approach taken in this thesis is similar to classical inverse optimization, which always assumes feasibility of

the imputed optimal decisions. More recently, data-driven inverse optimization techniques that relax the feasibility condition have proven to be effective in larger-scale settings and is an interesting area of future research. Furthermore, the IPO methods presented in this thesis are supervised and, moreover, both the prediction and optimization models assume a parametric form. As such, the decision optimization program imposes particular views on how complex systems (such as financial markets) are governed. Alternatively, non-parametric inverse optimization methods are generally agnostic in their views and instead objective functions and constraints are determined in a data-driven manner. The development of non-parametric inverse optimization methods for portfolio optimization is therefore an interesting area of future research.

2. **Applications:**

   - **Prediction model regularization:** throughout this thesis we observed that in some instances the IPO model would potentially overfit the training data. There exists a plethora of methods for regularizing prediction models and performing hyper-parameter optimization under the traditional 'predict, then optimize' setting [60, 69, 130]. The development of regularization methods for norm-penalization, feature subset selection, and dropout under the IPO framework is therefore an interesting area of future applied research.
   - **Robust optimization:** this thesis focused on the IPO framework applied to nominal decision-based optimization programs. Integrated prediction and optimization under a robust portfolio optimization setting that minimizes worst-case decision outcomes is another interesting area of future exploration.
   - **Unsupervised prediction models :** the majority of the IPO literature focuses on the integration of supervised prediction models with downstream decision-based optimization. Integrating unsupervised prediction models, such as k-means and hierarchical clustering, hidden Markov models and principle component analysis is an interesting and important direction of applied research. For example, how would cluster definitions change if clusters were optimized to minimize a downstream decision-based optimization objective rather than traditional objectives such as within-cluster sum of squared errors?
   - **Further applications to finance:** this thesis focused on the integration of prediction models with a variety of downstream convex portfolio optimization problems. However,

there exists several other direct applications of the IPO framework to more general problems in quantitative finance, such as:

  - Pairs trading and statistical arbitrage.
  - Fixed-income asset management and duration matching.
  - Project management: cash-flow and internal rate of return estimation for data-driven decision-making.
  - Index tracking-error optimization.
  - IPO for non-convex or heuristic portfolio optimization methods [39, 90, 97].

3. **Computational improvements:**

   - **Surrogate mapping:** the primary computational challenge of our proposed gradient-based IPO framework is that it requires solving, at each iteration of gradient descent, hundreds or thousands of optimization problems. Alternatively, recent work proposes approximating the solution to the computationally expensive lower-level optimization problem with a surrogate model, such as a Gaussian process model [111, 89]. Heuristic methods for solving the bi-level IPO problem based on surrogate mapping of the lower-level optimization, in combination with expected improvement criteria for surrogate refinement [59] are an interesting and relatively unexplored research direction.
   - **ADMM layer improvements:** in Chapter 7 we acknowledged several of the shortcomings of the proposed ADMM layer implementation. For example, the ADMM layer only supports linear equality and box inequality constraints. Generalizing the ADMM layer to support more general constraints as well as incorporating automatic preconditioning, scaling and acceleration methods is important for more widespread adoption and is an active area of development.
   - ***dboost* improvements:** in Chapter 8 we acknowledged that training *dboost* models is considerably more computationally expensive than traditional gradient boosting. Improving the performance and scalability of *dboost* is therefore an important area of research and development. Reducing the number of gradient boosting iterations by incorporating second-order Hessian information [31] or applying fixed-point acceleration methods [124] are interesting areas of future research.

Bibliography

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pages 9562–9574. Curran Associates, Inc., 2019.

[2] Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walaa M. Moursi. Differentiating through a cone program, 2019. URL `https://arxiv.org/abs/1904.09043`.

[3] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks, 2017. URL `https://arxiv.org/abs/1703.00443`.

[4] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control, 2019. URL `http://arxiv.org/abs/1810.13400`.

[5] Donald G. M. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12: 547–560, 1965.

[6] David Ardia, Guido Bolliger, Kris Boudt, and Jean-Philippe Gagnon-Fleury. The impact of covariance misspecification in risk-based portfolios. *Annals of Operations Research*, 254, 03 2017. doi: 10.1007/s10479-017-2474-7.

[7] Xi Bai, Katya Scheinberg, and Reha Tutuncu. Least-squares approach to risk parity in portfolio selection. *Quantitative Finance*, 16(3):357–376, 2016.

[8] Nick Baltas and Robert Kosowski. Momentum strategies in futures markets and trend-following funds. *SSRN Electronic Journal*, 2012. doi: 10.2139/ssrn.1968996.

[9] Gah-Yi Ban and Cynthia Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

[10] Shane Barratt. On the differentiability of the solution to convex optimization problems, 2018. URL https://arxiv.org/abs/1804.05098.

[11] L. Bauwens, S. Laurent, and J. Rombouts. Multivariate garch models: A survey. *Econometrics eJournal*, 2003.

[12] Yoshua Bengio. Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, 08(04):433–443, 1997. doi: 10.1142/S0129065797000422.

[13] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

[14] F. Black and R. Litterman. Asset allocation combining investor views with market equilibrium. *Journal of Fixed Income*, 1(2):7–18, 1991.

[15] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-Lopez, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation, 2021. URL http://arxiv.org/abs/2105.15183.

[16] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307 – 327, 1986. ISSN 0304-4076.

[17] Tim Bollerslev. Modelling the coherence in short-run nominal exchange rates: A multivariate generalized arch model. *The Review of Economics and Statistics*, 72(3):498–505, 1990. ISSN 00346535, 15309142. URL http://www.jstor.org/stable/2109358.

[18] Tim Bollerslev, Robert F. Engle, and Jeffrey M. Wooldridge. A capital asset pricing model with time-varying covariances. *Journal of Political Economy*, 96(1):116–131, 1988.

[19] Tim Bollerslev, Robert Engle, and Daniel B. Nelson. Arch models. In R. F. Engle and D. McFadden, editors, *Handbook of Econometrics*, volume 4, pages 2961–3031. Elsevier, 1 edition, 1994.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.

[21] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 01 2011. doi: 10.1561/2200000016.

[22] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[23] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification And Regression Trees*. Routledge, 10 2017.

[24] Benjamin Bruder, Tung-Lam Dao, Jean-Charles Richard, and Thierry Roncalli. Trend filtering methods for momentum strategies. *SSRN Electronic Journal*, 12 2011. doi: 10.2139/ssrn.2289097.

[25] Enzo Busseti, Walaa M. Moursi, and Stephen Boyd. Solution refinement at regular points of conic problems. *Computational Optimization and Applications*, 74(3):627–643, December 2019. doi: 10.1007/s10589-019-00122-.

[26] Andrew Butler and Roy Kwon. Efficient differentiable quadratic programming layers: an admm approach, 2021. URL `https://arxiv.org/abs/2112.07464`.

[27] Andrew Butler and Roy Kwon. Covariance estimation for risk-based portfolio optimization: an integrated approach. *Journal of Risk*, 24(2), December 2021.

[28] Andrew Butler and Roy H. Kwon. Integrating prediction in mean-variance portfolio optimization, 2021. URL `https://arxiv.org/abs/2102.09287`.

[29] Andrew Butler and Roy H. Kwon. Data-driven integration of regularized mean-variance portfolios, 2021. URL `https://arxiv.org/abs/2112.07016`.

[30] Timothy C. Y. Chan, Rafid Mahmood, and Ian Yihang Zhu. Inverse optimization: Theory and applications, 2021. URL `https://arxiv.org/abs/2109.03920`.

[31] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[32] Wei Chen, Haoyu Zhang, Mukesh Kumar Mehlawat, and Lifen Jia. Mean variance portfolio optimization using machine learning based stock prediction. *Applied Soft Computing*, 100: 106943, 2021. ISSN 1568-4946.

[33] Guillaume Chevalier, Guillaume Coqueret, and Thomas Raffinot. Supervised portfolios. *SSRN Electronic Journal*, 4 2022. URL `https://ssrn.com/abstract=3954109`.

[34] Y. Choueifaty and Y. Coignard. Toward maximum diversification. *The Journal of Portfolio Management*, 35(1):40–51, 2008.

[35] Yves Choueifaty, Tristan Froidure, and Julien Reynier. Properties of the most diversified portfolio. *Journal of Investment Strategies*, 2(2), 2013.

[36] Roger G Clarke, Harindra de Silva, and Robert Murdock. A factor approach to asset allocation. *The Journal of Portfolio Management*, 32(1):10–21, 2005.

[37] Gerard Cornuejols and Reha Tutuncu. *Optimization Methods in Finance*. Cambridge University Press, 2007. doi: 10.1017/CBO9780511753886.

[38] Giorgio Costa and Roy Kwon. A robust framework for risk parity portfolios. *Journal of Asset Management*, 21, 09 2020. doi: 10.1057/s41260-020-00179-w.

[39] Giorgio Costa and Roy Kwon. Generalized risk parity portfolio optimization: an admm approach. *Journal of Global Optimization*, 78, 09 2020. doi: 10.1007/s10898-020-00915-x.

[40] Gianluca De Nard, Olivier Ledoit, and Michael Wolf. Factor Models for Portfolio Selection in Large Dimensions: The Good, the Better and the Ugly. *Journal of Financial Econometrics*, 01 2019. ISSN 1479-8409. doi: 10.1093/jjfinec/nby033. URL `https://doi.org/10.1093/jjfinec/nby033`.

[41] Victor DeMiguel, Lorenzo Garlappi, Raman Uppal, and L Nogales. A generalized approach to portfolio optimization: improving performance by constraining portfolio norms. *Management Science*, 55(5):798–812, 2009.

[42] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled optimization with deep priors, 2017. URL `https://arxiv.org/abs/1705.08041`.

[43] Justin Domke. Generic methods for optimization-based modeling. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 318–326, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL `https://proceedings.mlr.press/v22/domke12.html`.

[44] Asen Dontchev and R Rockafellar. *Implicit Functions and Solution Mappings: A View from Variational Analysis*. Springer New York, 01 2009. ISBN 978-0-387-87820-1. doi: 10.1007/978-0-387-87821-8.

[45] Priya Donti, Brandon Amos, and J. Zico 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*, volume 30, pages 5484 – 5494. Curran Associates, Inc., 2017.

[46] Holger Drees and Catalin Starica. A simple non-stationary model for stock returns, 2002.

[47] Adam Elmachtoub and Paul Grigas. Smart 'predict, then optimize'. *Management Science*, 10 2017. doi: 10.1287/mnsc.2020.3922.

[48] Adam N. Elmachtoub, Jason Cheuk Nam Liang, and Ryan McNellis. Decision trees for decision-making under the predict-then-optimize framework, 2020. URL http://arxiv.org/abs/2003.00360.

[49] Robert Engle. Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, 20(3):339–350, 2002.

[50] Robert Engle and Riccardo Colacito. Testing and valuing dynamic correlations for asset allocation. *Journal of Business & Economic Statistics*, 24(2):238–253, 2006.

[51] Robert F. Engle. Autoregressive conditional heteroskedasticity with estimates of the variance of uk inflation. *Econometrica*, 50(1):987–1008, 1982.

[52] Robert F. Engle, Victor K. Ng, and Michael Rothschild. Asset pricing with a factor-arch covariance structure: Empirical estimates for treasury bills. *Journal of Econometrics*, 45(1): 213 – 237, 1990. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(90)90099-F. URL http://www.sciencedirect.com/science/article/pii/030440769090099F.

