MOSEK Portfolio Optimization Cookbook

Release 1.6.0

MOSEK ApS

05 November 2025

Contents

1 Preface **1**
- 1.1 Purpose . . . 1
- 1.2 Content . . . 1
- 1.3 Online resources . . . 2

2 Markowitz portfolio optimization **3**
- 2.1 The mean–variance model . . . 3
- 2.2 Constraints, modifications . . . 6
- 2.3 Conic formulation . . . 8
- 2.4 Example . . . 9

3 Input data preparation **15**
- 3.1 Scenarios . . . 15
- 3.2 Modeling the distribution of returns . . . 17
- 3.3 Extensions . . . 19
- 3.4 Example . . . 23

4 Dealing with estimation error **26**
- 4.1 Estimation of mean . . . 26
- 4.2 Estimation of the covariance matrix . . . 29
- 4.3 Using constraints . . . 31
- 4.4 Improve the optimization . . . 32
- 4.5 Example . . . 34

5 Factor models **37**
- 5.1 Definition . . . 37
- 5.2 Classes of factor models . . . 38
- 5.3 Portfolio optimization with factor model . . . 41
- 5.4 Shrinkage as factor model . . . 42
- 5.5 Example . . . 43

6 Transaction costs **49**
- 6.1 Variable transaction costs . . . 50
- 6.2 Fixed transaction costs . . . 50
- 6.3 Market impact costs . . . 51
- 6.4 Cardinality constraints . . . 52
- 6.5 Buy-in threshold . . . 53
- 6.6 Example . . . 54

7 Benchmark relative portfolio optimization **61**
7.1 Active return . . . 61
7.2 Factor model on active returns . . . 62
7.3 Optimization . . . 62
7.4 Extensions . . . 63
7.5 Example . . . 64

8 Other risk measures **68**
8.1 Deviation measures . . . 69
8.2 Tail risk measures . . . 72
8.3 Expected utility maximization . . . 76
8.4 Example . . . 79

9 Risk budgeting **83**
9.1 Risk contribution . . . 83
9.2 Risk budgeting portfolio . . . 83
9.3 Risk budgeting with variance . . . 84
9.4 Example . . . 86

10 Robust optimization **90**
10.1 Types of uncertainty . . . 90
10.2 Uncertainty in security returns . . . 91
10.3 Uncertainty in the factor model . . . 91
10.4 Parameters . . . 94
10.5 Example . . . 96

11 Multi-period portfolio optimization **99**
11.1 Single-period model . . . 100
11.2 Multi-period model . . . 101
11.3 Example . . . 103

12 Regression and regularization **106**
12.1 Linear regression . . . 106
12.2 Regularization . . . 108
12.3 Example . . . 110

13 Appendix **114**
13.1 Conic optimization refresher . . . 114
13.2 Mixed-integer models . . . 120
13.3 Quadratic cones and riskless solution . . . 122
13.4 Monte Carlo Optimization Selection (MCOS) . . . 123

14 Notation and definitions **125**
14.1 Financial notations . . . 125
14.2 Mathematical notations . . . 126
14.3 Abbreviations . . . 127

Bibliography 128

© Copyright 2025 MOSEK ApS

Chapter 1

Preface

1.1 Purpose

This book provides an introduction to the topic of portfolio optimization and discusses several branches of practical interest from this broad subject.

We intended it to be a practical guide, a cookbook, that not only serves as a reference but also supports the reader with practical implementation. We do not assume that the reader is acquainted with portfolio optimization, thus the book can be used as a starting point, while for the experienced reader it can serve as a review.

First we familiarize the reader with the basic concepts and the most relevant approaches in portfolio optimization, then we also present computational examples with code to illustrate these concepts and to provide a basis for implementing more complex and more specific cases. We aim to keep the discussion concise and self-contained, covering only the main ideas and tools, the most important pitfalls, both from theoretical and from technical perspective. The reader is directed towards further reading in each subject through references.

1.2 Content

Each of the chapters is organized around a specific subject:

- Sec. 2 is a general introduction, and is recommended to be read first to familiarize with the problem formulation and notations used throughout this book. Here we also present a code example showing how to use **MOSEK** to model the problem and how to solve it.

- Sec. 3 summarizes concepts and pitfalls related to the preparation of raw data. Here we discuss how to arrive at security returns data suitable for optimization, starting from raw data that is commonly available on the internet. This chapter is recommended as a second read.

- Each of the subsequent chapters are focusing on a specific topic: mitigating estimation error, using factor models, modeling transaction costs, optimizing relative to a benchmark, optimizing different risk measures, risk budgeting, robust optimization, and finally the multiperiod portfolio optimization problem. These chapters are independent and thus can be read in any order.

- The book assumes basic familiarity with conic optimization. Sec. 13.1 can be used as a quick refresher in this topic. It shows how to convert traditional quadratic optimization (QO) and quadratically constrained quadratic optimization (QCQO) problems into equivalent quadratic conic optimization models, and what are the advantages of this conversion. Then it briefly introduces other types of cones as well.

1.3 Online resources

The code examples appearing in this cookbook are written using the **MOSEK** Fusion API for Python, and focus on the modeling part, omitting error handling, status checks, logging and other technical facilities. The full technical documentation, as well as manuals for **MOSEK** APIs in other programming languages can be found at:

https://www.mosek.com/documentation

The notebooks with full executable code examples are available at GitHub:

https://github.com/MOSEK/PortfolioOptimization

Visit **MOSEK** at:

https://www.mosek.com

The cookbook, code examples, manuals for other APIs and other portfolio optimization resources can be found at the webpage of this publication:

https://www.mosek.com/content/portfolio-optimization

Chapter 2

Markowitz portfolio optimization

In this section we introduce the Markowitz model in portfolio optimization, and discuss its different formulations and the most important input parameters.

2.1 The mean–variance model

Consider an investor who wishes to allocate capital among N securities at time $t = 0$ and hold them over a single period of time until $t = h$. We denote $p_{0,i}$ the (known) price of security i at the beginning of the investment period and $P_{h,i}$ the (random) price of security i at the end of the investment period $t = h$. The *rate of return* of security i over period h is then modeled by the random variable $R_i = P_{h,i}/p_{0,i} - 1$, and its expected value is denoted by $\mu_i = \mathbb{E}(R_i)$. The risk-averse investor seeks to maximize the return of the investment, while trying to keep the investment *risk*, i. e., the uncertainty of the future security returns R_i on an acceptable low level.

The part of investment risk called specific risk (the risk associated with each individual security) can be reduced (for a fixed expected rate of return) through diversification, which means the selection of multiple unrelated securities into a portfolio.[1] Modern Portfolio Theory formalized this process by using variance as a measure of portfolio risk, and constructing a optimization problem.

Thus we make the investment decision at time $t = 0$ by specifying the N-dimensional decision vector $\mathbf{x}$ called *portfolio*, where x_i is the fraction of funds invested into security i. We can then express the random portfolio return as $R_\mathbf{x} = \sum_i x_i R_i = \mathbf{x}^\mathsf{T} R$, where R is the vector of security returns. The optimal $\mathbf{x}$ is given based on the following inputs of the portfolio optimization problem:

- The expected portfolio return

$$\mu_\mathbf{x} = \mathbb{E}(R_\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbb{E}(R) = \mathbf{x}^\mathsf{T}\mu.$$

- The portfolio variance

$$\sigma_\mathbf{x}^2 = \mathrm{Var}(R_\mathbf{x}) = \sum_i \mathrm{Cov}(R_i, R_j) x_i x_j = \mathbf{x}^\mathsf{T}\Sigma\mathbf{x}.$$

[1] The other component of risk called systematic risk can only be reduced by decreasing the expected return.

Here μ is the vector of expected returns, Σ is the covariance matrix of returns, summarizing the risks associated with the securities. Note that the covariance matrix is symmetric positive semidefinite by definition, and if we assume that none of the securities is redundant[2], then it is positive definite. This can also be seen if we consider that the portfolio variance must always be a positive number. After the above parameters the problem is also referred to as mean–variance optimization (MVO). The choice of variance as the risk measure results that MVO is a *quadratic optimization* (QO) problem.[3]

Using these input parameters, the MVO problem seeks to select a portfolio of securities $\mathbf{x}$ in such a way that it finds the optimal tradeoff between expected portfolio return and portfolio risk. In other words it seeks to maximize the return while limiting the level of risk, or it wants to minimize the risk while demanding at least a given level of return. Thus portfolio optimization can be seen as a bi-criteria optimization problem.

2.1.1 Solution of the mean–variance model

To be able to solve the portfolio optimization problem, we have to assume that the investor knows the value of the expected return vector μ and covariance matrix Σ. In practice it is only possible to have estimates, which we will denote by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ respectively. (Details on the properties of these estimates and ways to improve them will be the topic of Sec. 4.)

In the simplest form of the MVO problem only risky securities are considered. Also the portfolio is fully invested with no initial holdings, implying the linear equality constraint $\sum_i x_i = \mathbf{1}^\mathsf{T}\mathbf{x} = 1$, where $\mathbf{1}$ denotes a vector of ones.

We can formulate this portfolio optimization problem in three equivalent ways:

1. Minimize the portfolio risk, with the constraint expressing a lower bound on the portfolio return:

$$\begin{array}{lrcl} \text{minimize} & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & & \\ \text{subject to} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & \geq & r_{\min}, \\ & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.1}$$

   Here $r_{\min}$ is the target expected return, which the investor wishes to reach. In this problem the objective function is convex quadratic, all the constraints are linear, thus it is a quadratic optimization (QO) problem. While QO problems are easy to solve, this formulation is not ideal in practice because investors might prefer specifying an upper bound on the portfolio risk instead of specifying a lower bound on the expected portfolio return.

2. Maximize the expected portfolio return, with the constraint expressing an upper bound on the portfolio risk:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.2}$$

[2] A security is redundant within the universe of securities in consideration, if there is a *replicating portfolio*, i. e., a combination of other securities in the universe, that has the same properties. It means that we can drop the redundant security from the universe without loss of opportunities for diversification. Keeping it in the portfolio could cause the covariance matrix Σ to be singular.

[3] This was a reasonable choice from both the financial and the mathematical point of view, because modern portfolio theory and the theory of quadratic optimization started to develop about the same time in history, in the 1950s. However, the way we model risk can be different. See later in Sec. 8.

In this formulation it is possible to explicitly constrain the risk measure (the variance), which makes it more practical. We can also add constraints on multiple types of risk measures more easily. However, the problem in this form is not a QO because of the quadratic constraint. We can reformulate it as a conic problem; see Sec. 2.3.

3. Maximize the utility function of the investor:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \frac{\delta}{2}\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.3}$$

In this case we construct the (concave) quadratic utility function $\boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \frac{\delta}{2}\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x}$ to represent the risk-averse investor's preferred tradeoff between portfolio return and portfolio risk. This preference can be adjusted using the *risk-aversion coefficient* δ. However this parameter might not have intuitive investment meaning for the investor.

We get a variant of this optimization problem by using the standard deviation instead of the variance:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \tilde{\delta}\sqrt{\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x}} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.4}$$

This form is favorable because then the standard deviation penalty term will be of the same scale as portfolio return.

Moreover, if we assume portfolio return to be normally distributed, then $\tilde{\delta}$ has a more tangible meaning; it is the z-score[4] of portfolio return. Also, the objective function will be the $(1-\alpha)$-quantile of portfolio return, which is the opposite of the α confidence level *value-at-risk* (VaR). The number α comes from $\Phi(-\tilde{\delta}) = 1 - \alpha$, where Φ is the cumulative distribution function of the normal distribution.

We can see that for $\tilde{\delta} = 0$ we maximize expected portfolio return. Then by increasing $\tilde{\delta}$ we put more and more weight on tail risk, i. e., we maximize a lower and lower quantile of portfolio return. This makes selection of $\tilde{\delta}$ more intuitive in practice. Note that computing quantiles is more complicated for other distributions, because in general they are not determined by only mean and standard deviation.

These two problems will result in the same set of optimal solutions as the other two formulations. They also allow an easy computation of the entire efficient frontier. Maximizing problem (2.3) is again a QO, but problem (2.4) is not because of the square root. This latter problem can be solved using conic reformulation, as we will see in Sec. 2.3.

The optimal portfolio $\mathbf{x}$ computed by the Markowitz model is *efficient* in the sense that there is no other portfolio giving a strictly higher return for the same amount of risk or a strictly lower risk for the same amount of return. In other words an efficient portfolio is Pareto optimal. The collection of such points form the *efficient frontier* in mean return–variance space.

The above introduced simple formulations are trivial to solve. Through the method of Lagrangian multipliers we can get analytical formulas, which involve $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ as

[4] The z-score is the distance from the mean measured in units of standard deviation.

parameters. Thus the solution is simply to invert the covariance matrix (see e. g. in [CJPT18]). This makes them useful as a demonstration example, but they have only limited investment value.

2.2 Constraints, modifications

In investment practice additional constraints are essential, and it is common to add linear equalities and inequalities, convex inequalities, and/or extra objective function terms to the model, which represent various restrictions on the optimal security weights.

Constraints can reflect investment strategy or market outlook information, they can be useful for imposing quality controls on the portfolio management process, or for controlling portfolio structure and avoiding inadvertent risk exposures. In these more realistic use cases however, only numerical optimization algorithms are suitable for finding the solution.

In the following, we discuss some of the constraints commonly added to portfolio optimization problems.

2.2.1 Budget constraint

In general we can assume that we have $\mathbf{x}_0$ fraction of initial holdings, and x_0^{f} fraction of (risk-free) initial wealth to invest into risky securities. Then $\mathbf{1}^{\mathsf{T}}\mathbf{x}_0 + x_0^{\mathrm{f}} = 1$ is an initial condition, meaning that both the risky and the risk-free securities take up some fraction of the initial portfolio. After portfolio optimization, the portfolio composition changes, but no cash is added to or taken out from the portfolio. Thus the condition

$$\mathbf{1}^{\mathsf{T}}\mathbf{x} + x^{\mathrm{f}} = 1 \tag{2.5}$$

has to hold. For the differences $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ and $\tilde{x}^{\mathrm{f}} = x^{\mathrm{f}} - x_0^{\mathrm{f}}$ it follows that $\mathbf{1}^{\mathsf{T}}\tilde{\mathbf{x}} + \tilde{x}^{\mathrm{f}} = 0$ has to hold.

If we do not want to keep any wealth in the risk-free security, then in addition x^{f} must be zero.

2.2.2 Diversification constraints

These constraints can help limit portfolio risk by limiting the exposure to individual positions or industries, sectors.

For a single position, they can be given in the form

$$l_i \leq x_i \leq u_i.$$

For a group of securities such as an industry or sector, represented by the set $\mathcal{I}$ they will be

$$l_i \leq \sum_{i \in \mathcal{I}} x_i \leq u_i.$$

We can also create a constraint that limits the total fraction of the m largest investments to at most p. Based on Sec. 13.1.1 this is represented by the following set of constraints:

$$mt + \mathbf{1}^T\mathbf{u} \leq p, \quad \mathbf{u} + t \geq \mathbf{x}, \quad \mathbf{u} \geq 0,$$

where $\mathbf{u}$ and t are new variables.

2.2.3 Leverage constraints

Componentwise short sale limit

The simplest form of leverage constraint is when we do not allow any short selling. This is the long-only constraint stated as

$$\mathbf{x} \geq 0.$$

We can allow some amount of short selling s_i on security i by stating

$$x_i \geq -s_i.$$

Total short sale limit

We can also bound the total short selling by the constraint

$$\sum_i \max(-x_i, 0) \leq S.$$

We can rewrite this as a set of linear constraints by modeling the maximum function based on Sec. 13.1.1 using an auxiliary vector $\mathbf{t}^-$:

$$\mathbf{1}^\mathsf{T}\mathbf{t}^- \leq S, \quad \mathbf{t}^- \geq -\mathbf{x}, \quad \mathbf{t}^- \geq 0.$$

Collateralization requirement

An interesting variant of this constraint is the *collateralization requirement*, which limits the total of short positions to a fraction of the total of long positions. We can model it by introducing new variables $\mathbf{x}^+$ and $\mathbf{x}^-$ for the long and short part of $\mathbf{x}$, based on Sec. 13.1.1. Then the collateralization requirement will be:

$$\sum_i x_i^- \leq c \sum_i x_i^+.$$

Leverage strategy

A leverage strategy is to do short selling, then use the proceeds to buy other securities. We can express such a strategy in general using two constraints:

- $\mathbf{1}^\mathsf{T}\mathbf{x} = 1$,
- $\sum_i |x_i| \leq c$ or equivalently $\|\mathbf{x}\|_1 \leq c$,

where $c = 1$ yields the long-only constraint and $c = 1.6$ means the 130/30 strategy. The 1-norm constraint is nonlinear, but can be modeled as a linear constraint based on Sec. 13.1.1: $-\mathbf{z} \leq \mathbf{x} \leq \mathbf{z}$, $\mathbf{1}^\mathsf{T}\mathbf{z} = c$.

2.2.4 Turnover constraints

The turnover constraint limits the total change in the portfolio positions, which can help limiting e. g. taxes and cost of transactions.

Suppose $\mathbf{x}_0$ is the initial holdings vector. Then we can write the turnover constraint as

$$\|\mathbf{x} - \mathbf{x}_0\|_1 \leq c.$$

This nonlinear expression can be modeled as linear constraint using Sec. 13.1.1.

2.2.5 Practical considerations

We can of course add many other constraints that might be imposed by regulations or arise from specific investor preferences. As long as all of these are linear or conic, the problem will not become significantly harder to solve. (See Sec. 13.1 for a summary on conic constraints.)

However, we must not add too many constraints. Overconstraining a portfolio can lead to infeasibility of the optimization problem, or even if there exists a solution, out-of-sample performance of it could be poor.

Throughout this book the general notation $\mathbf{x} \in \mathcal{F}$ will indicate any further constraints added to the problem, that are not relevant to the given example. Then we can write the general MVO problem in the form

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{2.6}$$

2.3 Conic formulation

We have seen that some of the portfolio optimization problems in Sec. 2.1.1 are quadratic optimization problems. Solving QO problems in their original form is popular and considered easy, because this model was studied starting from early in history (in the 1950s), allowing it to become a well known and mature approach. However, more recent results show us that conic optimization models can improve on QO models both in the theoretical and in the practical sense. More on this in Sec. 13.1.2.

In this section we will show to convert problems (2.1)-(2.4) into conic form. Assuming that the covariance matrix estimate $\boldsymbol{\Sigma}$ is positive definite, it is possible to decompose it as $\boldsymbol{\Sigma} = \mathbf{G}\mathbf{G}^\mathsf{T}$, where $\mathbf{G} \in \mathbb{R}^{N\times k}$. We can do this for example by Cholesky decomposition, see Sec. 13.1.1 for a list of other possibilities. An interesting approach in financial applications is the *factor model*, which can have the advantage of having a direct financial interpretation (see in Sec. 5).

Even if there are no redundant securities, $\boldsymbol{\Sigma}$ can be positive semidefinite if the number of samples is lower than the number of securities. In this case the Cholesky decomposition might not be computable.[5] We can then apply regularization techniques (e. g. shrinkage, see in Sec. 4.2.1) to ensure positive definiteness. Otherwise if suitable linear returns data

[5] The Cholesky decomposition is possible for positive semidefinite matrices, but software implementations might need positive definiteness depending on the algorithm used (e. g. Python numpy).

(see in Sec. 3) is available, then a centered and normalized data matrix can also serve as matrix $\mathbf{G}$.

Using the decomposition we can write the portfolio variance as $\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{G}\mathbf{G}^\mathsf{T}\mathbf{x} = \|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2^2$. This leads to two different conic forms (see also in Sec. 13.1.1).

1. Modeling with rotated quadratic cone:

   We can directly model the squared norm constraint $\|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2^2 \leq \gamma^2$ using the rotated quadratic cone as $(\gamma^2, \frac{1}{2}, \mathbf{G}^\mathsf{T}\mathbf{x}) \in \mathcal{Q}_\mathrm{r}^{k+2}$. This will give us the conic equivalent of e. g. problem (2.3):

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & (\gamma^2, \frac{1}{2}, \mathbf{G}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{k+2}, \\ & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.7}$$

2. Modeling with quadratic cone:

   We can also consider the equivalent norm constraint $\|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2 \leq \gamma$ instead of the squared norm, and model it using the quadratic cone as $(\gamma, \mathbf{G}^\mathsf{T}\mathbf{x}) \in \mathcal{Q}^{k+1}$. The conic equivalent of problem (2.3) will then look like

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & (\gamma, \mathbf{G}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}^{k+1}, \\ & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.8}$$

   This way we can also model problem (2.4). We introduce the variable s to represent the upper bound of the portfolio standard deviation, and model the constraint $\|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2 \leq s$ using the quadratic cone as

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \tilde{\delta}s & & \\ \text{subject to} & (s, \mathbf{G}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}^{k+1}, \\ & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1. \end{array} \tag{2.9}$$

   While modeling problem (2.3) using the rotated quadratic cone might seem more natural, it can also be reasonable to model it using the quadratic cone for the benefits of the smaller scale and intuitive interpretation of $\tilde{\delta}$. This is the same as modeling of problem (2.4), which shows that conic optimization can also provide efficient solution to problems that are inefficient to solve in their original form.

In general, transforming the optimization problem into the conic form has multiple practical advantages. Solving the problem in this format will result in a more robust and usually faster and more reliable solution process. Check Sec. 13.1.2 for details.

2.4 Example

We have seen how to transform a portfolio optimization problem into conic form. Now we will present a detailed example in **MOSEK** Fusion. Assume that the input data estimates $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given. The latter is also positive definite. For methods and examples on how to obtain these see Sec. 3.

Suppose we would like to create a long only portfolio of eight stocks. Assume that we receive the following variables:

```
# Expected returns and covariance matrix
m = np.array(
    [0.0720, 0.1552, 0.1754, 0.0898, 0.4290, 0.3929, 0.3217, 0.1838]
)
S = np.array([
    [0.0946, 0.0374, 0.0349, 0.0348, 0.0542, 0.0368, 0.0321, 0.0327],
    [0.0374, 0.0775, 0.0387, 0.0367, 0.0382, 0.0363, 0.0356, 0.0342],
    [0.0349, 0.0387, 0.0624, 0.0336, 0.0395, 0.0369, 0.0338, 0.0243],
    [0.0348, 0.0367, 0.0336, 0.0682, 0.0402, 0.0335, 0.0436, 0.0371],
    [0.0542, 0.0382, 0.0395, 0.0402, 0.1724, 0.0789, 0.0700, 0.0501],
    [0.0368, 0.0363, 0.0369, 0.0335, 0.0789, 0.0909, 0.0536, 0.0449],
    [0.0321, 0.0356, 0.0338, 0.0436, 0.0700, 0.0536, 0.0965, 0.0442],
    [0.0327, 0.0342, 0.0243, 0.0371, 0.0501, 0.0449, 0.0442, 0.0816]
])
```

The similar magnitude and positivity of all the covariances suggests that the selected securities are closely related. In Sec. 3.4.2 we will see that they are indeed from the same market and that the data was collected from a highly bullish time period, resulting in the large expected returns. Later in Sec. 5.5.1 we will analyze this covariance matrix more thoroughly.