[53] Robert F. Engle, Olivier Ledoit, and Michael Wolf. Large dynamic covariance matrices. *Journal of Business & Economic Statistics*, 37(2):363–375, 2019. doi: 10.1080/07350015.2017.1345683.

[54] Frank J. Fabozzi, Petter N. Kolm, Dessislava A. Pachamanova, and Sergio M. Focardi. Robust portfolio optimization. *The Journal of Portfolio Management*, 33(3):40–48, 2007. ISSN 0095-4918.

[55] Eugene F. Fama and Kenneth R. French. The cross section of expected stock returns. *Journal of Financial Finance*, 47(2), 1992.

[56] Eugene F. Fama and Kenneth R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(3):3–56, 1993.

[57] Eugene F. Fama and Kenneth R. French. A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1 – 22, 2015. ISSN 0304-405X.

[58] Jianqing Fan, Yingying Fan, and Jinchi Lv. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, 147(1):186 – 197, 2008. ISSN 0304-4076. doi: https://doi.org/10.1016/j.jeconom.2008.09.017. URL http://www.sciencedirect.com/science/article/pii/S0304407608001346. Econometric modelling in finance and risk management: An overview.

[59] Mark Franey, Pritam Ranjan, and Hugh Chipman. Branch and bound algorithms for maximizing expected improvement functions, 2010. URL https://arxiv.org/abs/1003.0804.

[60] Jerome Friedman, Trevor Hastie, Holger Hofling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2), Dec 2007. ISSN 1932-6157. doi: 10.1214/07-aoas131. URL http://dx.doi.org/10.1214/07-AOAS131.

[61] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[62] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics With Applications*, 2: 17–40, 1976.

[63] Roland Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

[64] D. Goldfarb and G. Iyengar. Robust portfolio selection problems. *Mathematics of Operations Research*, 28(1):1–38, 2003.

[65] D. Goldfarb and Shucheng Liu. An o(n3l) primal interior point algorithm for convex quadratic programming. *Mathematical Programming*, 49:325–340, 1991.

[66] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization, 2016. URL http://arxiv.org/abs/1607.05447.

[67] Paul Grigas, Meng Qi, Zuo-Jun, and Shen. Integrated conditional estimation-optimization, 2021. URL http://arxiv.org/abs/2110.12351.

[68] I. Griva, S.G. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. Society for Industrial and Applied Mathematics, 2009.

[69] Trevor Hastie, Robert Tibshirani, and Ryan J. Tibshirani. Extended comparisons of best subset selection, forward stepwise selection, and the lasso, 2017. URL https://arxiv.org/abs/1707.08692.

[70] Michael Ho, Zheng Sun, and Jack Xin. Weighted elastic net penalized mean-variance portfolio design and computation. *SIAM Journal on Financial Mathematics*, 6(1):1220–1244, 2015.

[71] Corey Hoffstein, Nathan Faber, and Steven Braun. Rebalance timing luck: The (dumb) luck of smart beta. *SSRN*, 2020. URL https://ssrn.com/abstract=3673910.

[72] Der-Ann Hsu, Robert B. Miller, and Dean W. Wichern. On the stable paretian behavior of stock-market prices. *Journal of the American Statistical Association*, 69(345):108–113, 1974.

[73] Dashan Huang, Jiangyuan Li, Liyao Wang, and Guofu Zhou. Time series momentum: Is it there? *Journal of Financial Economics*, 135(3):774–794, 2020. ISSN 0304-405X.

[74] Ronen Israel, Bryan Kelly, and Tobias Moskowitz. Can machines learn finance. *Journal Of Investment Management*, 18, 11 2020.

[75] Ravi Jagannathan and Tongshu Ma. Risk reduction in large portfolios: Why imposing the wrong constraints helps. *The Journal of Finance*, 58(4):1651–1683, 2003.

[76] J Jeong, P Jaggi, A Butler, and S Sanner. An exact symbolic approach for benchmarking predict-then-optimize solvers. *In Proceedings of the 39th International Conference on Machine Learning*, 2022.

[77] A. G. Jones and C. Taylor. Solving inverse problems in computer vision by scale space reconstruction. In *MVA*, 1994.

[78] Nathan Kallus and Xiaojie Mao. Stochastic optimization forests, 2020. URL https://arxiv.org/abs/2008.07473.

[79] Rohit Kannan, Guzin Bayraksan, and James R. Luedtke. Data-driven sample average approximation with covariate information, 2022. URL https://arxiv.org/abs/2207.13554.

[80] Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1:606 – 617, 01 2008. doi: 10.1109/JSTSP.2007.910971.

[81] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `https://arxiv.org/abs/1412.6980`.

[82] Ralph S.J. Koijen, Tobias J. Moskowitz, Lasse Heje Pedersen, and Evert B. Vrugt. Carry. *Journal of Financial Economics*, 127(2):197–225, 2018.

[83] Takuya Konishi and Takuro Fukunaga. End-to-end learning for prediction and optimization with gradient boosting. In *Machine Learning and Knowledge Discovery in Databases*, pages 191–207. Springer International Publishing, 02 2021.

[84] Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. volume 120 of *Proceedings of Machine Learning Research*, pages 761–770. PMLR, 10–11 Jun 2020.

[85] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004.

[86] Olivier Ledoit and Michael Wolf. Nonlinear shrinkage estimation of large-dimensional covariance matrices. *The Annals of Statistics*, 40(2):1024–1060, 2012.

[87] Olivier Ledoit and Michael Wolf. Nonlinear Shrinkage of the Covariance Matrix for Portfolio Selection: Markowitz Meets Goldilocks. *The Review of Financial Studies*, 30(12):4349–4388, 06 2017.

[88] Donald Lien, Yiu Tse, and Albert Tsui. Evaluating the hedging performance of the constant-correlation garch model. *Applied Financial Economics*, 12:791–98, 02 2002. doi: 10.1080/09603100110046045.

[89] Yang Liu, Yu Weng, Rufan Yang, Quoc-Tuan Tran, and Hung D. Nguyen. Gaussian process-based approach for bilevel optimization in the power system – a critical load restoration case, 2020. URL `https://arxiv.org/abs/2012.01388`.

[90] Marcos Lopez de Prado. Building diversified portfolios that outperform out of sample. *The Journal of Portfolio Management*, 42(4):59–69, 2016. ISSN 0095-4918. doi: 10.3905/jpm.2016.42.4.059. URL `https://jpm.pm-research.com/content/42/4/59`.

[91] Yilin Ma, Ruizhu Han, and Weizhong Wang. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, 165: 113973, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113973. URL `https://www.sciencedirect.com/science/article/pii/S0957417420307521`.

[92] Ravi Sastry Ganti Mahapatruni and Alexander Gray. Cake: Convex adaptive kernel density estimation. In Geoffrey Gordon, David Dunson, and Miroslav Dudik, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 498–506, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `https://proceedings.mlr.press/v15/mahapatruni11a.html`.

[93] Sebastien Maillard, Thierry Roncalli, and Jerome Teiletche. The properties of equally weighted risk contribution portfolios. *The Journal of Portfolio Management*, 36(4):60–70, 2010. ISSN 0095-4918. doi: 10.3905/jpm.2010.36.4.060.

[94] Jayanta Mandi and Tias Guns. Interior point solving for lp-based prediction+optimisation, 2020. URL `http://arxiv.org/abs/2010.13943`.

[95] Jaynta Mandi, Emir Demirovic, Peter. J Stuckey, and Tias Guns. Smart predict-and-optimize for hard combinatorial optimization problems, 2019. URL `http://arxiv.org/abs/1911.10092`.

[96] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[97] Suraj S. Meghwani and Manoj Thakur. Multi-objective heuristic algorithms for practical portfolio optimization and rebalancing with transaction cost. *Applied Soft Computing*, 67: 865–894, 2018. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2017.09.025.

[98] Richard Michaud and Robert Michaud. Efficient asset management: A practical guide to stock portfolio optimization and asset allocation. *New York: Oxford University Press*, 1, 2008.

[99] Richard Michaud and Robert Michaud. Estimation error and portfolio optimization: A resampling solution. *Journal of Investment Management*, 6(1):8–28, 2008.

[100] Tobias J. Moskowitz, Yao Hua Ooi, and Lasse Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012.

[101] Daniel Niedermayer, Andrasand Niedermayer. *Applying Markowitz's Critical Line Algorithm*, pages 383–400. Springer US, Boston, MA, 2010.

[102] Brendan O'Donoghue. Operator splitting for a homogeneous embedding of the linear complementarity problem, 2020. URL https://arxiv.org/abs/2004.02177.

[103] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding, 2016.

[104] R.R. Officer. A time series examination of the market factor of the new york stock exchange. *University of Chicago PhD dissertation*, 1971.

[105] Gregory Ongie, Ajil Jalal, Christopher A. Metzler, Richard G. Baraniuk, Alexandros G. Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging, 2020.

[106] Cavit Pakel, Robert F Engle, Kevin K. Shephard, and Neil Shephard. Fitting vast dimensional time-varying covariance models. *Journal of Business and Economic Statistics*, 2019.

[107] P. Ranjan, R. Haynes, and R. Karsten. A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data. *Technomectrics*, 52(4): 366–378, 2011.

[108] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[109] Michel Schubiger, Goran Banjac, and John Lygeros. Gpu acceleration of admm for large-scale quadratic programming. *Journal of Parallel and Distributed Computing*, 144: 55–67, 2020. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2020.05.021. URL https://www.sciencedirect.com/science/article/pii/S0743731520303063.

[110] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on Stochastic Programming*. Society for Industrial and Applied Mathematics, 2009. doi: 10.1137/1.9780898718751.

[111] Ankur Sinha, Tanmay Khandait, and Raja Mohanty. A gradient-based bilevel optimization approach for tuning hyperparameters in machine learning, 2020. URL https://arxiv.org/abs/2007.11022.

[112] Pantelis Sopasakis, Krina Menounou, and Panagiotis Patrinos. Superscs: fast and accurate large-scale conic optimization, 2019. URL https://arxiv.org/abs/1903.06477.

[113] Florin Spinu. An algorithm for computing risk parity weights, 2013. URL https://ssrn.com/abstract=2297383.

[114] Catalin Starica and Clive Granger. Nonstationarities in stock returns. *The Review of Economics and Statistics*, 87(3):503–522, 2005.

[115] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637?672, Feb 2020. ISSN 1867-2957. doi: 10.1007/s12532-020-00179-2. URL http://dx.doi.org/10.1007/s12532-020-00179-2.

[116] Van-Dai Ta, Chuan-Ming Liu, and Direselign Addis. Prediction and portfolio optimization in quantitative trading using machine learning techniques. New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450365390. doi: 10.1145/3287921.3287963. URL https://doi.org/10.1145/3287921.3287963.

[117] Yingcong Tan, Daria Terekhov, and Andrew Delong. Learning linear programs from optimal decisions, 2020. URL http://arxiv.org/abs/2006.08923.

[118] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1), 1996.

[119] A. N. Tikhonov. Solution of incorrectly formulated problemsand the regularization method. *Soviet Mathematics*, pages 1035–1038, 1963.

[120] Y.K Tse. A test for constant correlations in a multivariate garch model. *Journal of Econometrics*, 98(1):107 – 127, 2000. ISSN 0304-4076.