2.4.1 Maximizing return

We will solve problem (2.2) with an additional constraint to prevent short selling. We specify a risk level of $\gamma^2 = 0.05$ and assume that there are no transaction costs. The optimization problem will then be

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^T\mathbf{x} & & \\ \text{subject to} & \mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{1}^T\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{2.10}$$

Recall that by the Cholesky decomposition we have $\boldsymbol{\Sigma} = \mathbf{G}\mathbf{G}^T$. Then we can model the quadratic term $\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x} \leq \gamma^2$ using the rotated quadratic cone as $(\gamma^2, \frac{1}{2}, \mathbf{G}^T\mathbf{x}) \in \mathcal{Q}_r^{N+2}$, and arrive at

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^T\mathbf{x} & & \\ \text{subject to} & \left(\gamma^2, \frac{1}{2}, \mathbf{G}^T\mathbf{x}\right) & \in & \mathcal{Q}_r^{N+2}, \\ & \mathbf{1}^T\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{2.11}$$

In the code, $\mathbf{G}$ will be an input parameter, so we compute it first. We use the Python package `numpy` for this purpose, abbreviated here as `np`.

```
N = m.shape[0]  # Number of securities
gamma2 = 0.05  # Risk limit

# Cholesky factor of S to use in conic risk constraint
G = np.linalg.cholesky(S)
```

Next we define the Fusion model for problem (2.11):

```
with Model("MarkowitzReturn") as M:
     # Decision variable (fraction of holdings in each security)
     # The variable x is the fraction of holdings in each security.
     # x must be positive, this imposes the no short-selling constraint.
     x = M.variable("x", N, Domain.greaterThan(0.0))

     # Budget constraint
     M.constraint('budget', Expr.sum(x) == 1)

     # Objective
     M.objective('obj', ObjectiveSense.Maximize, x.T @ m)

     # Imposes a bound on the risk
     M.constraint('risk', Expr.vstack(gamma2, 0.5, G.T @ x),
                          Domain.inRotatedQCone())

     # Solve optimization
     M.solve()

     returns = M.primalObjValue()
     portfolio = x.level()
```

The resulting portfolio return is $\boldsymbol{\mu}^T\mathbf{x} = 0.2767$ and the portfolio composition is $\mathbf{x} = [0, 0.0913, 0.2691, 0, 0.0253, 0.3216, 0.1765, 0.1162]$.

2.4.2 Efficient frontier

In this section we compute the full efficient frontier using the same input variables. This is the easiest to do based on the formulation (2.3) by varying the delta parameter. We also prevent short selling in this case and assume no transaction costs. The optimization problem becomes

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^T\mathbf{x} - \tilde{\delta}\sqrt{\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x}} & & \\ \text{subject to} & \mathbf{1}^T\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{2.12}$$

With the help of Cholesky decomposition, we can model the quadratic term $\sqrt{\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x}} = \sqrt{\mathbf{x}^T\mathbf{G}\mathbf{G}^T\mathbf{x}} = \|\mathbf{G}^T\mathbf{x}\|_2$ in the objective as a conic constraint. Let us introduce the variable s and apply that $\|\mathbf{G}^T\mathbf{x}\|_2 \leq s$ is equivalent to the quadratic cone membership $(s, \mathbf{G}^T\mathbf{x}) \in \mathcal{Q}^{N+1}$. Then the problem will take the form

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^T\mathbf{x} - \tilde{\delta}s & & \\ \text{subject to} & \mathbf{1}^T\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0, \\ & (s, \mathbf{G}^T\mathbf{x}) & \in & \mathcal{Q}^{N+1}. \end{array} \tag{2.13}$$

First we compute the input variable $\mathbf{G}$:

```
N = m.shape[0]  # Number of securities
```

(continues on next page)

(continued from previous page)

```
# Cholesky factor of S to use in conic risk constraint
G = np.linalg.cholesky(S)
```

Next we define the Fusion model for problem (2.13):

```
with Model("MarkowitzFrontier") as M:
    # Variables
    # The variable x is the fraction of holdings in each security.
    # x must be positive, this imposes the no short-selling constraint.
    x = M.variable("x", N, Domain.greaterThan(0.0))

    # The variable s models the portfolio variance term in the objective.
    s = M.variable("s", 1, Domain.unbounded())

    # Budget constraint
    M.constraint('budget', Expr.sum(x) == 1)

    # Objective (quadratic utility version)
    delta = M.parameter()
    M.objective('obj', ObjectiveSense.Maximize,
                       x.T @ m - delta * s)

    # Conic constraint for the portfolio variance
    M.constraint('risk', Expr.vstack(s, G.T @ x), Domain.inQCone())
```

Here we use a feature in Fusion that allows us to define model parameters. The line `delta = M.parameter()` in the code does this. Defining a parameter is ideal for the computation of the efficient frontier, because it allows us to reuse an already existing optimization model, by only changing its parameter value.

To do so, we define the range of values for the risk aversion parameter δ to be 20 numbers between 10^{-1} and $10^{1.5}$:

```
deltas = np.logspace(start=-1, stop=1.5, num=20)[::-1]
```

Then we will solve the optimization model repeatedly in a loop, setting a new value for the parameter δ in each iteration, without having to redefine the whole model each time.

```
columns = ["delta", "obj", "return", "risk"] + df_prices.columns.tolist()
df_result = pd.DataFrame(columns=columns)
for d in deltas:
    # Update parameter
    delta.setValue(d)

    # Solve optimization
    M.solve()

    # Save results
    portfolio_return = m @ x.level()
    portfolio_risk = s.level()[0]
```

(continues on next page)

(continued from previous page)

```
    row = pd.Series([d, M.primalObjValue(), portfolio_return,
                     portfolio_risk] + list(x.level()), index=columns)
    df_result = df_result.append(row, ignore_index=True)
```

Then we can plot the results. See the risk-return plane on Fig. 2.1. We generate it by the following code:

```
# Efficient frontier
df_result.plot(x="risk", y="return", style="-o",
               xlabel="portfolio risk (std. dev.)",
               ylabel="portfolio return", grid=True)
```

We can observe the portfolio composition for different risk-aversion levels on Fig. 2.2, generated by

```
# Portfolio composition
my_cmap = LinearSegmentedColormap.from_list("non-extreme gray",
              ["#111111", "#eeeeee"], N=256, gamma=1.0)
df_result.set_index('risk').iloc[:, 3:].plot.area(colormap=my_cmap,
              xlabel='portfolio risk (std. dev.)', ylabel="x")
```

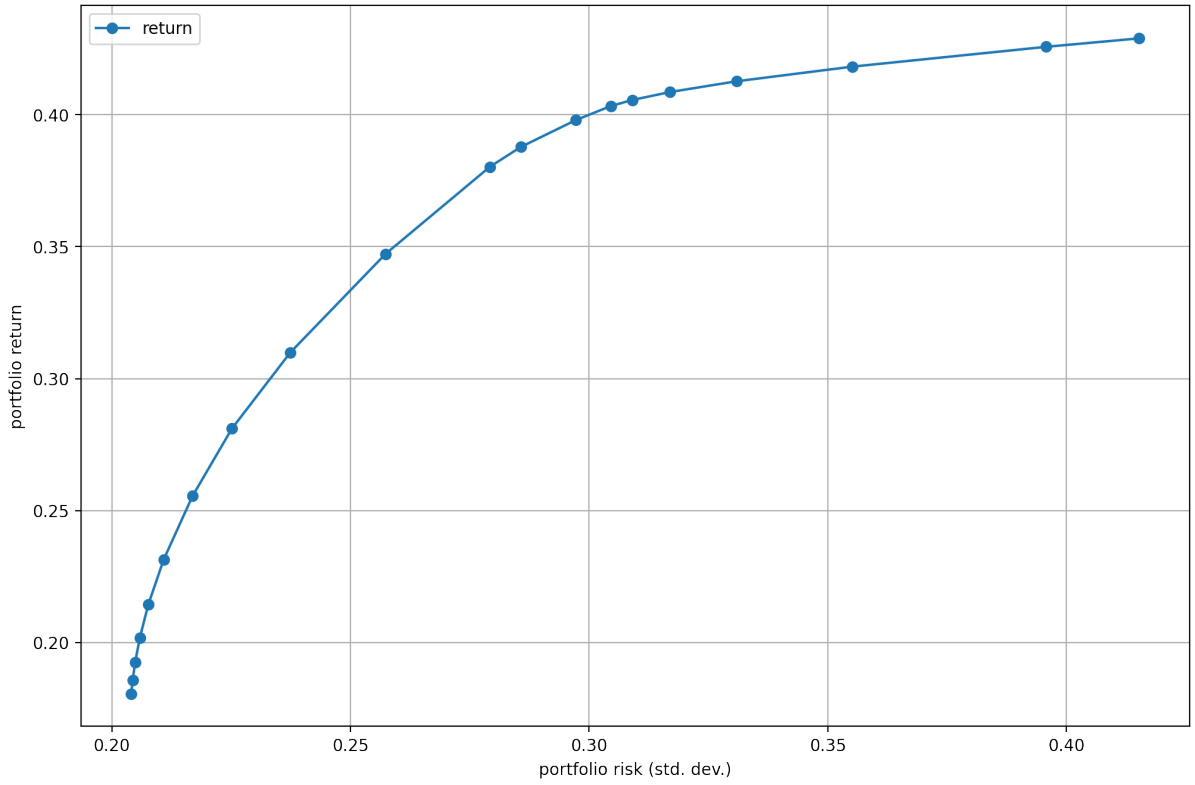

Fig. 2.1: The efficient frontier.