[121] Ayse Sinem Uysal, Xiaoyue Li, and John M. Mulvey. End-to-end risk budgeting portfolio optimization with neural networks, 2021. URL http://arxiv.org/abs/2107.04636.

[122] V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 831–838. Morgan-Kaufmann, 1992.

[123] I. Varga-Haszonits and I. Kondor. Noise sensitivity of portfolio selection in constant conditional correlation garch models. *Physica A: Statistical Mechanics and its Applications*, 385 (1):307 – 318, 2007. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2007.06.017. URL http://www.sciencedirect.com/science/article/pii/S0378437107006292.

[124] Shobha Venkataraman and Brandon Amos. Neural fixed-point acceleration for convex optimization, 2021. URL https://arxiv.org/abs/2107.10254.

[125] Homer Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM J. Numerical Analysis*, 49:1715–1735, 08 2011. doi: 10.2307/23074353.

[126] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1658–1665, July 2019. doi: 10.1609/aaai.v33i01.33011658.

[127] Xingyu Xie, Jianlong Wu, Zhisheng Zhong, Guangcan Liu, and Zhouchen Lin. Differentiable linearized admm, 2019. URL `https://arxiv.org/abs/1905.06179`.

[128] Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. Admm-net: A deep learning approach for compressive sensing mri, 2017. URL `https://arxiv.org/abs/1705.06869`.

[129] Chao Zhang, Zihao Zhang, Mihai Cucuringu, and Stefan Zohren. A universal end-to-end approach to portfolio optimization via deep learning, 2021. URL `https://arxiv.org/abs/2111.09170`.

[130] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[131] Gilles Zumbach. The riskmetrics 2006 methodology. *Econometrics: Applied Econometrics & Modeling eJournal*, 2007.

Appendix A

Integrating prediction in mean-variance optimization

We begin with the following proposition that will become useful later.

Proposition 15. *Let* $\mathbf{V} \in \mathbb{R}^{m \times m}$ *be a symmetric positive definite matrix. Let* $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ *and consider the quadratic form* $\mathbf{A} = \boldsymbol{B}^T \mathbf{V} \boldsymbol{B}$*. Then* $\mathbf{A}$ *is a symmetric positive definite matrix if* $\boldsymbol{B}$ *has full column rank.*

Proof. The symmetry of $\mathbf{A}$ follows directly from the definition. To prove positive definiteness, let $\mathbf{x} \in \mathbb{R}^n$ be a non-zero vector and consider the quadratic form $\mathbf{x}^T \mathbf{A} \mathbf{x}$:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{x}^T \boldsymbol{B}^T \mathbf{V} \boldsymbol{B} \mathbf{x} = \mathbf{y}^T \mathbf{V} \mathbf{y} \,.$$

Clearly $\mathbf{y}^T \mathbf{V} \mathbf{y} > 0$ for all $\mathbf{y} \neq 0$ and $\mathbf{y}^T \mathbf{V} \mathbf{y} = 0 \iff \boldsymbol{B} \mathbf{x} = 0$. But $\boldsymbol{B}$ has full column rank and therefore the only solution to $\boldsymbol{B} \mathbf{x} = 0$ is the trivial solution $\mathbf{x} = 0$. It follows then that $\mathbf{x}^T \boldsymbol{B}^T \mathbf{V} \boldsymbol{B} \mathbf{x} > 0$ and therefore $\mathbf{A}$ is positive definite. □

A.1 Proof of Proposition 2

Let $\mathbb{Z} = \mathbb{R}^{d_z}$, then the solution to the MVO Program (2.15) is given by:

$$\mathbf{z}^*(\hat{\mathbf{y}}^{(i)}) = \frac{1}{\delta} \hat{\mathbf{V}}^{-1^{(i)}} \hat{\mathbf{y}}^{(i)} = \frac{1}{\delta} \hat{\mathbf{V}}^{-1^{(i)}} \mathbf{P} \operatorname{diag}(\mathbf{x}^{(i)}) \boldsymbol{\theta} \,. \tag{A.1}$$

Direct substitution of (A.1) into Equation (5.9) yields the following quadratic objective:

$$\bar{c}(\boldsymbol{\theta}) = \frac{1}{2}\,\boldsymbol{\theta}^T\,\mathbf{H}(\mathbf{x})\,\boldsymbol{\theta} - \boldsymbol{\theta}^T\,\mathbf{d}(\mathbf{x},\mathbf{y}) \tag{A.2}$$

where

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{y}^{(i)}\,\Big) \tag{A.3}$$

and

$$\mathbf{H}(\mathbf{x}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{V}^{(i)}\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\Big). \tag{A.4}$$

Applying Proposition 15 it follows then that if there exists an $\mathbf{x}^{(i)}$ such that $\mathbf{x_j}^{(i)} \neq 0 \quad \forall j \in 1,...,d_x$ then $\mathbf{H}(\mathbf{x},\mathbf{y}) \succ 0$ and therefore (A.5) is a convex quadratic program:

$$\underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}} \quad \frac{1}{2}\,\boldsymbol{\theta}^T\,\mathbf{H}(\mathbf{x})\,\boldsymbol{\theta} - \boldsymbol{\theta}^T\,\mathbf{d}(\mathbf{x},\mathbf{y}). \tag{A.5}$$

In the absence of constraints on $\boldsymbol{\theta}$, then the first-order conditions are necessary and sufficient for optimality, with optimal IPO coefficients given by:

$$\boldsymbol{\theta}^* = \mathbf{H}(\mathbf{x})^{-1}\,\mathbf{d}(\mathbf{x},\mathbf{y}) \tag{A.6}$$

A.2 Proof of Proposition 3

Let $\boldsymbol{\theta}^*$ and $\mathbf{d_u}(\mathbf{x})$ be as defined by Equation (5.15) and Equation (5.17), respectively. It follows then that:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\theta}^*] &= \mathbb{E}\big[\,\mathbf{H}(\mathbf{x})^{-1}\,\mathbf{d}(\mathbf{x},\mathbf{y})\big] \\
&= \mathbf{H}(\mathbf{x})^{-1}\frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbb{E}\big[\,\mathbf{y}^{(i)}\,\big]\Big) \\
&= \mathbf{H}(\mathbf{x})^{-1}\frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}\,\Big) \\
&= \mathbf{H}(\mathbf{x})^{-1}\,\mathbf{d_u}(\mathbf{x})\,\boldsymbol{\theta}
\end{aligned} \tag{A.7}$$

Corollary 3 follows directly from Equation (A.7). Observe that when $\hat{\mathbf{V}}^{(i)} = \mathbf{V}^{(i)}\,\forall i \in \{1,...,m\}$, then

$$\mathbf{H}(\mathbf{x}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\Big).$$

It follows then that:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}^*] &= \mathbb{E}\big[\,\mathbf{H}(\mathbf{x})^{-1}\,\mathbf{d}(\mathbf{x},\mathbf{y})\big] \\ &= \mathbf{H}(\mathbf{x})^{-1}\frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}\,\Big) \\ &= \mathbf{H}(\mathbf{x})^{-1}\,\mathbf{H}(\mathbf{x})\,\boldsymbol{\theta} \\ &= \boldsymbol{\theta}\,.\end{aligned} \tag{A.8}$$

A.3 Proof of Proposition 4

Let $\{\mathbf{y}^{(i)}\}_{i=1}^{m}$ be independent random variables with $\mathbf{y}^{(i)} \sim \mathcal{N}(\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}, \Sigma)$. Let $\hat{\Sigma}$ and $\mathbf{M}$ be as defined by Equation (5.18) and Equation (5.19), respectively. It follows then that:

$$\begin{aligned}\mathrm{Var}(\boldsymbol{\theta}^*) &= \mathrm{Var}\left(\,\mathbf{H}(\mathbf{x})^{-1}\,\mathbf{d}(\mathbf{x},\mathbf{y})\right) \\ &= \mathbf{H}(\mathbf{x})^{-1}\,\mathrm{Var}\left(\frac{1}{m\delta}\sum_{i=1}^{m}\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{y}^{(i)}\right)\mathbf{H}(\mathbf{x})^{-1} \\ &= \mathbf{H}(\mathbf{x})^{-1}\frac{1}{m^2\delta^2}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathrm{Var}(\mathbf{y}^{(i)})\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\Big)\,\mathbf{H}(\mathbf{x})^{-1} \\ &= \mathbf{H}(\mathbf{x})^{-1}\frac{1}{m^2\delta^2}\sum_{i=1}^{m}\Big(\,\mathrm{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\hat{\mathbf{V}}^{-1^{(i)}}\,\hat{\Sigma}\,\hat{\mathbf{V}}^{-1^{(i)}}\,\mathbf{P}\,\mathrm{diag}(\mathbf{x}^{(i)})\Big)\,\mathbf{H}(\mathbf{x})^{-1} \\ &= \mathbf{H}(\mathbf{x})^{-1}\,\mathbf{M}\,\mathbf{H}(\mathbf{x})^{-1}.\end{aligned} \tag{A.9}$$

A.4 Proof of Proposition 5

Let $\mathbf{z}^*(\mathbf{y}^{(i)})$ and $\mathbf{z}^*(\hat{\mathbf{y}}^{(i)})$ be as defined in Equation (5.2) and Equation (5.11), respectively. Recall, the objective function of the minimum tracking-error representation of the IPO program is:

$$L_{\mathrm{te}}(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\|\mathbf{z}^*(\mathbf{x}^{(i)},\boldsymbol{\theta}) - \mathbf{z}^*(\mathbf{y}^{(i)})\|^2_{\mathbf{V}^{(i)}} \tag{A.10}$$

Let $\mathbb{Z} = \mathbb{R}^{d_z}$, then the first-order necessary conditions for optimality of Program (2.15) state:

$$\mathbf{V}^{(i)}\,\mathbf{z}^*(\mathbf{y}^{(i)}) = \mathbf{y}^{(i)} \tag{A.11}$$

Expanding Equation (A.10) and substituting in Equation (A.11) completes the proof:

$$\begin{aligned} L_{\text{te}}(\boldsymbol{\theta}) &= \frac{1}{2m} \sum_{i=1}^{m} \left(\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}) - \mathbf{z}^*(\mathbf{y}^{(i)}) \right)^T \mathbf{V}^{(i)} \left(\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}) - \mathbf{z}^*(\mathbf{y}^{(i)}) \right) \\ &= \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})^T \mathbf{V}^{(i)} \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}) - \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})^T \mathbf{V}^{(i)} \mathbf{z}^*(\mathbf{y}^{(i)}) + \frac{1}{2} \mathbf{z}^*(\mathbf{y}^{(i)})^T \mathbf{V}^{(i)} \mathbf{z}^*(\mathbf{y}^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})^T \mathbf{V}^{(i)} \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}) - \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})^T \mathbf{y}^{(i)} + \mathbf{z}^*(\mathbf{y}^{(i)})^T \mathbf{V}^{(i)} \mathbf{z}^*(\mathbf{y}^{(i)}). \end{aligned} \tag{A.12}$$

Note that the proof of Proposition 9 follows a similar argument for the case of equality constrained MVO portfolios.

A.5 Proof of Proposition 6

In the presence of equality constraints then the solution to the MVO Program is given by:

$$\mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}) = \frac{1}{\delta} \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{P} \operatorname{diag}(\mathbf{x}^{(i)}) \boldsymbol{\theta} + (\mathbf{I} - \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \hat{\mathbf{V}}^{(i)}) \mathbf{z}_0, \tag{A.13}$$

where $\mathbf{z}_0$ be a particular element of $\mathbb{Z} = \{\mathbf{A}\,\mathbf{z} = \mathbf{b}\}$ and $\mathbf{F}$ is a basis for the nullspace of $\mathbf{A}$.