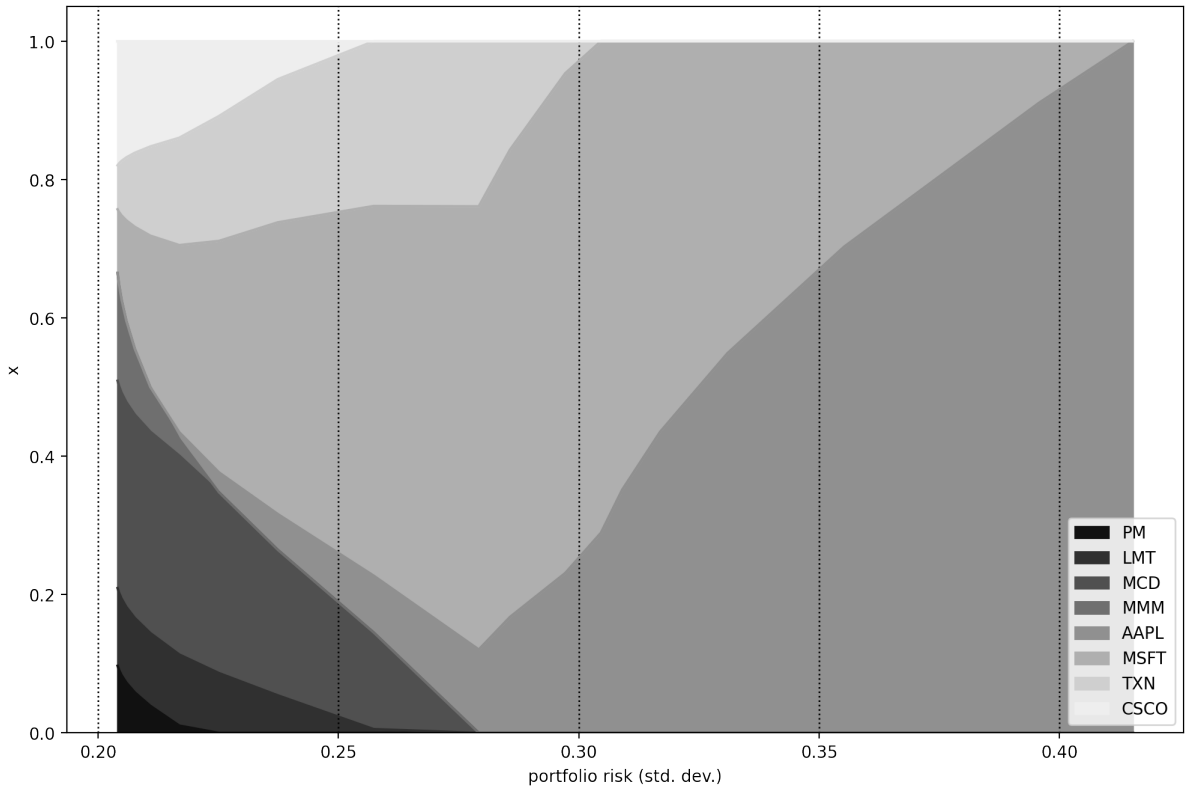

Fig. 2.2: Portfolio composition $\mathbf{x}$ with varying level if risk-aversion δ.

Chapter 3

Input data preparation

In Sec. 2.1.1 we mentioned that the expected return vector μ and the covariance matrix Σ of the securities needs to be estimated. In this chapter we will discuss this topic through a more general approach.

Portfolio optimization problems are inherently stochastic because they contain uncertain forward-looking quantities, most often the return of securities R. In the case of the basic mean–variance optimization problems (2.1)-(2.3) the simple form of the objective and constraint functions make it possible to isolate uncertainty from the decision variables and wrap it into the parameters. This allows us to separate stochastic optimization into parameter estimation and deterministic optimization steps.

In many cases, for instance those involving different risk measures, however, it is not possible to estimate the optimization inputs as a separate step. These objectives or constraints can be a non-trivial function of portfolio returns, and thus we have to explicitly model the randomness. See examples in Sec. 8. Therefore more generally our goal is to estimate the *distribution of returns* over the investment time period. The simplest way to do that is by using the concept of *scenario*.

3.1 Scenarios

A scenario is a possible realization $\mathbf{z}_k$ of the N dimensional random vector Z of a quantity corresponding to a given time period [MWOS15]. Thus we can also see a set of scenarios as a discretization of the distribution of Z.

We denote the number of scenarios by T. The probability of scenario $\mathbf{z}_k$ occurring will be p_k, with $\sum_{k=1}^{T} p_k = 1$. In practice when the scenarios come from historical data or are obtained via computer simulation, all scenarios are assumed to have the same probability $1/T$.

We commonly work with scenarios of the security returns R. Scenario k of security returns will be r_k. Assuming that $p_k = 1/T$, we can arrange the scenarios as columns of the $N \times T$ matrix $\mathbf{R}$.

Models involving certain risk measures can most easily be solved as convex optimization problems using scenarios (see Sec. 8). This approach is called *sample average approximation* in stochastic optimization.

3.1.1 Scenario generation

Both the quality and the quantity of scenarios is important to get reliable optimal portfolios. Scenarios must be representative and also there must be enough of them to accurately model a random variable. Too few scenarios could lead to approximation errors, while too many scenarios could result in excessive computational cost. Next we discuss some common ways of generating scenarios.

3.1.2 Historical data

If we can assume that past realizations of a random quantity represents possible future outcomes, we can simply use historical data to create scenarios. Observations are then collected from a predetermined time window, with a given frequency. Each scenario would correspond to a vector of simultaneous observations for all N securities.

Using historical data is a simple non-parametric approach, but it could be inaccurate or insufficient in some cases:

- The underlying assumption about past realizations representing the future might not be realistic, if markets fundamentally change in the observed time period, or unexpected extreme events happen. Such changes can happen at least in every several years, often making the collection of enough representative historical data impossible.
- It can represent the tails of the distribution inaccurately because of low number of samples corresponding to the low probability tail events. Also these extreme values can highly depend on the given historical sample, and can fluctuate a lot when the sample changes.
- It depends on the observations being independent and identically distributed. Without this assumption, for example if the data exhibits autocorrelation, volatility clustering, etc., the distribution estimated based on historical scenarios will be inaccurate.

3.1.3 Monte Carlo simulation

With this approach we can generate a large number of scenarios according to a specified distribution. It can be useful in cases when there is a lack of historical scenarios available in the desired timeframe, but we have a model of the relevant probability distribution. This allows us to generate scenarios through a parametric approach. It has the following steps:

1. Select a (multivariate) probability distribution that explains all interesting features of the quantity of interest, e. g. fat tails and tail-dependence of return.
2. Estimate the distribution parameters using historical (independent and identically distributed) data samples.
3. Generate scenarios by sampling from the fitted distribution.

Later we will see an example of Monte Carlo scenario generation in Sec. 5.5.1.

3.1.4 Performance assessment

We advise to have two separate sets of scenario data.

- The first one is the *in-sample data*, which we use for running the optimization and building the portfolio. We assume it known at the time of portfolio construction.
- The other data set is the *out-of-sample data*, which is for assessing the performance of the optimal portfolio. Out-of sample data is assumed to become available after the portfolio has been built.

3.2 Modeling the distribution of returns

In this section we describe how to estimate the distribution of security return over the time period of investment. It can be in the form of the estimated expected return and covariance matrix of securities, or in the form of a set of return scenarios.

The simplest case is when we have sufficient amount of return data for the desired time period, then we can estimate their distribution easily and proceed with the optimization. However, such data is seldom available, especially if the time period is very long, for reasons mentioned in Sec. 3.1.2. Therefore often we need to use higher frequency (shorter timeframe) data to estimate the distribution, then project the distribution to the longer timeframe of interest. This is typically not straightforward to do. We will describe this procedure first for stocks, then also outline the general method for other securities.

3.2.1 Notions of return

There are two different notions of return. Let P_{t_0} and P_{t_1} be the prices of a security at the beginning and end of a time period.

Linear return

Also called *simple return*, calculated as

$$R^{\text{lin}} = \frac{P_{t_1}}{P_{t_0}} - 1.$$

This return definition is the one we used so far in this book, the return whose distribution we would like to compute over the time period of investment.

We can aggregate linear returns across securities, meaning that the linear return of a portfolio is the average of the linear returns of its components:

$$R^{\text{lin}}_{\mathbf{x}} = \sum_i x_i R^{\text{lin}}_i. \tag{3.1}$$

This is a property we need to be able to validly compute portfolio return and portfolio risk. However, it is not easy to scale linear return from one time period to another, for example to compute monthly return from daily return.

Logarithmic return

Also called *continuously compounded return*, calculated as

$$R^{\log} = \log\left(\frac{P_{t_1}}{P_{t_0}}\right).$$

We can aggregate logarithmic returns across time, meaning that the total logarithmic return over K time periods is the sum of all K single-period logarithmic returns: $R_K^{\log} = \log\left(P_{t_K}/P_{t_0}\right) = \sum_{k=1}^{K} \log\left(P_{t_k}/P_{t_{k-1}}\right)$. This property makes it very easy to scale logarithmic return from one time period to another. However, we cannot average logarithmic returns the same way as linear returns, therefore they are unsuitable for computation of portfolio return and portfolio risk.

Relationship between returns

The two notions of return are related by

$$R^{\log} = \log\left(1 + R^{\text{lin}}\right). \tag{3.2}$$

We must not confuse them, especially on longer investment time periods [Meu10a]. Logarithmic returns are suitable for projecting from shorter to longer time periods, while linear returns are suitable for computing portfolio level quantities.

3.2.2 Data preparation for stocks

Because both returns have properties necessary for portfolio optimization, the data preparation procedure for a stock portfolio is the following:

1. We start with logarithmic return data over the time period of estimation (e. g. one day if the data has daily frequency).

2. We estimate the distribution of logarithmic returns on this time period. If we assume that logarithmic returns are normally distributed, then here we estimate their mean vector and covariance matrix.

3. Using the property of aggregation across time we scale this distribution to the time period of investment (e. g. one year). We can scale the mean and covariance of logarithmic returns by the "square-root rule"[1]. This means that if h is the time period of investment and τ is the time period of estimation, then

$$\boldsymbol{\mu}_h = \frac{h}{\tau}\boldsymbol{\mu}_\tau, \quad \boldsymbol{\Sigma}_h = \frac{h}{\tau}\boldsymbol{\Sigma}_\tau.$$

4. Finally, we convert it into a distribution of linear returns over the period of investment. Then we can use the property of cross-sectional aggregation to compute portfolio return and portfolio risk. We show an example working with the assumption of normal distribution for logarithmic returns in Sec. 3.4.

[1] Originally the square-root rule for stocks states that we can annualize the standard deviation σ_τ of τ-day logarithmic returns by $\sigma_{\text{ann}} = \sigma_\tau\sqrt{252/\tau}$, where σ_{ann} is the annualized standard deviation, and 252 is the number of business days in a year.

3.2.3 Data preparation in general

We can generalize the above procedure to other types of securities; see in detail in [Meu05]. The role of the logarithmic return in case of stocks is generalized by the concept of *market invariant*. A market invariant is a quantity that determines the security price and is repeating (approximately) identically across time. Thus we can model it as a sequence of independent and identically distributed random variables. The steps will be the following:

1. Identify the market invariants. As we have seen, the invariant for the stock market is the logarithmic return. However, for other markets it is not necessarily the return, e. g. for the fixed-income market the invariant is the change in yield to maturity.

2. Estimate the joint distribution of the market invariant over the time period of estimation. Apart from historical data on the invariants, any additional information can be used. We can use parametric assumptions, approximate through moments, apply factor models, shrinkage estimators, etc., or simply use the empirical distribution, i. e., a set of scenarios.

3. Project the distribution of invariants to the time period of investment. Here we will assume that the invariants are additive across time, i. e., the invariant corresponding to a larger time interval is the sum of invariants corresponding to smaller intervals (the property that logarithmic returns have). Then the projection is easy to do through the characteristic function (see in [Meu05]), or we can also scale the moments by applying the "square-root rule" (see in [Meu10b]).

4. Map the distribution of invariants into the distribution of security prices at the investment horizon through a pricing function. Then compute the distribution of linear returns from the distribution of prices. If we have a set of scenarios, we can simply apply the pricing function on each of them.

This process will be followed in the case studies section in an example.

3.3 Extensions

In this section we discuss some use cases which need a more general definition of the MVO problem.