Direct substitution of (A.13) into Equation (5.9) yields the following quadratic objective:

$$\bar{c}(\boldsymbol{\theta}) = \frac{\delta}{2} \sum_{i=1}^{m} \mathrm{L}_1{}^{(i)}(\boldsymbol{\theta}) - \sum_{i=1}^{m} \mathrm{L}_2{}^{(i)}(\boldsymbol{\theta}), \tag{A.14}$$

where

$$\begin{aligned} \mathrm{L}_1{}^{(i)}(\boldsymbol{\theta}) &= \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})^T \mathbf{V}^{(i)} \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta}) \\ &= \frac{1}{\delta^2} \boldsymbol{\theta}^T \operatorname{diag}(\mathbf{x}^{(i)}) \mathbf{P}^T \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{V}^{(i)} \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{P} \operatorname{diag}(\mathbf{x}^{(i)}) \boldsymbol{\theta} \\ &\quad + \frac{2}{\delta} \boldsymbol{\theta}^T \operatorname{diag}(\mathbf{x}^{(i)}) \mathbf{P}^T \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{V}^{(i)} (\mathbf{I} - \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \hat{\mathbf{V}}^{(i)}) \mathbf{z}_0 \\ &\quad + \mathbf{z}_0^T (\mathbf{I} - \hat{\mathbf{V}}^{(i)} \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T) \mathbf{V}^{(i)} (\mathbf{I} - \mathbf{F} (\mathbf{F}^T \hat{\mathbf{V}}^{(i)} \mathbf{F})^{-1} \mathbf{F}^T \hat{\mathbf{V}}^{(i)}) \mathbf{z}_0 \end{aligned} \tag{A.15}$$

and

$$\begin{aligned}\mathrm{L_2}^{(i)}(\boldsymbol{\theta}) &= \mathbf{z}^*(\mathbf{x}^{(i)}, \boldsymbol{\theta})^T \, \mathbf{y}^{(i)} \\ &= \frac{1}{\delta} \, \boldsymbol{\theta}^T \, \mathrm{diag}(\mathbf{x}^{(i)}) \, \mathbf{P}^T \, \mathbf{F} (\mathbf{F}^T \, \hat{\mathbf{V}}^{(i)} \, \mathbf{F})^{-1} \, \mathbf{F}^T \, \mathbf{y}^{(i)} + \mathbf{z}_0^T \, (\mathbf{I} - \mathbf{F} (\mathbf{F}^T \, \hat{\mathbf{V}}^{(i)} \, \mathbf{F})^{-1} \, \mathbf{F}^T \, \hat{\mathbf{V}}^{(i)}) \, \mathbf{y}^{(i)}\end{aligned} \tag{A.16}$$

Simplifying Equation (A.14) and removing constant terms yields the following quadratic objective:

$$\bar{c}(\boldsymbol{\theta}) = \frac{1}{2} \, \boldsymbol{\theta}^T \, \mathbf{H}_{\mathrm{eq}}(\mathbf{x}) \, \boldsymbol{\theta} - \boldsymbol{\theta}^T \, \mathbf{d}_{\mathrm{eq}}(\mathbf{x}, \mathbf{y}) \tag{A.17}$$

where

$$\mathbf{d}_{\mathrm{eq}}(\mathbf{x}, \mathbf{y}) = \frac{1}{m\delta} \sum_{i=1}^{m} \Big(\mathrm{diag}(\mathbf{x}^{(i)}) \, \mathbf{P}^T \, \mathbf{F} (\mathbf{F}^T \, \hat{\mathbf{V}}^{(i)} \, \mathbf{F})^{-1} \, \mathbf{F}^T (\mathbf{y}^{(i)} - \mathbf{V}^{(i)} (\mathbf{I} - \mathbf{F} (\mathbf{F}^T \, \hat{\mathbf{V}}^{(i)} \, \mathbf{F})^{-1} \, \mathbf{F}^T \, \hat{\mathbf{V}}^{(i)}) \, \mathbf{z}_0) \Big) \tag{A.18}$$

and

$$\mathbf{H}_{\mathrm{eq}}(\mathbf{x}) = \frac{1}{m\delta} \sum_{i=1}^{m} \Big(\mathrm{diag}(\mathbf{x}^{(i)}) \, \mathbf{P}^T \, \mathbf{F} (\mathbf{F}^T \, \hat{\mathbf{V}}^{(i)} \, \mathbf{F})^{-1} \, \mathbf{F}^T \, \mathbf{V}^{(i)} \, \mathbf{F} (\mathbf{F}^T \, \hat{\mathbf{V}}^{(i)} \, \mathbf{F})^{-1} \, \mathbf{F}^T \, \mathbf{P} \, \mathrm{diag}(\mathbf{x}^{(i)}) \Big). \tag{A.19}$$

Applying Proposition 15 it follows then that if there exists an $\mathbf{x}^{(i)}$ such that $\mathbf{x_j}^{(i)} \neq 0 \quad \forall j \in 1, ..., d_x$ then $\mathbf{H}_{\mathrm{eq}}(\mathbf{x}) \succ 0$ and therefore (A.20) is a convex quadratic program:

$$\underset{\boldsymbol{\theta} \in \Theta}{\text{minimize}} \quad \frac{1}{2} \, \boldsymbol{\theta}^T \, \mathbf{H}_{\mathrm{eq}}(\mathbf{x}) \, \boldsymbol{\theta} - \boldsymbol{\theta}^T \, \mathbf{d}_{\mathrm{eq}}(\mathbf{x}, \mathbf{y}). \tag{A.20}$$

In the absence of constraints on $\boldsymbol{\theta}$, then the first-order conditions are necessary and sufficient for optimality, with optimal IPO coefficients given by:

$$\boldsymbol{\theta}^*_{\mathrm{eq}} = \mathbf{H}_{\mathrm{eq}}(\mathbf{x})^{-1} \, \mathbf{d}_{\mathrm{eq}}(\mathbf{x}, \mathbf{y}). \tag{A.21}$$

A.6 Proof of Proposition 7

Let $\boldsymbol{\theta}^*_{\text{eq}}$ and $\mathbf{d_e}(\mathbf{x})$ be as defined by Equation (5.29) and Equation (5.30), respectively. It follows then that:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\theta}^*_{\text{eq}}] &= \mathbb{E}\big[\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\mathbf{d}_{\text{eq}}(\mathbf{x},\mathbf{y})\big] \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbb{E}\big[\,\mathbf{y}^{(i)}\,\big]\Big) \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}\,\Big) \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\mathbf{d}_{\mathbf{e}}(\mathbf{x})\,\boldsymbol{\theta}
\end{aligned} \tag{A.22}$$

Corollary 4 follows directly from Equation (A.22). Observe that when $\hat{\mathbf{V}}^{(i)} = \mathbf{V}^{(i)}\,\forall i \in \{1,...,m\}$, then:

$$\mathbf{H}_{\text{eq}}(\mathbf{x}) = \frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\Big).$$

It follows then that:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\theta}^*_{\text{eq}}] &= \mathbb{E}\big[\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\mathbf{d}_{\text{eq}}(\mathbf{x},\mathbf{y})\big] \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\frac{1}{m\delta}\sum_{i=1}^{m}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}\,\Big) \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\mathbf{H}_{\text{eq}}(\mathbf{x})\,\boldsymbol{\theta} \\
&= \boldsymbol{\theta}
\end{aligned} \tag{A.23}$$

A.7 Proof of Proposition 8

Let $\{\mathbf{y}^{(i)}\}_{i=1}^m$ be independent random variables with $\mathbf{y}^{(i)} \sim \mathcal{N}(\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\,\boldsymbol{\theta}, \Sigma)$. Let $\hat{\Sigma}$ and $\mathbf{M}_{\text{eq}}$ be as defined by Equation (5.18) and Equation (5.31), respectively. It follows then that:

$$\begin{aligned}
\text{Var}(\boldsymbol{\theta}^*_{\text{eq}}) &= \text{Var}\left(\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\mathbf{d}\,\text{eq}(\mathbf{x},\mathbf{y})\right) \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\text{Var}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T(\mathbf{y}^{(i)} - \mathbf{V}^{(i)}(\mathbf{I} - \mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)})\,\mathbf{z}_0)\Big)\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1} \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\frac{1}{m^2\delta^2}\sum_{i=1}^{m}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\text{Var}(\mathbf{y}^{(i)})\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\Big)\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1} \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\frac{1}{m^2\delta^2}\sum_{i=1}^{m}\Big(\,\text{diag}(\mathbf{x}^{(i)})\,\mathbf{P}^T\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\hat{\Sigma}\,\mathbf{F}(\mathbf{F}^T\,\hat{\mathbf{V}}^{(i)}\,\mathbf{F})^{-1}\,\mathbf{F}^T\,\mathbf{P}\,\text{diag}(\mathbf{x}^{(i)})\Big)\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1} \\
&= \mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}\,\mathbf{M}_{\text{eq}}\,\mathbf{H}_{\text{eq}}(\mathbf{x})^{-1}.
\end{aligned} \tag{A.24}$$

A.8 Experiment details

All experiments were conducted on an Apple Mac Pro computer (2.7 GHz 12-Core Intel Xeon E5,128 GB 1066 MHz DDR3 RAM) running macOS 'Catalina'. The software was written using the R programming language (version 4.0.0) and torch (version 0.2.0).

Asset Class		**Market (Symbol)**	
Energy	WTI crude (CL)	Heating oil (HO)	Gasoil (QS)
	RBOB gasoline (XB)		
Grain	Bean oil (BO)	Corn (C)	KC Wheat (KW)
	Soybean (S)	Soy meal (SM)	Wheat (W)
Livestock	Feeder cattle (FC)	Live cattle (LC)	Lean hogs (LH)
Metal	Gold (GC)	Copper (HG)	Palladium (PA)
	Platinum (PL)	Silver (SI)	
Soft	Cocoa (CC)	Cotton (CT)	Robusta Coffee (DF)
	Coffee (KC)	Canola (RS)	Sugar (SB)

Table A.1: Futures market universe. Symbols follow Bloomberg market symbology. Data is provided by Commodity Systems Inc (CSI).

Appendix B

Integrated covariance estimation for risk-based portfolio optimization

B.1 Implementation details

The factor covariance model is implemented using a single-layer linear neural network with the appropriate input and output layer dimensions. The univariate and multivariate GARCH processes are modelled according to Equations (6.5) and (6.9), respectively, and implemented using 1-dimensional convolutional neural network layer. GARCH coefficients are constrained to $[0, 1]$, which we implement by applying a standard sigmoid transformation: $S(x) = (1 + e^{-x})^{-1}$.

The 'predict, then optimize' model parameters are optimized by standard OLS and maximum-likelihood procedures, as described in Section 6.2. The IPO parameters are optimized by employing the ADAM stochastic gradient descent routine [81]. We find that using 100% of the training observations with a learning rate of 0.10 results in numerically stable and consistent parameter estimation. For completeness, the pseudo-code training of IPO models is outlined in Algorithm 2.

All experiments were conducted on an Apple Mac Pro computer (2.7 GHz 12-Core Intel Xeon E5,128 GB 1066 MHz DDR3 RAM) running macOS 'Catalina'. The software was written using the R programming language (version 4.0.0) and torch (version 0.2.0).