3.3.1 Generalization of linear return

While it is common to use linear return in portfolio optimization, it is not the most general approach. In certain cases we need to extend its definition to be able to do model the optimization problem [Meu10c]. As we have seen in Sec. 3.2.1, the key property of linear return is the aggregation across securities (3.1). It allows us to define the expected portfolio return and the portfolio variance. We repeat the formula here in detail:

$$R_{\mathbf{x}}^{\text{lin}} = \frac{W_h - w_0}{w_0} = \sum_i \frac{v_i(P_{h,i} - p_{0,i})}{\mathbf{v}^\mathsf{T}\mathbf{p}_0} = \sum_i \frac{v_i p_{0,i}}{\sum_i v_i p_{0,i}} \frac{P_{h,i} - p_{0,i}}{p_{0,i}} = \sum_i x_i R_i^{\text{lin}}, \tag{3.3}$$

where $w_0 = \mathbf{v}^\mathsf{T}\mathbf{p}_0$ is the initial wealth, $W_h = \mathbf{v}^\mathsf{T}P_h$ is the final wealth, $\mathbf{v}$ is the vector of security holdings in terms of quantity, $\mathbf{p}_0$ is the vector of initial security prices, and P_h

is the vector of final security prices. We can see that maximizing linear return implicitly assumes that we wish to maximize final wealth.

Based on equation (3.3) there are two components in the aggregation formula that have to be well defined:

- Linear return of a security: It is given for security i as

$$R_i^{\text{lin}} = \frac{P_{h,i} - p_{0,i}}{p_{0,i}} \tag{3.4}$$

  Linear return is not well defined if the price of the security $p_{0,i}$ is zero. This can occur for example with swaps and futures.

- Portfolio weight: It is given for security i as

$$x_i = \frac{v_i p_{0,i}}{w_0}. \tag{3.5}$$

  Portfolio weight is not well defined if the initial wealth w_0 (the total portfolio value at time $t = 0$) is zero. This can occur for example in the case of dollar neutral long-short portfolios.

To be able to do portfolio optimization in the above mentioned special cases, we need to extend the definitions of portfolio weight and linear return such that we can apply the aggregation property (3.3).

- To extend the definition of linear return, we associate with each security a generalized normalizing quantity or *basis* b_i that takes the role of the security price $p_{0,i}$ in the denominator of formula (3.4):

$$R_i^{\text{lin}} = \frac{P_{h,i} - p_{0,i}}{b_i} \tag{3.6}$$

  The basis depends on the nature of the security; for swaps it can be the notional, for options it can be the strike or the underlying value, etc. It can be any value that results in an intuitive return definition to the portfolio manager. In most cases though it will be the price of the security, giving back the usual linear return definition (3.4).

- To extend the definition of the portfolio weight, we introduce a generalized *portfolio basis* b_{P} that takes the role of the initial portfolio value w_0 in the denominator of formula (3.5). We also use the relevant security basis b_i in the numerator:

$$x_i = \frac{v_i b_i}{b_{\text{P}}}. \tag{3.7}$$

In general, the basis quantities b_i $(i = 1, \ldots, N)$ and b_{P} have to satisfy the following properties:

1. They are positive. This guarantees that for a long position, a positive net profit (P&L) will correspond to a positive return.

2. They are measured in the same money units as the P&L. This ensures that the return is dimensionless.

3. They are homogeneous, i.e. if b is the basis for one unit of security, then the basis for n unit of securities will be nb. This ensures that the return is independent of the position size.

4. They are known at the beginning of the investment period (at time $t = 0$). This ensures that the return will correspond to the full investment period.

By defining a basis value for each security and a basis value for the portfolio, computing the expected portfolio return and portfolio variance will be possible through the generalized aggregation formula:

$$R_{\mathbf{x}}^{\text{lin}} = \sum_i \frac{v_i b_i}{b_{\text{P}}} \frac{P_{h,i} - p_{0,i}}{b_i} = \sum_i x_i R_i^{\text{lin}}, \tag{3.8}$$

We can observe that in formula (3.8), the security level basis values b_i cancel out. This means that the value of the security level basis does not affect the portfolio return. It only matters in the context of interpreting the security weight and security return.

3.3.2 Portfolio scaling

During the introduction of the MVO problem in Sec. 2 we interpreted x_i as the fraction of wealth invested into security i, such that the sum of weights is 1. This is, however, not the only possibility. From formula (3.8) we can see that the portfolio return and its interpretation or scaling is only affected by the portfolio level basis value b_{P}. We can select it independently of the security level basis values b_i ($i = 1, \ldots, N$), and thus change the scaling of the portfolio weights. Some natural choices:

- $b_{\text{P}} = \sum_i v_i b_i$. This choice ensures that the portfolio weights x_i ($i = 1, \ldots, N$) sum to 1, and thus represent fractions of b_{P}.
- $b_{\text{P}} = \sum_i v_i p_{0,i} = w_0$. In this case we normalize the P&L with the total portfolio value. For long only portfolio this is the same as the initial capital.
- $b_{\text{P}} = \sum_i |v_i| p_{0,i}$, the gross exposure. This provides an intuitive fraction interpretation to x_i for long-short portfolios as well.
- $b_{\text{P}} = C_{\text{init}}$, the initial capital.
- $b_{\text{P}} = 1$. In this case formula (3.8) will become the total net profit (P&L) in dollars, and the portfolio weights will also be measured in dollars.

In this book, we will use $b_{\text{P}} = \sum_i v_i b_i$ by default. Any other choice will be explicitly stated.

3.3.3 Long-short portfolios

When working with a long-short portfolio, we also have to extend the MVO problem slightly. If the investor can short sell and also use leverage (e. g. margin loan), then the total value of the investment (the gross exposure) can be greater than the amount of initial capital. In case of leverage, the gross exposure provides better insight into the risks taken than the total capital, so being fully invested is no longer meaningful as a sole constraint.

Therefore in the optimization problem formulation of typical long-short portfolios, we will substitute the budget constraint (2.5) to a gross exposure limit, i. e., a leverage constraint:

$$\sum_i |v_i| p_{0,i} \leq L \cdot C_{\text{init}},$$

where C_{init} is the initial invested capital, and L is the leverage limit. If we normalize here with $b_{\text{P}} = C_{\text{init}}$, then we get

$$\|\mathbf{x}\|_1 \leq L.$$

For example, Regulation-T states that for long positions the margin requirement is 50% of the position value, and for short positions the collateral requirement is 150% of the position value (of which 100% comes from short sale proceeds). This translates into $L = 2$. A special case of $L = 1$ would mean that we can short sell but have no leverage.

A more general version of the leverage constraint is

$$\sum_i m_i |x_i| \leq 1.$$

where m_i is the margin requirement of position i. In case of Reg-T we had $m_i = \frac{1}{2}$ for all i.

We might also impose an *enhanced active equity* structure on the portfolio, like 120/20 or 130/30. This is typically considered when it is possible to use the short sale proceeds to purchase more securities (another form of leverage), so there is no need to use margin loans. Such a portfolio has 100% market exposure, which is expressed by

$$\sum_i v_i p_{0,i} = C_{\text{init}},$$

or writing the same using $\mathbf{x}$ after normalizing again with $b_{\text{P}} = C_{\text{init}}$, we get the ususal budget constraint

$$\sum_i x_i = 1.$$

For example, 130/30 type portfolio would have the constraints $\|\mathbf{x}\|_1 \leq 1.6$ and $\mathbf{1}^{\mathsf{T}}\mathbf{x} = 1$.

We can also use *factor neutrality* constraints on a long-short portfolio. We can achieve this simply by adding a linear equality constraint expressing that the portfolio should be orthogonal to a vector β of factor exposures:

$$\beta^{\mathsf{T}}\mathbf{x} = 0.$$

A special case of this is when the factor exposures are all ones; then the portfolio will be *dollar neutral*:

$$\mathbf{1}^{\mathsf{T}}\mathbf{x} = 0.$$

We have to note that in the case of dollar neutrality, the total portfolio value is 0. Referring back to the discussion about portfolio weights in Sec. 3.3.1, in this case the portfolio weights cannot be defined as (3.5). Instead, we need to define a b_{P} that is different from w_0. See also in Sec. 3.3.2.

Note also that if we model using the quadratic cone instead of the rotated quadratic cone and $\mathbf{x} = \mathbf{0}$ is a feasible solution, then there will be no optimal portfolios which are not fully invested. The solutions will be either $\mathbf{x} = \mathbf{0}$ or some risky portfolio with $\|\mathbf{x}\|_1 = 1$. See a detailed discussion about this in Sec. 13.3.

3.4 Example

In this example we will show a case with real data, that demonstrates the steps in Sec. 3.2.3 and leads to the expected return vector and the covariance matrix we have seen in Sec. 2.4.

3.4.1 Problem statement

Suppose that we wish to invest in the U.S. stock market with a long-only strategy. We have chosen a set of eight stocks and we plan to re-invest any dividends. We start to invest at time $t = 0$ and the time period of investment will be $h = 1$ year, ending at time $t = 1$.

We also have a pre-existing allocation $\mathbf{v}_0$ measured in number of shares. We can express this allocation also in fractions of total dollar value $\mathbf{v}_0^\mathsf{T}\mathbf{p}_0$, by $\mathbf{x}_0 = \mathbf{v}_0 \circ \mathbf{p}_0 / \mathbf{v}_0^\mathsf{T}\mathbf{p}_0$.

Our goal is to derive a representation of the distribution of one year linear returns at time $t = 1$, and use it to solve the portfolio optimization problem. Here we will represent the distribution with the estimate $\boldsymbol{\mu}$ of the expected one year linear returns $\mu = \mathbb{E}(R)$ and with the estimate $\boldsymbol{\Sigma}$ of the covariance of one year linear returns $\Sigma = \mathrm{Cov}(R)$, at the time point $t = 1$.

3.4.2 Data collection

We obtain a time-series of daily stock prices from data providers, for a five year period (specifically from 2016-03-21 until 2021-03-18), adjusted for splits and dividends. During this time period the market was generally bullish, with only short downturn periods. This will be reflected in the estimated mean return vector and in the covariance structure. See a deeper analysis in Sec. 5.5.1.

We choose an estimation time period of one week as a balance between having enough independent observations and the homogeneity of the data series. Thus $\tau = 1/52$ year.

At this point we assume that the price data is available in the pandas DataFrame `df_prices`. The graphs of the price time-series can be seen on Fig. 3.1:

3.4.3 Estimation of moments

Now we will go over the steps in Sec. 3.2.3.