Algorithm 2 Integrated Predict then Optimize Training

Set learning rate, $\alpha \in (0, 1]$.
Set batch size ratio $r \in (0, 1]$
Set maximum number of iterations of gradient descent: $\text{iter}_{\max} > 1$:
Define training set $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{V}^{(i)})\}_{i=1}^{m}$.
procedure INITIALIZE
    Init OLS coefficients: $\boldsymbol{\theta}_1 \in [-3, 3]^{d_x \times d_y}$.
    Init residual matrix: $\mathbf{F} \in [-12, -6]^{d_y}$
    Init GARCH coefficients: $\boldsymbol{\theta}_2 = [\boldsymbol{\omega}, \boldsymbol{\alpha}, \boldsymbol{\beta}] \in [-4, 1]^{d_x} \times [0, 4]^{d_x} \times [-8, -6]^{d_x}$.
    Init DCC-GARCH coefficients $\boldsymbol{\theta}_3 = (a_1, a_2) \in [-4, 1] \times [0, 4]$.
end procedure
for k in $1, 2, ..., \text{iter}_{\max}$ **do**
    procedure FORWARD PASS
        Randomly sample batch $B \subset [1, ..., m]$ of size $r \cdot m$.
        for i in B **do**
            procedure ESTIMATE COVARIANCE
                $\boldsymbol{\theta}_2^* \leftarrow \frac{1}{1+e^{-\boldsymbol{\theta}_2}}$
                $\boldsymbol{\theta}_3^* \leftarrow \frac{1}{1+e^{-\boldsymbol{\theta}_3}}$
                $\hat{\mathbf{V}}^{(i)} = \boldsymbol{\theta}_1^T W(\mathbf{x}^{(i)}, \boldsymbol{\theta}_2^*, \boldsymbol{\theta}_3^*)\, \boldsymbol{\theta}_1 + \hat{\mathbf{F}}$
            procedure OPTIMIZE PORTFOLIO
                $\mathbf{z}^{*(i)} = \text{argmin}_{\mathbf{z} \in \mathbb{Z}}\, c(\mathbf{z}, \hat{\mathbf{V}}^{(i)})$
            procedure EVALUATE REALIZED COST
                $\text{c}^{(i)} = c(\mathbf{z}^{*(i)}, \mathbf{V}^{(i)})$
        end for
        procedure AVERAGE REALIZED COST
            $\bar{c} = \frac{1}{|B|} \sum_{i \in B} \text{c}^{(i)}$
    end procedure
    procedure UPDATE COEFFICIENTS
        $\boldsymbol{g}_{\boldsymbol{\theta}_1} = \frac{1}{|B|} \sum_{i \in B} \left(\frac{\partial c}{\partial \boldsymbol{\theta}_1} \right)_{|(\mathbf{z}^{*(i)}, \boldsymbol{\theta}_1), \mathbf{V}^{(i)})}$
        $\boldsymbol{g}_{\boldsymbol{\theta}_2} = \frac{1}{|B|} \sum_{i \in B} \left(\frac{\partial c}{\partial \boldsymbol{\theta}_2} \right)_{|(\mathbf{z}^{*(i)}, \boldsymbol{\theta}_2), \mathbf{V}^{(i)})}$
        $\boldsymbol{g}_{\boldsymbol{\theta}_3} = \frac{1}{|B|} \sum_{i \in B} \left(\frac{\partial c}{\partial \boldsymbol{\theta}_3} \right)_{|(\mathbf{z}^{*(i)}, \boldsymbol{\theta}_3), \mathbf{V}^{(i)})}$
        $\boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_1 - \alpha \boldsymbol{g}_{\boldsymbol{\theta}_1}$
        $\boldsymbol{\theta}_2 \leftarrow \boldsymbol{\theta}_2 - \alpha \boldsymbol{g}_{\boldsymbol{\theta}_2}$
        $\boldsymbol{\theta}_3 \leftarrow \boldsymbol{\theta}_3 - \alpha \boldsymbol{g}_{\boldsymbol{\theta}_3}$
    end procedure
end for

B.2 Economic performance metrics

Let $\mathbf{r}^{(i)}$ and $\mathbf{p}^{(i)}$ denote the realized portfolio return and equity, respectively, on week i. Let ${\mathrm{r_f}}^{(i)}$ denote the weekly risk-free rate, as measured by the 13-week US treasury bill.

- **Mean (Excess):** mean annualized return in excess of the risk-free rate.

$$\mu_p = \frac{52}{m} \sum_{i=1}^{m} \left(\mathbf{r}^{(i)} - {\mathrm{r_f}}^{(i)} \right)$$

- **Volatility:** annualized return standard-deviation:

$$\sigma_p = \sqrt{\frac{52}{m-1} \sum_{i=1}^{m} \left(\mathbf{r}^{(i)} - \frac{1}{m} \sum_{i=1}^{m} \mathbf{r}^{(i)} \right)^2}$$

- **Sharpe Ratio:** annualized ratio of mean excess return to volatility:

$$\mathrm{SR}_p = \frac{\mu_p}{\sigma_p}$$

- **VaR:** weekly portfolio α%-ile value-at-risk:

$$\mathrm{VaR}_{1-\alpha}(\mathbf{r}) = \inf_{x \in \mathbb{R}} \{ x \mid \Pr(\mathbf{r} \leq x) \geq 1 - \alpha \}$$

- **Drawdown:** weekly portfolio percent loss from previous high-water mark:

$$\mathbf{d}^{(j)} = \mathbf{p}^{(j)} \,/\, \max_{0 \leq i \leq j} \{ \mathbf{p}^{(i)} \} - 1$$

- **Avg DD:** average weekly portfolio drawdown:

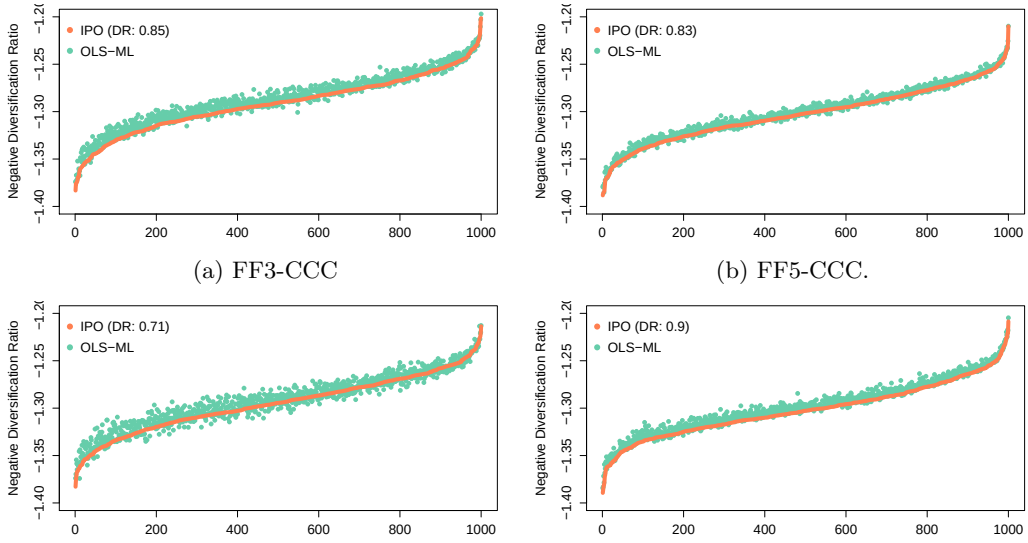
$$\frac{52}{m} \sum_{i=1}^{m} \mathbf{d}^{(i)}$$

- **CDaR:** weekly portfolio α%-ile conditional drawdown at risk:

$$\mathrm{CDaR}_{1-\alpha}(\mathbf{d}) = \inf_{x \in \mathbb{R}} \{ x \mid \Pr(\mathbf{d} \leq x) \geq 1 - \alpha \}$$

B.3 Results: maximum-diversification

We present the results for the U.S. industry data experiments for the maximum-diversification portfolio. Figure B.1 compares the realized portfolio negative diversification-ratios across 1000 out-of-sample realizations. The IPO method provides a consistent marginal reduction in realized negative diversification-ratios. The magnitude of the reduction, however, is not economically meaningful. This conclusion is justified by the economic performance metrics and corresponding equity growth charts, provided in Table B.1 and Figure B.2, respectively, which demonstrate very similar realized performance.



(a) FF3-CCC (b) FF5-CCC.

(c) FF3-DCC (d) FF5-DCC.

Figure B.1: Out-of-sample negative diversification ratio cost of the IPO and OLS-ML methods for the constrained maximum-diversification portfolio.

Statistic	IPO	OLS-ML
Excess Mean	0.0874	0.0829
Volatility	0.1544	0.1481
Sharpe	0.5658	0.5598
VaR	-0.0491	-0.0470
Avg DD	-0.0328	-0.0312
CDaR	-0.2217	-0.2086
Div Ratio	1.2903	1.2854

(a) FF3-CCC

Statistic	IPO	OLS-ML
Excess Mean	0.0879	0.0877
Volatility	0.1534	0.1501
Sharpe	0.5732	0.5844
VaR	-0.0491	-0.0480
Avg DD	-0.0314	-0.0315
CDaR	-0.2204	-0.2153
Div Ratio	1.3019	1.2993

(b) FF5-CCC

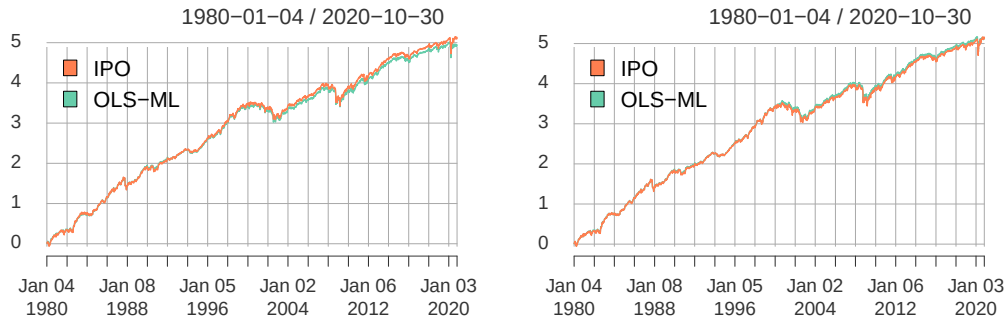
Statistic	IPO	OLS-ML
Excess Mean	0.0872	0.0825
Volatility	0.1544	0.1475
Sharpe	0.5647	0.5592
VaR	-0.0491	-0.0470
Avg DD	-0.0338	-0.0320
CDaR	-0.2330	-0.2083
Div Ratio	1.2925	1.2888

(c) FF3-DCC

Statistic	IPO	OLS-ML
Excess Mean	0.0918	0.0895
Volatility	0.1519	0.1495
Sharpe	0.6045	0.5987
VaR	-0.0484	-0.0477
Avg DD	-0.0313	-0.0314
CDaR	-0.2055	-0.2109
Div Ratio	1.3028	1.2988

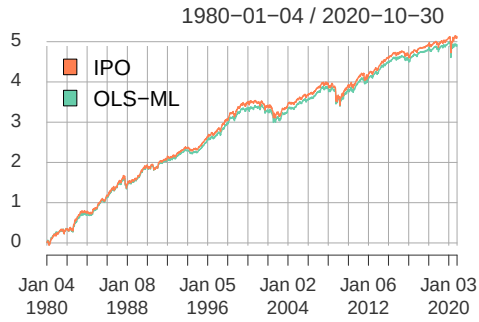
(d) FF5-DCC

Table B.1: Out-of-sample economic and risk metrics of the IPO and OLS-ML methods for the constrained maximum-diversification portfolio.