Market invariants

For stocks, both linear and logarithmic returns are market invariants, however it is easier to project logarithmic returns to longer time periods. It also has an approximately symmetrical distribution, which makes it easier to model. Therefore we resample each price time series to weekly frequency, and obtain a series of approximately 250 non-overlapping observations:

```
df_weekly_prices = df_prices.resample('W').last()
```

Next we compute weekly logarithmic returns from the weekly prices, followed by basic handling of missing values:

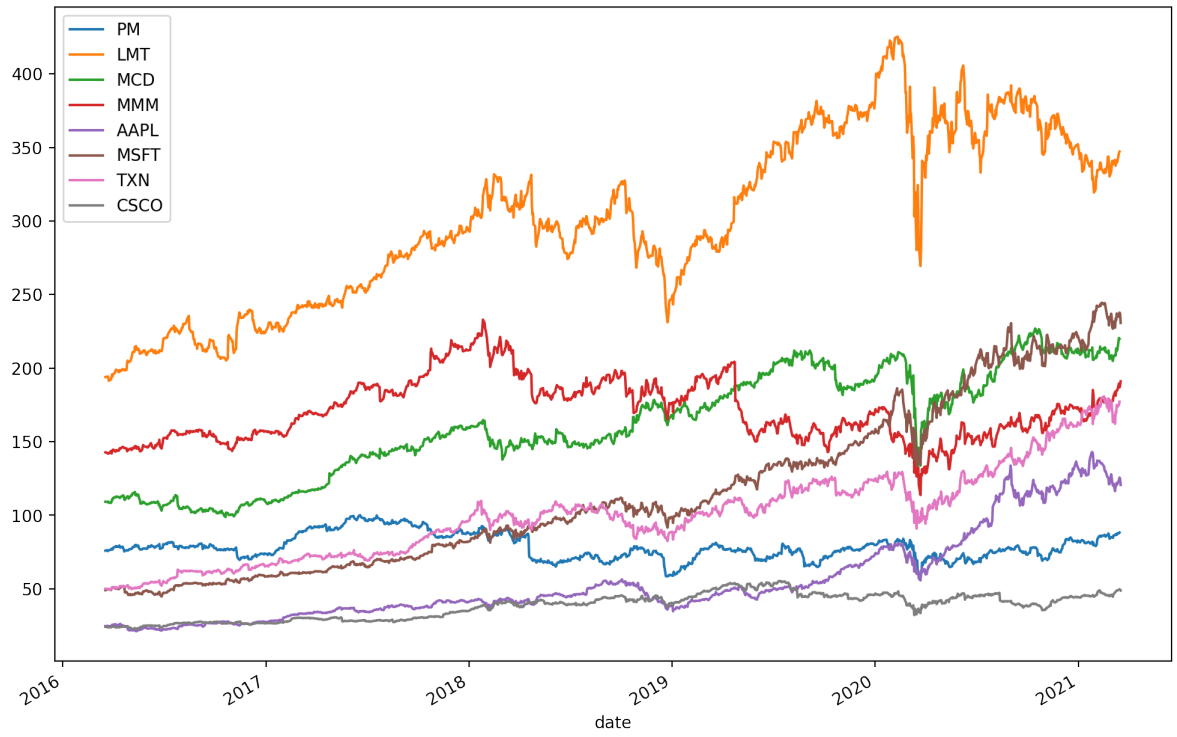

Fig. 3.1: Daily prices of the 8 stocks in the example portfolio.

```
df_weekly_log_returns = \
    np.log(df_weekly_prices) - np.log(df_weekly_prices.shift(1))
df_weekly_log_returns = df_weekly_log_returns.dropna(how='all')
df_weekly_log_returns = df_weekly_log_returns.fillna(0)
```

Distribution of invariants

To estimate the distribution of market invariants, in this example we choose a parametric approach and fit the weekly logarithmic returns to a multivariate normal distribution. This requires the estimation of the distribution parameters $\mu_\tau^{\log}$ and $\Sigma_\tau^{\log}$. For simplicity we use the sample statistics denoted by $\boldsymbol{\mu}_\tau^{\log}$ and $\boldsymbol{\Sigma}_\tau^{\log}$. The "log" superscript indicates that these statistics correspond to the logarithmic returns. In code this looks like

```
return_array = df_weekly_log_returns.to_numpy()
m_weekly_log = np.mean(return_array, axis=0)
S_weekly_log = np.cov(return_array.transpose())
```

Projection of invariants

We project the distribution of the weekly logarithmic returns represented by $\boldsymbol{\mu}_\tau^{\log}$ and $\boldsymbol{\Sigma}_\tau^{\log}$ to the one year investment horizon. Because the logarithmic returns are additive across time, the projected distribution will also be normal with parameters $\boldsymbol{\mu}_h^{\log} = \frac{h}{\tau}\boldsymbol{\mu}_\tau^{\log}$ and $\boldsymbol{\Sigma}_h^{\log} = \frac{h}{\tau}\boldsymbol{\Sigma}_\tau^{\log}$.

```
m_log = 52 * m_weekly_log
S_log = 52 * S_weekly_log
```

Distribution of linear returns

To obtain the distribution of linear returns at the investment horizon h, we first derive the distribution of security prices at the investment horizon. Using the characteristic function of the normal distribution, and the pricing function $P_h = \mathbf{p}_0 \exp\left(R^{\log}\right)$, we get

$$\begin{aligned} \mathbb{E}(P_h) &= \mathbf{p}_0 \circ \exp\left(\boldsymbol{\mu}_h^{\log} + \tfrac{1}{2}\operatorname{diag}(\boldsymbol{\Sigma}_h^{\log})\right), \\ \operatorname{Cov}(P_h) &= \mathbb{E}(P_h)\mathbb{E}(P_h)^\mathsf{T} \circ (\exp(\boldsymbol{\Sigma}_h^{\log}) - 1). \end{aligned} \tag{3.9}$$

In code, this will look like

```
p_0 = df_weekly_prices.iloc[0].to_numpy()
m_P = p_0 * np.exp(m_log + 1/2*np.diag(S_log))
S_P = np.outer(m_P, m_P) * (np.exp(S_log) - 1)
```

Then the estimated moments of the linear return is easy to get by $\boldsymbol{\mu} = \frac{1}{\mathbf{p}_0} \circ \mathbb{E}(P_h) - \mathbf{1}$ and $\boldsymbol{\Sigma} = \frac{1}{\mathbf{p}_0\mathbf{p}_0^\mathsf{T}} \circ \operatorname{Cov}(P_h)$, where $\circ$ denotes the elementwise product and division by a vector or a matrix is also done elementwise.

```
m = 1 / p_0 * m_P - 1
S = 1 / np.outer(p_0, p_0) * S_P
```

Notice that we could have computed the distribution of linear returns from the distribution of logarithmic returns directly is this case, using the simple relationship (3.2). However in general for different securities, especially for derivatives we derive the distribution of linear returns from the distribution of prices. This case study is designed to demonstrate the general procedure. See details in [Meu05, Meu11].

Also in particular for stocks we could have started with linear returns as market invariants, and model their distribution. Projecting it to the investment horizon, however, would have been much more complicated. There exist no scaling formulas for the linear return distribution or its moments as simple as the ones for logarithmic returns.

Chapter 4

Dealing with estimation error

As we have seen in Sec. 2, Markowitz portfolio optimization is a convenient and useful framework. However, in practice there is one important limitation [MM08]. The optimal solution is extremely sensitive to changes in the vector $\boldsymbol{\mu}$ of estimated expected returns and the estimated covariance matrix $\boldsymbol{\Sigma}$. Small changes in these input parameters often result in large changes in the optimal portfolio. The optimization basically magnifies the impact of estimation errors, yielding unintuitive, questionable portfolios and extremely small or large positions, which are difficult to implement. The instability causes unwarranted turnover and transaction costs.

It would help to use a larger estimation window, but the necessary amount of data grows unrealistically high quickly, because the size of the covariance matrix increases quadratically with the number of securities N [Bra10, CY16].

Fortunately there are multiple approaches to mitigating this issue. These can significantly reduce the impact of estimation errors and stabilize the optimization.

In the following we explore the main aspects of overcoming the estimation error issues.

4.1 Estimation of mean

The optimization is particularly sensitive to estimation errors in expected returns, small variations in sample means can lead to sizeable portfolio readjustments. Estimation based only on historical time-series can result in poor out-of-sample portfolio performance. Sample mean is hard to estimate accurately from historical data, because for all unbiased estimators the standard error fundamentally depends on the length (duration) of the time-series [Dod12]. Therefore to get better accuracy it is necessary to extend the time horizon, which is often problematic because data from a longer time interval might contain more structural changes, i. e. lack stationarity in its underlying parameters.

Despite these difficulties, the below methods offer an improvement on mean estimation.

4.1.1 Black-Litterman model

One way to deal with estimation error is to combine noisy data with prior information. We can base prior beliefs on recent regulatory, political and socioeconomic events, macroeconomy news, financial statements, analyst ratings, asset pricing theories, insights about market dynamics, econometric forecasting methods, etc. [AZ10]

The Black-Litterman model considers the market equilibrium returns implied by CAPM as a prior estimate, then allows the investor to update this information with their own views on security returns. This way we do not need noisy historical data for estimation [Bra10, CJPT18].

Thus we assume the distribution of expected return μ to be $\mathcal{N}(\boldsymbol{\mu}_{\text{eq}}, \mathbf{Q})$, where $\boldsymbol{\mu}_{\text{eq}}$ is the equilibrium return vector, and the covariance matrix $\mathbf{Q}$ represents the investor's confidence in the equilibrium return vector as a realistic estimate.

The investor's views are then modeled by the equation $\mathbf{B}\mu = \mathbf{q}+\varepsilon$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$. Each row of this equation represents a view in the form of a linear equation, allowing for both absolute and relative statements on one or more securities. The views are also uncertain, we can use the diagonal of $\mathbf{V}$ to set the confidence in them, small variance meaning strong confidence in a view.

By combining the prior with the views using Bayes's theorem we arrive at the estimate

$$\boldsymbol{\mu}_{\text{BL}} = \mathbf{Mq} + (\mathbf{I} - \mathbf{MB})\pi, \tag{4.1}$$

where $\mathbf{M} = \mathbf{QB}^{\mathsf{T}}(\mathbf{BQB}^{\mathsf{T}} + \mathbf{V})^{-1}$.

The prior $\boldsymbol{\mu}_{\text{eq}}$ is implied by the solution $\mathbf{x} = \frac{1}{\delta}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ of problem (2.3), by assuming that $\mathbf{x} = \mathbf{x}_{\text{mkt}}$, i. e., weights according to market capitalization.

We can also understand the Black–Litterman model as an application of generalized least-squares to combine two information sources (e. g. historical data and exogeneous views) [Bra10, The71]. Assuming the sources are independent, we write them into one linear equation with block structure as

$$\begin{pmatrix} \pi \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} \mathbf{I} \\ \mathbf{B} \end{pmatrix} \mu + \begin{pmatrix} \varepsilon_\pi \\ \varepsilon_{\mathbf{q}} \end{pmatrix}, \tag{4.2}$$

where π is the prior, $\varepsilon_\pi \sim \mathcal{N}(0, \mathbf{Q})$, $\varepsilon_{\mathbf{q}} \sim \mathcal{N}(0, \mathbf{V})$, $\mathbf{Q}$ and $\mathbf{V}$ nonsingular. The solution will be

$$\boldsymbol{\mu}_{\text{GLS}} = (\mathbf{Q}^{-1} + \mathbf{B}^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{B})^{-1}(\mathbf{Q}^{-1}\pi + \mathbf{B}^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{q}). \tag{4.3}$$

Substituting $\pi = \boldsymbol{\mu}_{\text{eq}}$ we get a different form of $\boldsymbol{\mu}_{\text{BL}}$.[1]

Because the Black-Litterman model gives an expected return vector relative to the market equilibrium return, the optimal portfolio will also be close to the market portfolio, meaning it should not contain extreme positions. A drawback of this method is that the exogeneous views are assumed independent of historical data, which can be difficult to satisfy in practice. The model is extended further in [BGP12] to views on volatility and market dynamics.

[1] We can derive this using $(\mathbf{A}+\mathbf{B})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{A}+\mathbf{B})^{-1}$ for nonsingular matrices.

4.1.2 Shrinkage estimation

Sample estimators are unbiased, but perform well only in the limit case of an infinite number of observations. For small samples their variance is large. In contrast, constant estimators have no variance, but display large bias. *Shrinkage estimators* improve the sample estimator by adjusting (shrinking) the sample estimator towards a constant estimator, the *shrinkage target*, basically finding the optimal tradeoff between variance and bias.

Shrinkage can also be thought of as regularization or a form of empirical Bayes estimation. Also the shrinkage target has to be consistent with other prior information and optimization constraints.

1. **James–Stein estimator**:

   This is the most widely known shrinkage estimator for the estimation of mean vector of a normal random variable with known covariance [EM76, LC98, Ric99]. The sample mean $\boldsymbol{\mu}_{\text{sam}}$ is normally distributed with mean μ and covariance Σ/T, where Σ is the covariance matrix of the data. Assuming $\mathbf{b}$ as target, the James–Stein estimator will be

   $$\boldsymbol{\mu}_{\text{JS}} = \mathbf{b} + (1-\alpha)(\boldsymbol{\mu}_{\text{sam}} - \mathbf{b}),$$

   where the *shrinkage coefficient* is

   $$\alpha = \frac{1}{T}\frac{\tilde{N}-2}{(\boldsymbol{\mu}_{\text{sam}} - \mathbf{b})^{\mathsf{T}}\Sigma^{-1}(\boldsymbol{\mu}_{\text{sam}} - \mathbf{b})},$$

   and $\tilde{N} = \text{Tr}(\Sigma)/\lambda_{\max}(\Sigma)$ is the effective dimension. The estimator improves $\boldsymbol{\mu}_{\text{sam}}$ for $\tilde{N} > 2$, i. e., when Σ/T has not too different eigenvalues. Otherwise an estimator with coordinatewise different shrinkage coefficient is preferable, such as the one in [EM75].

   The target $\mathbf{b}$ is arbitrary, but it is common to set it as the mean of means $\mathbf{1}^{\mathsf{T}}\boldsymbol{\mu}_{\text{sam}}/N$. This depends on the data, making the number of parameters to estimate one less, so the dimension $\tilde{N}$ should be adjusted accordingly.

   We can further improve $\boldsymbol{\mu}_{\text{JS}}$ by the positive-part rule, when we set $(1-\alpha)$ to zero whenever it would be negative.

2. **Jorion's estimator** [Jor86]:

   This estimator was derived in the context of portfolio analysis through an empirical Bayes approach. It uses the target $b\mathbf{1}$, where b is the volatility-weighted mean of means:

   $$b = \frac{\mathbf{1}^{\mathsf{T}}\Sigma^{-1}\boldsymbol{\mu}_{\text{sam}}}{\mathbf{1}^{\mathsf{T}}\Sigma^{-1}\mathbf{1}}.$$

   The shrinkage coefficient in this case is

   $$\alpha = \frac{N+2}{N+2+T(\boldsymbol{\mu}_{\text{sam}} - b\mathbf{1})^{\mathsf{T}}\Sigma^{-1}(\boldsymbol{\mu}_{\text{sam}} - b\mathbf{1})}.$$

   If Σ is not known, it can be replaced by $\frac{T-1}{T-N-2}\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is the sample covariance matrix. The assumption of normality is not critical, but it gives more improvement over $\boldsymbol{\mu}_{\text{sam}}$ in "symmetric" situations, where variances are similar across data series.

4.2 Estimation of the covariance matrix

In contrast to the estimation of the mean, the standard error of the sample variance depends only on the sample size, making it possible to estimate from higher frequency data. Still, if the number of securities is high relative to the number of samples, the optimal solution can suffer from two issues [MM08]:

- Ill conditioning: We need to have sufficient amount of data to get a well-conditioned covariance estimate. A rule of thumb is that the number of observations T should be an order of magnitude greater than the number of securities N.
- Estimation error: The estimation error of the covariance matrix increases quadratically (in contrast to linear increase in the case of the mean), so if N/T is large it can quickly become the dominant contributing factor to loss of accuracy.

When N/T is too large, the following regularization methods can offer an improvement over the sample covariance matrix [CM13]. Moreover, we can also apply factor models in this context, we discuss this in more detail in Sec. 5.

4.2.1 Covariance shrinkage

The idea is the same as in Sec. 4.1.2, we try to achieve a balanced bias–variance tradeoff by shrinking the sample covariance matrix towards a shrinkage target. We can choose the latter in many ways, but it is important to be consistent with other prior information, e. g. what is implied by constraints.

The most well known shrinkage estimator is by Ledoit and Wolf; see [LW03a, LW03b, LW04] and [CM13] for a quick review. It combines the sample covariance matrix $\mathbf{\Sigma}_{\text{sam}}$ with the target $\mathbf{B}$:

$$\mathbf{\Sigma}_{\text{LW}} = (1-\alpha)\mathbf{\Sigma}_{\text{sam}} + \alpha\mathbf{B}. \tag{4.4}$$

We have to estimate the shrinkage intensity parameter α depending on the target $\mathbf{B}$. In practice it is also truncated so that $\alpha \in [0,1]$. Ledoit and Wolf propose three type of targets:

- Multiple of the identity matrix:

$$\mathbf{B}_{\text{I}} = \bar{s}^2\mathbf{I}, \tag{4.5}$$

  where $\bar{s}^2$ is the average sample variance. This target is optimal among all targets of the form $c\mathbf{I}$ with $c \in \mathbb{R}$. The corresponding optimal α is

$$\alpha = \frac{\frac{1}{T^2}\sum_{k=1}^{T} \text{Tr}\left(\left[\mathbf{z}_k\mathbf{z}_k^{\mathsf{T}} - \mathbf{\Sigma}_{\text{sam}}\right]^2\right)}{\text{Tr}\left(\left[\mathbf{\Sigma}_{\text{sam}} - \mathbf{B}_{\text{I}}\right]^2\right)},$$

  where $\mathbf{z}_k$ is observation k of the centered data matrix $\mathbf{Z}$. See the details in [LW04].

  The sample covariance matrix $\mathbf{\Sigma}_{\text{sam}}$ can be ill-conditioned because estimation tends to scatter the sample eigenvalues further away from the mean $\bar{\sigma}^2$ of the true unknown eigenvalues. This raises the condition number of $\mathbf{\Sigma}_{\text{sam}}$ relative to the true matrix Σ and this effect is stronger as the condition number of Σ is better or as N/T grows.

The above shrinkage estimator basically pulls all sample eigenvalues back towards their grand mean $\bar{s}^2$, the estimate of $\bar{\sigma}^2$, thus stabilizing the matrix. We can also see this as a form of regularization.

- Single factor covariance matrices implied by the CAPM[2]:

$$\mathbf{B}_{\text{CAPM}} = s^2_{\text{mkt}} \beta \beta^{\mathsf{T}} + \mathbf{\Sigma}_{\varepsilon}, \tag{4.6}$$

where s^2_{mkt} is the sample variance of market returns, β is the vector of factor exposures, and $\mathbf{\Sigma}_{\varepsilon}$ is the diagonal matrix containing the variances of the residuals. The single factor model is discussed in Sec. 5. In practice a proxy of the market factor could be the equally-weighted or the market capitalization-weighted portfolio of the constituents of a market index, resulting in two different shrinkage targets. The optimal shrinkage intensity is derived in [LW03b].

- Constant correlation covariance matrix:

$$\mathbf{B}_{\text{CC}} = \bar{r} \sqrt{\mathbf{s}} \sqrt{\mathbf{s}}^{\mathsf{T}} + (1 - \bar{r}) \text{Diag}(\mathbf{s}), \tag{4.7}$$

where $\mathbf{s} = \text{diag}(\mathbf{\Sigma}_{\text{sam}})$, and $\bar{r} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} r_{i,j}$ is the average sample correlation. The optimal shrinkage intensity is derived in [LW03a].

Notice that all three targets are positive definite matrices. Shrinkage towards a positive target guarantees that the resulting estimate is also positive, even when the sample covariance matrix itself is singular. Further advantage of this estimator is that it works also in the case of singular covariance matrices, when $N > T$.

In [LW20], the authors present a nonlinear version of this estimator, which can apply different shrinkage intensity to each sample eigenvalue, resulting in even better performance.

4.2.2 Covariance de-noising

According to [dP19], the shrinkage method in Sec. 4.2.1 does not discriminate between the eigenvalues associated with noise and the eigenvalues associated with signal, thus it weakens the signal as well. With the help of random matrix theory the method discussed here shrinks only the part of the covariance matrix that is associated with noise. Taking the covariance matrix $\mathbf{\Sigma}$ it consists of the following steps:

1. Compute the correlation matrix from the covariance matrix: $\mathbf{C} = \mathbf{D}^{-1/2} \mathbf{\Sigma} \mathbf{D}^{-1/2}$, where $\mathbf{D} = \text{Diag}(\mathbf{\Sigma})$.

2. Compute the eigendecomposition of $\mathbf{C}$ and compute the empirical distribution of the eigenvalues using Kernel Density Estimation.

3. Fit the Marčenko–Pastur distribution to the empirical distribution, which gives the theoretical bounds λ_+ and λ_- on the eigenvalues associated with noise. This way we separate the spectrum into a random matrix (noise) component and the signal component. The eigenvalues of the latter will be above the theoretical upper bound λ_+.

[2] The Capital Asset Pricing Model (CAPM) relates the expected return of securities (particularly stocks) to their systematic risk. We can calculate it as $\mathbb{E}(R_i) = r^{\text{f}} + \beta_i(\mathbb{E}(R_{\text{mkt}}) - r^{\text{f}})$, where $\mathbb{E}(R_i)$ is the expected return, R^{f} is the risk-free rate, β_i is the measure of systematic risk, and $\mathbb{E}(R_{\text{mkt}})$ is the expected return of the market.

4. Change the eigenvalues below λ_+ to their average, arriving at the de-noised correlation matrix $\tilde{\mathbf{C}}$.

5. Recover the de-noised covariance matrix: $\tilde{\mathbf{\Sigma}} = \mathbf{D}^{1/2}\tilde{\mathbf{C}}\mathbf{D}^{1/2}$.

4.2.3 Robust estimation

The reason for estimation error is partly that maximum likelihood estimators are highly sensitive to deviations from the assumed (typically normal) distribution. The empirical distribution of security returns usually deviates from the normal distribution [VD09]. Therefore it can be helpful to use robust estimation methods in the construction of mean-variance portfolios. Robust estimators are not as efficient as the maximum likelihood estimator but are stable even when the underlying model is not perfectly satisfied by the available data.

There are two approaches to involve robust statistics in portfolio selection:

- One-step methods: We perform the robust estimation and the portfolio optimization in a single step. We substitute the portfolio risk and portfolio return expressions by their robust counterparts. See for example in [VD09]. We discuss some examples in section Sec. 8.

- Two-step methods: First we compute the robust estimates of the mean vector and the covariance matrix of the data. This is followed by solving the usual portfolio optimization problem with the parameters replaced by their robust estimates. There are various robust covariance estimators, such as Minimum Covariance Determinant (MCD), 2D Winsorization, S-estimators. See for example in [RSTF20, WZ07]. We do not cover this topic in detail in this book.

4.3 Using constraints

Solutions from unconstrained portfolio optimization – while having elegant analytical formulas – can be unreliable in practice. The optimization can significantly overweight securities with large estimated returns, negative correlations, and small variances. The securities with the largest estimation error will get the largest positions. There will also be many zero positions, contradicting diversification [AZ10, MM08].

Financially meaningful constraints on the portfolio weights can also act as regularization, reducing error in the optimal portfolio [CM13]. In [DGNU09], L_1 and L_2 norm constraints are introduced for this purpose:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \frac{\delta}{2}\mathbf{x}^\mathsf{T}\mathbf{\Sigma}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \|\mathbf{x}\|_p^p & \leq & c. \end{array} \tag{4.8}$$

If $p = 1$ and $c = 1$ we get back the constraint $\mathbf{x} \geq 0$, which prevents short-selling. In [JM03] the authors found that preventing short-selling or limiting position size is equivalent to shrinking all covariances of securities for which the respective constraint is binding. The short-selling constraint also implies shrinkage effect on the mean vector toward zero. In a conic optimization problem, the 1-norm constraint can be modeled based on Sec. 13.1.1.