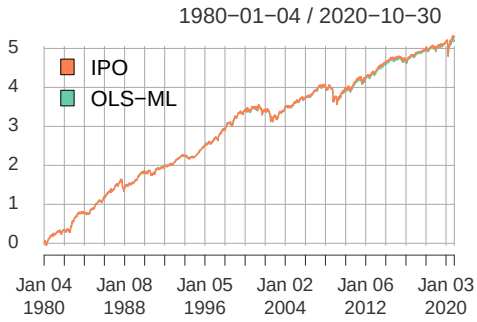


(a) FF3-CCC

(b) FF5-CCC.



(c) FF3-DCC

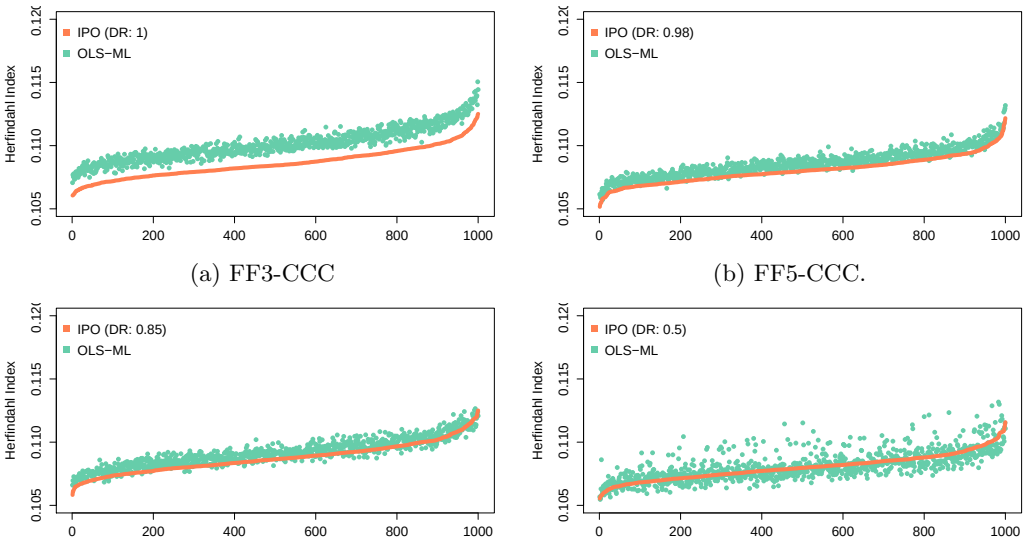


(d) FF5-DCC.

Figure B.2: Out-of-sample equity growth of the IPO and OLS-ML methods for the constrained max-diversification portfolio.

B.4 Results: equal-risk-contribution portfolio

We present the results for the U.S. industry data experiments for the equal-risk-contribution portfolio. Figure B.3 compares the realized portfolio risk contribution Herfindahl index across 1000 out-of-sample realizations. Again, IPO method provides a consistent marginal reduction in realized costs. The magnitude of the reduction, however, is not economically meaningful. This conclusion is justified by the economic performance metrics and corresponding equity growth charts, provided in Table B.2 and Figure B.4, respectively, which demonstrate near equivalent realized performance.



(a) FF3-CCC (b) FF5-CCC.

(c) FF3-DCC (d) FF5-DCC.

Figure B.3: Out-of-sample negative diversification ratio cost of the IPO and OLS-ML methods for the constrained maximum-diversification portfolio.

Statistic	IPO	OLS-ML
Excess Mean	0.0914	0.0913
Volatility	0.1553	0.1555
Sharpe	0.5884	0.5865
VaR	-0.0496	-0.0496
Avg DD	-0.0338	-0.0337
CDaR	-0.2296	-0.2306
Herf Indx	0.1085	0.1100

(a) FF3-CCC

Statistic	IPO	OLS-ML
Excess Mean	0.0908	0.0909
Volatility	0.1545	0.1551
Sharpe	0.5876	0.5857
VaR	-0.0494	-0.0495
Avg DD	-0.0326	-0.0332
CDaR	-0.2286	-0.2292
Herf Indx	0.1080	0.1086

(b) FF5-CCC

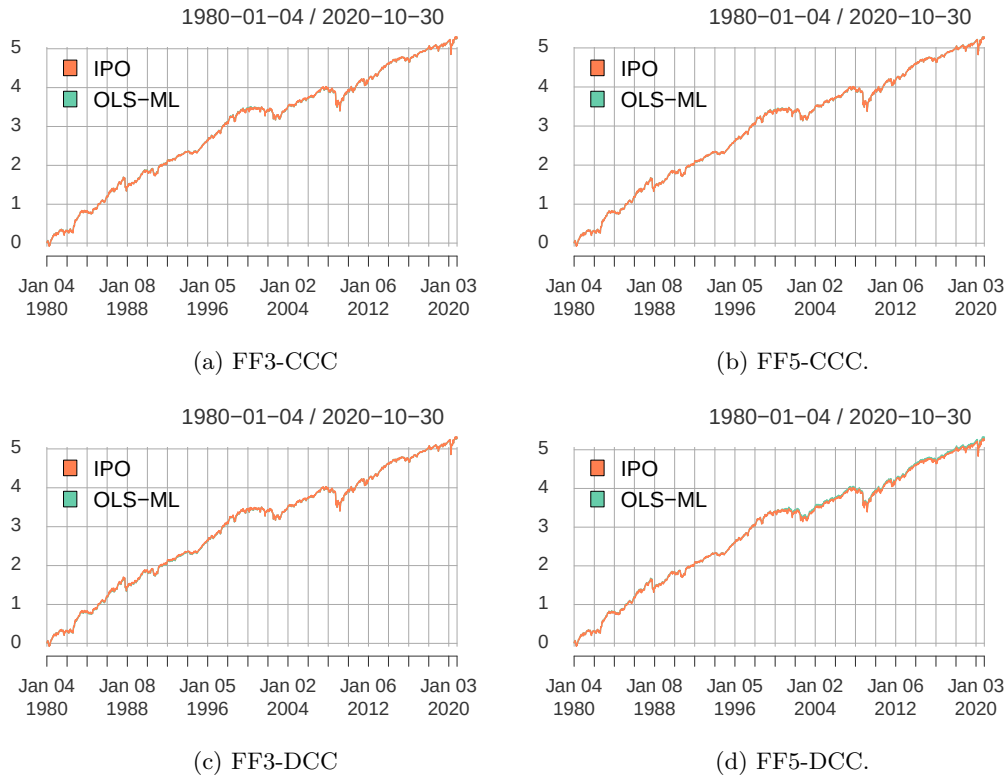
Statistic	IPO	OLS-ML
Excess Mean	0.0914	0.0913
Volatility	0.1552	0.1552
Sharpe	0.5896	0.5896
VaR	-0.0495	-0.0495
Avg DD	-0.0338	-0.0337
CDaR	-0.2300	-0.2296
Herf Indx	0.1087	0.1091

(c) FF3-DCC

Statistic	IPO	OLS-ML
Excess Mean	0.0908	0.0921
Volatility	0.1547	0.1547
Sharpe	0.5873	0.5954
VaR	-0.0494	-0.0494
Avg DD	-0.0327	-0.0330
CDaR	-0.2289	-0.2284
Herf Indx	0.1079	0.1080

(d) FF5-DCC

Table B.2: Out-of-sample economic and risk metrics of the IPO and OLS-ML methods for the constrained equal-risk-contribution portfolio.



(a) FF3-CCC

(b) FF5-CCC.

(c) FF3-DCC

(d) FF5-DCC.

Figure B.4: Out-of-sample equity growth of the IPO and OLS-ML methods for the constrained max-diversification portfolio.

B.5 Data Summary

GICS Sector	Stock Symbols							
Communication	CBB	CMCSA	DIS	FOX	IPG	LUMN	MDP	NYT
Services	T	VOD	VZ					
Consumer	BBY	CBRL	CCL	F	GPC	GPS	GT	HAS
Discretionary	HD	HOG	HRB	JWN	LB	LEG	LEN	LOW
	MCD	NKE	NVR	NWL	PHM	PVH	ROST	TGT
	TJX	VFC	WHR	WWW				
Consumer	ADM	ALCO	CAG	CASY	CHD	CL	CLX	COST
Staples	CPB	FLO	GIS	HSY	K	KMB	KO	KR
	MO	PEP	PG	SYY	TAP	TR	TSN	UVV
	WBA	WMK	WMT					
Energy	AE	APA	BKR	BP	COP	CVX	EOG	HAL
	HES	MRO	OKE	OXY	SLB	VLO	WMB	XOM
Financials	AFG	AFL	AIG	AJG	AON	AXP	BAC	BEN
	BK	BXS	C	GL	JPM	L	LNC	MMC
	PGR	PNC	RJF	SCHW	STT	TROW	TRV	UNM
	USB	WFC	WRB	WTM				
Health	ABMD	ABT	AMGN	BAX	BDX	BIO	BMY	CAH
Care	CI	COO	CVS	DHR	HUM	JNJ	LLY	MDT
	MRK	OMI	PFE	PKI	SYK	TFX	TMO	VTRS
	WST							
Industrials	ABM	AIR	ALK	AME	AOS	BA	CAT	CMI
	CSL	CSX	DE	DOV	EFX	EMR	ETN	FDX
	GD	GE	GWW	HON	IEX	ITW	JCI	KSU
	LMT	LUV	MAS	MMM	NOC	NPK	NSC	PCA
	RPH	PNR	ROK	ROL	RTX	SNA	SWK	TXT
	UNP							
Information	AAPL	ADBE	ADI	ADP	ADSK	AMAT	AMD	GLW
Technology	HPQ	IBM	INTC	MSFT	MSI	MU	ORCL	ROG
	SWKS	TER	TXN	TYL	WDC	XRX		
Materials	APD	AVY	BLL	CCK	CRS	ECL	FMC	GLT
	IFF	IP	MOS	NEM	NUE	OLN	PPG	SEE
	SHW	SON	VMC					
Real	ALX	FRT	GTY	HST	PEAK	PSA	VNO	WRI
Estate	WY							
Utilities	AEP	ATO	BKH	CMS	CNP	D	DTE	DUK
	ED	EIX	ETR	EVRG	EXC	LNT	NEE	NFG
	NI	NJR	OGE	PEG	PNM	PNW	PPL	SJW
	SO	SWX	UGI	WEC	XEL			

Table B.3: U.S. stock data, sorted by GICS Sector. Data provided by Quandl.

Appendix C

Efficient differentiable quadratic programming layers: an ADMM approach

C.1 Proof of Proposition 10

We define $\mathbf{v}^k = \mathbf{x}^{k+1} + \boldsymbol{\mu}^k$. We can therefore express Equation (7.14b) as:

$$\mathbf{z}^{k+1} = \Pi(\mathbf{x}^{k+1} + \boldsymbol{\mu}^k) = \Pi(\mathbf{v}^k), \tag{C.1}$$

and Equation (7.14c) as:

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1} = \mathbf{v}^k - \Pi(\mathbf{v}^k). \tag{C.2}$$

Substituting Equations (C.1) and (C.2) into Equation (7.14a) gives the desired fixed-point iteration:

$$\begin{bmatrix} \mathbf{v}^{k+1} \\ \boldsymbol{\eta}^{k+2} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{k+2} + \boldsymbol{\mu}^{k+1} \\ \boldsymbol{\eta}^{k+2} \end{bmatrix} \tag{C.3}$$

$$= - \begin{bmatrix} \mathbf{Q} + \rho \mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \rho(\mathbf{z}^{k+1} - \boldsymbol{\mu}^{k+1}) \\ -\mathbf{b} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mu}^{k+1} \\ 0 \end{bmatrix} \tag{C.4}$$

$$= - \begin{bmatrix} \mathbf{Q} + \rho \mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \rho(2\Pi(\mathbf{v}^k) - \mathbf{v}^k) \\ -\mathbf{b} \end{bmatrix} + \begin{bmatrix} \mathbf{v}^k \\ \boldsymbol{\eta}^{k+1} \end{bmatrix} - \begin{bmatrix} \Pi(\mathbf{v}^k) \\ \boldsymbol{\eta}^{k+1} \end{bmatrix}. \tag{C.5}$$

C.2 Proof of Proposition 11

We define $F \colon \mathbb{R}^{d_v} \times \mathbb{R}^{d_\eta} \to \mathbb{R}^{d_v} \times \mathbb{R}^{d_\eta}$ as:

$$F(\mathbf{v}, \boldsymbol{\eta}) = - \begin{bmatrix} \mathbf{Q} + \rho \mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \rho(2\Pi(\mathbf{v}) - \mathbf{v}) \\ -\mathbf{b} \end{bmatrix} + \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\eta} \end{bmatrix} - \begin{bmatrix} \Pi(\mathbf{v}) \\ \boldsymbol{\eta} \end{bmatrix}, \tag{C.6}$$

and let

$$\mathbf{M} = \begin{bmatrix} \mathbf{Q} + \rho \mathbf{I}_{\mathbf{v}} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix}. \tag{C.7}$$

Therefore we have

$$\mathbf{M} F(\mathbf{v}, \boldsymbol{\eta}) = - \begin{bmatrix} \mathbf{p} - \rho(2\Pi(\mathbf{v}) - \mathbf{v}) \\ -\mathbf{b} \end{bmatrix} + \mathbf{M} \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\eta} \end{bmatrix} - \mathbf{M} \begin{bmatrix} \Pi(\mathbf{v}) \\ \boldsymbol{\eta} \end{bmatrix}. \tag{C.8}$$

Taking the partial differentials of Equation (C.8) with respect to the relevant problem variables therefore gives:

$$\begin{aligned}\mathbf{M}\,\partial F(\mathbf{v},\boldsymbol{\eta}) &= -\begin{bmatrix}\partial\,\mathbf{p}\\ -\partial\,\mathbf{b}\end{bmatrix} + \partial\,\mathbf{M}\begin{bmatrix}\mathbf{v}\\ \boldsymbol{\eta}\end{bmatrix} - \partial\,\mathbf{M}\begin{bmatrix}\Pi(\mathbf{v})\\ \boldsymbol{\eta}\end{bmatrix} - \partial\,\mathbf{M}\,F(\mathbf{v},\boldsymbol{\eta})\\ &= -\begin{bmatrix}\partial\,\mathbf{p}\\ -\partial\,\mathbf{b}\end{bmatrix} - \partial\,\mathbf{M}\left[-\mathbf{M}^{-1}\begin{bmatrix}\mathbf{p}-\rho(2\Pi(\mathbf{v})-\mathbf{v})\\ -\mathbf{b}\end{bmatrix}\right]\\ &= -\begin{bmatrix}\partial\,\mathbf{p}\\ -\partial\,\mathbf{b}\end{bmatrix} - \partial\,\mathbf{M}\begin{bmatrix}\mathbf{x}^*\\ \boldsymbol{\eta}^*\end{bmatrix}\\ &= -\begin{bmatrix}\partial\,\mathbf{p}+\frac{1}{2}(\partial\,\mathbf{Q}\,\mathbf{x}^*+\partial\,\mathbf{Q}^T\,\mathbf{x}^*)+\partial\,\mathbf{A}^T\,\boldsymbol{\eta}^*\\ -\partial\,\mathbf{b}+\partial\,\mathbf{A}\,\mathbf{x}^*\end{bmatrix}.\end{aligned} \tag{C.9}$$

From Equation (C.8) we have that the differential $\partial F(\mathbf{v},\boldsymbol{\eta})$ is given by:

$$\partial F(\mathbf{v},\boldsymbol{\eta}) = -\,\mathbf{M}^{-1}\begin{bmatrix}\partial\,\mathbf{p}+\frac{1}{2}(\partial\,\mathbf{Q}\,\mathbf{x}^*+\partial\,\mathbf{Q}^T\,\mathbf{x}^*)+\partial\,\mathbf{A}^T\,\boldsymbol{\eta}^*\\ -\partial\,\mathbf{b}+\partial\,\mathbf{A}\,\mathbf{x}^*\end{bmatrix}. \tag{C.10}$$

Substituting the gradient action of Equation (C.10) into Equation (7.22) and taking the left matrix-vector product of the transposed Jacobian with the previous backward-pass gradient, $\frac{\partial \ell}{\partial\,\mathbf{z}^*}$, gives the desired result.

$$\begin{bmatrix}\hat{\mathbf{d}}_{\mathbf{x}}\\ \hat{\mathbf{d}}_{\boldsymbol{\eta}}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}+\rho\,\mathbf{I}_{\mathbf{v}} & \mathbf{A}^T\\ \mathbf{A} & 0\end{bmatrix}^{-1}\left[\mathbf{I}_{\tilde{\mathbf{v}}}-\nabla_{\tilde{\mathbf{v}}}F(\tilde{\mathbf{v}}(\boldsymbol{\theta}),\boldsymbol{\theta})\right]^{-T}\begin{bmatrix}D\Pi(\mathbf{v}) & 0\\ 0 & \mathbf{I}_{\boldsymbol{\eta}}\end{bmatrix}\begin{bmatrix}\left(-\frac{\partial \ell}{\partial\,\mathbf{z}^*}\right)^T\\ 0\end{bmatrix}. \tag{C.11}$$

From Equation (7.20) we have:

$$\mathbf{I}_{\tilde{\mathbf{v}}}-\nabla_{\tilde{\mathbf{v}}}F(\tilde{\mathbf{v}}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \begin{bmatrix}\mathbf{Q}+\rho\,\mathbf{I}_{\mathbf{v}} & \mathbf{A}^T\\ \mathbf{A} & \mathbf{0}\end{bmatrix}^{-1}\begin{bmatrix}-\rho(2D\Pi(\mathbf{v})-\mathbf{I}_{\mathbf{v}}) & 0\\ 0 & 0\end{bmatrix}+\begin{bmatrix}D\Pi(\mathbf{v}) & 0\\ 0 & \mathbf{I}_{\boldsymbol{\eta}}\end{bmatrix}. \tag{C.12}$$

Simplifying Equation (C.11) with Equation (C.12) yields the final expression:

$$\begin{bmatrix}\hat{\mathbf{d}}_{\mathbf{x}}\\ \hat{\mathbf{d}}_{\boldsymbol{\eta}}\end{bmatrix} = \left[\begin{bmatrix} D\Pi(\mathbf{v}) & 0\\ 0 & \mathbf{I}_{\boldsymbol{\eta}}\end{bmatrix}\begin{bmatrix}\mathbf{Q}+\rho\,\mathbf{I}_{\mathbf{v}} & \mathbf{A}^T\\ \mathbf{A} & 0\end{bmatrix} + \begin{bmatrix}-\rho(2D\Pi(\mathbf{v})-\mathbf{I}_{\mathbf{v}}) & 0\\ 0 & 0\end{bmatrix}\right]^{-1}\begin{bmatrix} D\Pi(\mathbf{v}) & 0\\ 0 & \mathbf{I}_{\boldsymbol{\eta}}\end{bmatrix}\begin{bmatrix}\big(-\frac{\partial \ell}{\partial \mathbf{z}^*}\big)^T\\ 0\end{bmatrix}. \tag{C.13}$$

C.3 Proof of Proposition 12

From the KKT system of equations (4.12) we have:

$$\mathbf{G}^T \operatorname{diag}(\tilde{\boldsymbol{\lambda}}^*)\hat{\mathbf{d}}_{\boldsymbol{\lambda}} = \operatorname{diag}(\rho\,\boldsymbol{\mu}^*)\hat{\mathbf{d}}_{\boldsymbol{\lambda}} = \Big(-\Big(\frac{\partial \ell}{\partial\, \mathbf{z}^*}\Big)^T - \mathbf{Q}\,\hat{\mathbf{d}}_{\mathbf{x}} - \mathbf{A}^T\,\hat{\mathbf{d}}_{\boldsymbol{\eta}}\Big). \tag{C.14}$$

From Equation (4.11) it follows that:

$$\frac{\partial \ell}{\partial \mathbf{l}} \neq 0 \iff \boldsymbol{\lambda}^*_- > 0 \quad \text{and} \quad \frac{\partial \ell}{\partial\, \mathbf{u}} \neq 0 \iff \boldsymbol{\lambda}^*_+ > 0, \tag{C.15}$$

and therefore Equation (C.14) uniquely determines the relevant non-zero gradients. Let $\tilde{\boldsymbol{\mu}}^*$ be as defined by Equation (7.25), then it follows that:

$$\hat{\mathbf{d}}_{\boldsymbol{\lambda}} = \operatorname{diag}(\rho\tilde{\boldsymbol{\mu}}^*)^{-1}\Big(-\Big(\frac{\partial \ell}{\partial\, \mathbf{z}^*}\Big)^T - \mathbf{Q}\,\hat{\mathbf{d}}_{\mathbf{x}} - \mathbf{A}^T\,\hat{\mathbf{d}}_{\boldsymbol{\eta}}\Big). \tag{C.16}$$

Substituting $\hat{\mathbf{d}}_{\boldsymbol{\lambda}}$ into Equation (7.16) gives the desired gradients.

C.4 Experiment 1: relative performance

C.4.1 Experiment 1: relative performance

		ADMM FP	ADMM KKT	ADMM Unroll	OptNet	SCS
	backward	1.0x	0.8x	0.4x	0.8x	0.7x
$\epsilon = 10^{-1}$	forward	1.0x	1.0x	1.0x	1.4x	0.9x
$d_z = 10$	total	1.0x	0.9x	0.8x	1.3x	0.9x
	backward	1.0x	0.8x	1.0x	0.6x	0.7x
$\epsilon = 10^{-3}$	forward	1.0x	1.0x	1.0x	1.0x	0.4x
$d_z = 10$	total	1.0x	1.0x	1.0x	0.9x	0.5x
	backward	1.0x	0.8x	1.9x	0.6x	0.7x
$\epsilon = 10^{-5}$	forward	1.0x	1.0x	1.0x	0.6x	0.2x
$d_z = 10$	total	1.0x	1.0x	1.1x	0.6x	0.3x
		ADMM FP	ADMM KKT	ADMM Unroll	OptNet	SCS
	backward	1.0x	1.9x	0.9x	1.2x	1.9x
$\epsilon = 10^{-1}$	forward	1.0x	1.0x	1.0x	2.2x	2.4x
$d_z = 50$	total	1.0x	1.2x	1.0x	2.0x	2.3x
	backward	1.0x	1.9x	1.8x	1.2x	1.9x
$\epsilon = 10^{-3}$	forward	1.0x	1.0x	1.0x	1.7x	1.3x
$d_z = 50$	total	1.0x	1.1x	1.1x	1.6x	1.4x
	backward	1.0x	1.8x	2.9x	1.2x	1.9x
$\epsilon = 10^{-5}$	forward	1.0x	1.0x	1.0x	1.1x	0.9x
$d_z = 50$	total	1.0x	1.0x	1.1x	1.1x	1.0x
		ADMM FP	ADMM KKT	ADMM Unroll	OptNet	SCS
	backward	1.0x	3.3x	1.7x	2.1x	3.8x
$\epsilon = 10^{-1}$	forward	1.0x	1.0x	1.2x	4.2x	5.7x
$d_z = 100$	total	1.0x	1.8x	1.3x	3.5x	5.1x
	backward	1.0x	3.3x	3.0x	2.1x	3.8x
$\epsilon = 10^{-3}$	forward	1.0x	1.0x	1.1x	3.1x	3.5x
$d_z = 100$	total	1.0x	1.5x	1.5x	2.9x	3.6x
	backward	1.0x	3.2x	4.5x	2.3x	3.8x
$\epsilon = 10^{-5}$	forward	1.0x	1.0x	1.1x	2.8x	2.6x
$d_z = 100$	total	1.0x	1.3x	1.6x	2.7x	2.8x
		ADMM FP	ADMM KKT	ADMM Unroll	OptNet	SCS
	backward	1.0x	6.9x	1.6x	3.7x	7.4x
$\epsilon = 10^{-1}$	forward	1.0x	1.0x	1.0x	13.1x	12.9x
$d_z = 250$	total	1.0x	3.6x	1.3x	9.0x	10.5x
	backward	1.0x	7.0x	3.6x	3.6x	7.4x
$\epsilon = 10^{-3}$	forward	1.0x	1.0x	1.1x	11.3x	8.9x
$d_z = 250$	total	1.0x	3.0x	1.9x	8.7x	8.4x
	backward	1.0x	6.9x	6.3x	4.5x	7.7x
$\epsilon = 10^{-5}$	forward	1.0x	1.0x	1.1x	11.1x	6.7x
$d_z = 250$	total	1.0x	2.4x	2.3x	9.5x	7.0x
		ADMM FP	ADMM KKT	ADMM Unroll	OptNet	SCS
	backward	1.0x	8.6x	1.4x	3.9x	9.0x
$\epsilon = 10^{-1}$	forward	1.0x	1.0x	1.0x	18.9x	15.8x
$d_z = 500$	total	1.0x	5.1x	1.2x	10.8x	12.2x
	backward	1.0x	8.5x	2.6x	3.6x	9.0x
$\epsilon = 10^{-3}$	forward	1.0x	0.9x	0.9x	17.5x	11.1x
$d_z = 500$	total	1.0x	4.3x	1.7x	11.3x	10.2x
	backward	1.0x	8.6x	8.2x	4.1x	9.0x
$\epsilon = 10^{-5}$	forward	1.0x	1.0x	1.0x	17.4x	6.0x
$d_z = 500$	total	1.0x	3.0x	2.9x	13.9x	6.8x
		ADMM FP	ADMM KKT	ADMM Unroll	OptNet	SCS
	backward	1.0x	13.5x	1.1x	6.6x	15.0x
$\epsilon = 10^{-1}$	forward	1.0x	1.0x	1.0x	23.9x	16.4x
$d_z = 1000$	total	1.0x	7.3x	1.0x	15.1x	15.7x
	backward	1.0x	13.4x	2.1x	6.6x	15.0x
$\epsilon = 10^{-3}$	forward	1.0x	1.0x	1.0x	27.3x	14.5x
$d_z = 1000$	total	1.0x	6.8x	1.5x	17.6x	14.7x
	backward	1.0x	13.4x	11.3x	6.7x	14.5x
$\epsilon = 10^{-5}$	forward	1.0x	1.0x	1.0x	19.7x	7.4x
$d_z = 1000$	total	1.0x	4.3x	3.8x	16.2x	9.3x

Table C.1: Computational performance of ADMM KKT, ADMM Unroll, Optnet and SCS relative to ADMM FP for various problem sizes, d_z, and stopping tolerances. Batch size = 128.

Appendix D

Gradient boosting for convex cone predict and optimize problems

D.1 Proof of Proposition 13

The iterations in Equation (8.11) can be cast as a fixed-point iteration as follows. Let $\mathbf{w}^k = \tilde{\mathbf{u}}^{k+1} - \mathbf{v}^k$, then:

$$\mathbf{u}^{k+1} = \Pi_{\mathcal{C}}(\tilde{\mathbf{u}}^{k+1} - \mathbf{v}^k) = \Pi_{\mathcal{C}}(\mathbf{w}^k), \tag{D.1}$$

and:

$$\mathbf{v}^{k+1} = \mathbf{v}^k + \mathbf{u}^{k+1} - \tilde{\mathbf{u}}^{k+1} = \Pi_{\mathcal{C}}(\mathbf{w}^k) - \mathbf{w}^k\,. \tag{D.2}$$

Substituting Equations (D.1) and (D.2) into Equation (8.11a) gives the desired fixed-point iteration:

$$\begin{aligned}\mathbf{w}^{k+1} &= \tilde{\mathbf{u}}^{k+2} - \mathbf{v}^{k+1} \\ &= (\mathbf{I}_{\mathbf{u}} + \mathbf{M})^{-1}(\mathbf{u}^{k+1} + \mathbf{v}^{k+1} - \mathbf{q}) - \mathbf{v}^{k+1} \\ &= (\mathbf{I}_{\mathbf{w}} + \mathbf{M})^{-1}(2\Pi_{\mathcal{C}}(\mathbf{w}^k) - \mathbf{w}^k - \mathbf{q}) + \mathbf{w}^k - \Pi_{\mathcal{C}}(\mathbf{w}^k).\end{aligned} \tag{D.3}$$

D.2 Proof of Proposition 14

We define $F\colon \mathbb{R}^{d_z+d_y} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z+d_y}$ as:

$$F(\mathbf{w}, \boldsymbol{\theta}) = (\mathbf{I}_{\mathbf{w}} + \mathbf{M})^{-1}(2\Pi_{\mathcal{C}}(\mathbf{w}) - \mathbf{w} - \mathbf{q}) + \mathbf{w} - \Pi_{\mathcal{C}}(\mathbf{w}). \tag{D.4}$$

The Jacobian, $\nabla_{\mathbf{w}} F$, is therefore defined as:

$$\nabla_{\mathbf{w}} F = (\mathbf{I_w} + \mathbf{M})^{-1}(2D\Pi_{\mathcal{C}}(\mathbf{w}) - \mathbf{I_w}) + \mathbf{I_w} - D\Pi_{\mathcal{C}}(\mathbf{w}). \tag{D.5}$$

We perform the left multiplication of F by $(\mathbf{I_w} + \mathbf{M})$ and compute the partial derivative of F with respect to all other problem variables at the fixed-point $\mathbf{w}^*$ as follows:

$$\begin{aligned}
(\mathbf{I_w} + \mathbf{M})\partial F(\mathbf{w}^*) &= -\partial\,\mathbf{q} + \partial\,\mathbf{M}\,\mathbf{w}^* - \partial\,\mathbf{M}\,\Pi_{\mathcal{C}}(\mathbf{w}^*) - \partial\,\mathbf{M}\,F \\
&= -\partial\,\mathbf{q} + \partial\,\mathbf{M}(\mathbf{w}^* - \Pi_{\mathcal{C}}(\mathbf{w}^*) - F(\mathbf{w}^*)) \\
&= -\partial\,\mathbf{q} - \partial\,\mathbf{M}(\mathbf{I_w} + \mathbf{M})^{-1}(2\Pi_{\mathcal{C}}(\mathbf{w}^*) - \mathbf{w}^* - \mathbf{q}) \\
&= -\partial\,\mathbf{q} - \partial\,\mathbf{M}\,\mathbf{u}^*
\end{aligned} \tag{D.6}$$

It follows then that $\partial F(\mathbf{w}^*)$ is given by:

$$\partial F(\mathbf{w}^*) = \left[\,\mathbf{I_w} + \mathbf{M}\,\right]^{-1} \begin{bmatrix} \partial\,\mathbf{c} + \frac{1}{2}(\partial\,\mathbf{P} + \partial\,\mathbf{P}^T)\,\mathbf{z}^* + \partial\,\mathbf{A}^T\,\mathbf{y}^* \\ \partial\,\mathbf{b} - \partial\,\mathbf{A}\,\mathbf{z}^* \end{bmatrix} \tag{D.7}$$

Applying Corollary 1 we compute the gradient action of Equation (D.7) and the left matrix-vector product of the transposed Jacobian with the gradient, $\frac{\partial \ell}{\partial\,\mathbf{z}^*}$, to arrive at the desired result.

$$\begin{aligned}
\begin{bmatrix} \hat{\mathbf{d}}_{\mathbf{z}} \\ \hat{\mathbf{d}}_{\mathbf{y}} \end{bmatrix} &= \left[\,\mathbf{I_w} + \mathbf{M}\,\right]^{-T}\left[D\Pi_{\mathcal{C}}(\mathbf{w}^*) - (\mathbf{I_w} + \mathbf{M})^{-1}(2D\Pi_{\mathcal{C}}(\mathbf{w}^*) - \mathbf{I_w})\right]^{-T} D\Pi_{\mathcal{C}}(\mathbf{w}^*)^T \begin{bmatrix} -(\frac{\partial \ell}{\partial\,\mathbf{z}^*})^T \\ \mathbf{0} \end{bmatrix} \\
&= \left[(\mathbf{I_w} + \mathbf{M})D\Pi_{\mathcal{C}}(\mathbf{w}^*) - (2D\Pi_{\mathcal{C}}(\mathbf{w}^*) - \mathbf{I_w})\right]^{-T} D\Pi_{\mathcal{C}}(\mathbf{w}^*)^T \begin{bmatrix} (\frac{\partial \ell}{\partial\,\mathbf{z}^*})^T \\ \mathbf{0} \end{bmatrix}
\end{aligned} \tag{D.8}$$

Finally, the gradients of the loss function, ℓ, with respect to problem variables $\mathbf{P}$, $\mathbf{c}$, $\mathbf{A}$ and $\mathbf{b}$ are given by:

$$\begin{aligned}
\frac{\partial \ell}{\partial\,\mathbf{P}} &= \frac{1}{2}\left(\hat{\mathbf{d}}_{\mathbf{z}}\,\mathbf{z}^{*T} + \mathbf{z}^*\,\hat{\mathbf{d}}_{\mathbf{z}}^T\right) & \frac{\partial \ell}{\partial\,\mathbf{c}} &= \hat{\mathbf{d}}_{\mathbf{z}} \\
\frac{\partial \ell}{\partial\,\mathbf{A}} &= \mathbf{y}^*\,\hat{\mathbf{d}}_{\mathbf{z}}^T - \hat{\mathbf{d}}_{\mathbf{y}}\,\mathbf{z}^{*T} & \frac{\partial \ell}{\partial\,\mathbf{b}} &= \hat{\mathbf{d}}_{\mathbf{y}}
\end{aligned} \tag{D.9}$$