For the case $p = 1$ and $c > 1$ with the constraint $\mathbf{1}^\mathsf{T}\mathbf{x} = 1$ also present, we get a "short-sale budget", meaning that the total weight of short positions will be bounded by $(c - 1)/2$. This allows the possibility of optimizing e. g. 130/30 type portfolios with $c = 1.6$.

If $p = 2$ and the only constraint is $\mathbf{1}^\mathsf{T}\mathbf{x} = 1$, then the minimum feasible value of c is $1/N$, giving the $\mathbf{x} = \mathbf{1}/N$ portfolio as the only possible solution. With larger values of c we get an effect equivalent to the Ledoit–Wolf shrinkage towards multiple of the identity. For other shrinkage targets $\mathbf{B}$, the corresponding constraint will be $\mathbf{x}^\mathsf{T}\mathbf{B}\mathbf{x} \leq c$ or $\|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2^2 \leq c$ where $\mathbf{G}\mathbf{G}^\mathsf{T} = \mathbf{B}$. In a conic optimization problem, we can model the 2-norm constraint based on Sec. 13.1.1.

In general,

- adding an L_1 norm-constraint is equivalent to shrinking all covariances of securities that are being sold short. Instead of the elementwise short-sale constraints here we impose a global short-sale budget using only one constraint. This means that the implied shrinkage coefficient will be the same for all securities. According to the usual behavior of LASSO regression, the L_1 norm-constrained portfolios are likely to assign a zero weight to at least some of the securities. This could be beneficial if we wish to invest only into a small number of securities.

- adding an L_2 norm-constraint is equivalent to shrinking the sample covariance matrix towards the identity matrix. According to properties of ridge regression, the L_2 norm-constrained portfolios will typically have a non-zero weight for all securities, which is good if we want full diversification.

4.4 Improve the optimization

In dealing with estimation error it is also a possibility to directly improve the optimization procedure.

4.4.1 Resampled portfolio optimization

To overcome estimation uncertainty we can use nonparametric bootstrap resampling on the available historical return data. This way we can create multiple instances of the data and generate multiple expected return and covariance matrix estimates using any estimation procedure. Then we compute the efficient frontier from all of them. Finally, we average the obtained solutions to make the effect of the estimation errors cancel out. This procedure is thoroughly analysed in [MM08].

Note that we should not average over portfolios of too different variances. Thus the averaging procedure assumes that for each pair of inputs we generate m points on the efficient frontier, and rank them 1 to m. Then the portfolios with the same rank will be averaged.

In this process we can treat the number of simulated return samples in each batch of resampled data as a parameter. It can be used to measure the level of certainty of the expected return estimate. A large number of simulated returns is consistent with more certainty, while a small number is consistent with less certainty.

While this method offers good results in terms of diversification and mitigating the effects of estimation error, it has to be noted that it does this at the cost of intensive computational work.

According to [MM08], improving the optimization procedure is more beneficial than to use improved estimates (e. g. shrinkage) as input parameter. However, the two approaches do not exclude each other, so they can be used simultaneously.

4.4.2 Nested Clustering Optimization (NCO)

A recent method to tackle estimation error and error sensitivity of the optimization is described in [dP19]. The NCO method is capable of controlling for noise in the inputs, and also for instabilities arising from the covariance structure:

- Noise can arise from lack of data. As N/T approaches 1, the number of data points will approach the number of degrees of freedom in covariance estimation. Also based on random matrix theory, the smallest eigenvalue of the covariance matrix will be close to zero, raising the condition number.
- Instability in the optimal solution comes from the correlation structure, the existence of highly correlating clusters of securities. These will also raise the condition number of the covariance matrix. The most favorable correlation structure is that of the identity matrix.

The steps of NCO are the following:

1. Cluster the securities into highly-correlated groups based on the correlation matrix to get a partition P of the security universe. We could apply hierarchical clustering methods, or the method recommended in [dPL19]. It could be worthwhile to do the clustering also on the absolute value of the correlation matrix to see if it works better.
2. Run portfolio optimization for each cluster separately. Let $\mathbf{x}_k$ denote the N-dimensional optimal weight vector of cluster P_k, such that $x_{k,i} = 0,\ \forall i \notin P_k$.
3. Reduce the original covariance matrix $\mathbf{\Sigma}$ into a covariance matrix $\mathbf{\Sigma}^{\text{cl}}$ between clusters: $\mathbf{\Sigma}^{\text{cl}}_{k,l} = \mathbf{x}_k^{\mathsf{T}} \mathbf{\Sigma} \mathbf{x}_l$. The correlation matrix computed from $\mathbf{\Sigma}^{\text{cl}}$ will be closer to identity, making the inter-cluster optimization more accurate.
4. Run the inter-cluster portfolio optimization by treating each cluster as one security and using the reduced covariance matrix $\mathbf{\Sigma}^{\text{cl}}$. We will denote the resulting k-dimensional optimal weight vector by $\mathbf{x}^{\text{cl}}$.
5. Generate the final optimal portfolio: $\mathbf{x} = \sum_k x^{\text{cl}}_k \mathbf{x}_k$.

By splitting the problem in the above way into two independent parts, the error caused by intra-cluster noise cannot propagate across clusters.

In the paper the author also proposes a method for estimating the error in the optimal portfolio weights. We can use it with any optimization method, and facilitates comparison of methods in practice. The steps of this procedure can be found in Sec. 13.4.

4.5 Example

In this part we add shrinkage estimation to the case study introduced in the previous chapters. We start at the point where the expected weekly logarithmic returns and their covariance matrix is available:

```
return_array = df_weekly_log_returns.to_numpy()
m_weekly_log = np.mean(return_array, axis=0)
S_weekly_log = np.cov(return_array.transpose())
```

The eigenvalues of the covariance matrix are `1e-3 * [0.2993, 0.3996, 0.4156, 0.5468, 0.6499, 0.9179, 1.0834, 4.7805]`. We apply the Ledoit–Wolf shrinkage with target (4.5):

```
N = S_weekly_log.shape[0]
T = return_array.shape[0]

# Ledoit--Wolf shrinkage
S = S_weekly_log
s2_avg = np.trace(S) / N
B = s2_avg * np.eye(N)
Z = return_array.T - m_weekly_log[:, np.newaxis]
alpha_num = alpha_numerator(Z, S)
alpha_den = np.trace((S - B) @ (S - B))
alpha = alpha_num / alpha_den
S_shrunk = (1 - alpha) * S + alpha * B
```

The function `alpha_numerator` is to compute the sum in the numerator of α:

```
def alpha_numerator(Z, S):
    s = 0
    T = Z.shape[1]
    for k in range(T):
        z = Z[:, k][:, np.newaxis]
        X = z @ z.T - S
        s += np.trace(X @ X)
    s /= (T**2)
    return s
```

In this example, we get $\alpha = 0.0843$, and the eigenvalues will become `1e-3 * [0.3699, 0.4618, 0.4764, 0.5965, 0.6909, 0.9363, 1.0879, 4.4732]`, closer to their sample mean as expected.

Next we implement the James–Stein estimator for the expected return vector:

```
# James--Stein estimator
m = m_weekly_log[:, np.newaxis]
o = np.ones(N)[:, np.newaxis]
S = S_shrunk
iS = np.linalg.inv(S)
b = (o.T @ m / N) * o
N_eff = np.trace(S) / np.max(np.linalg.eigvalsh(S))
alpha_num = max(N_eff - 3, 0)
```

(continues on next page)

(continued from previous page)

```
alpha_den = T * (m - b).T @ iS @ (m - b)
alpha = alpha_num / alpha_den
m_shrunk = b + max(1 - alpha, 0) * (m - b)
m_shrunk = m_shrunk[:, 0]
```

In this case though it turns out that the effective dimension $\tilde{N}$ would be too low for the estimator to give an improvement over the sample mean. This is because the eigenvalues of the covariance matrix are spread out too much, meaning that the uncertainty of the mean vector is primarily along the direction of one or two eigenvectors, effectively reducing the dimensionality of the estimation.

We can check the improvement on the plot of the efficient frontier Fig. 4.2. We can observe the portfolio composition for different risk-aversion levels on Fig. 4.2.

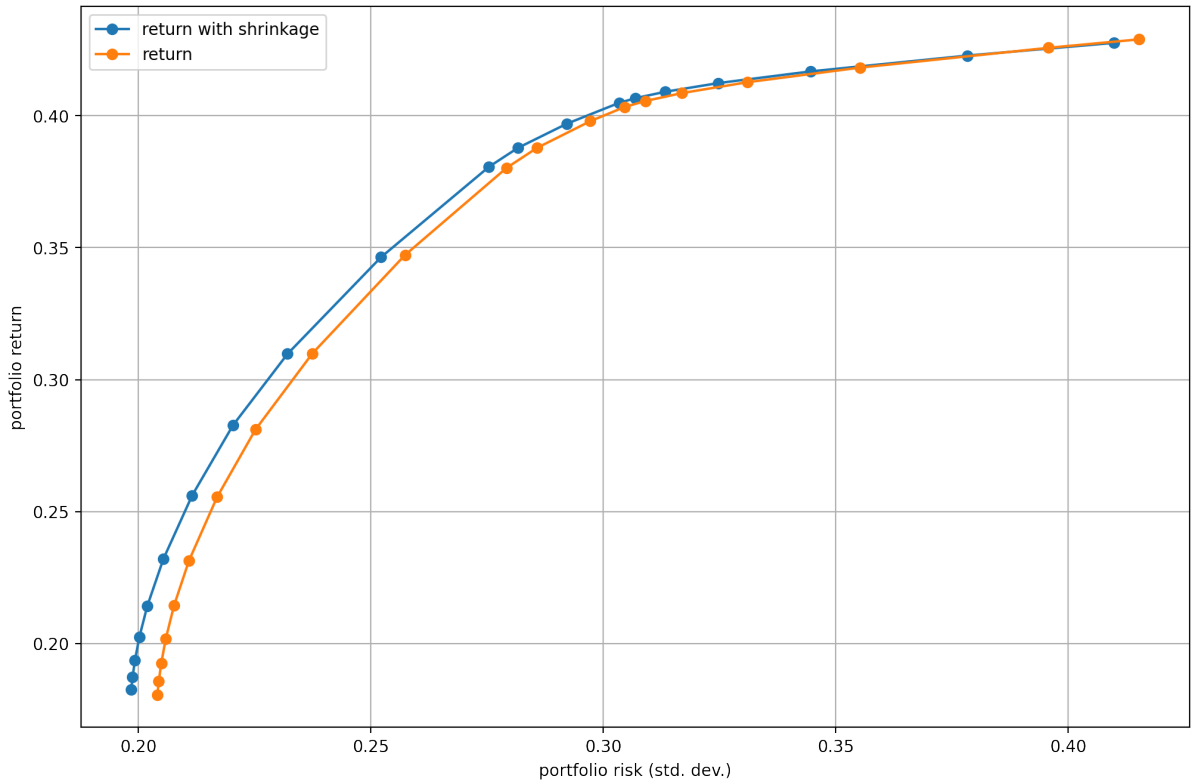

Fig. 4.1: The efficient frontier.

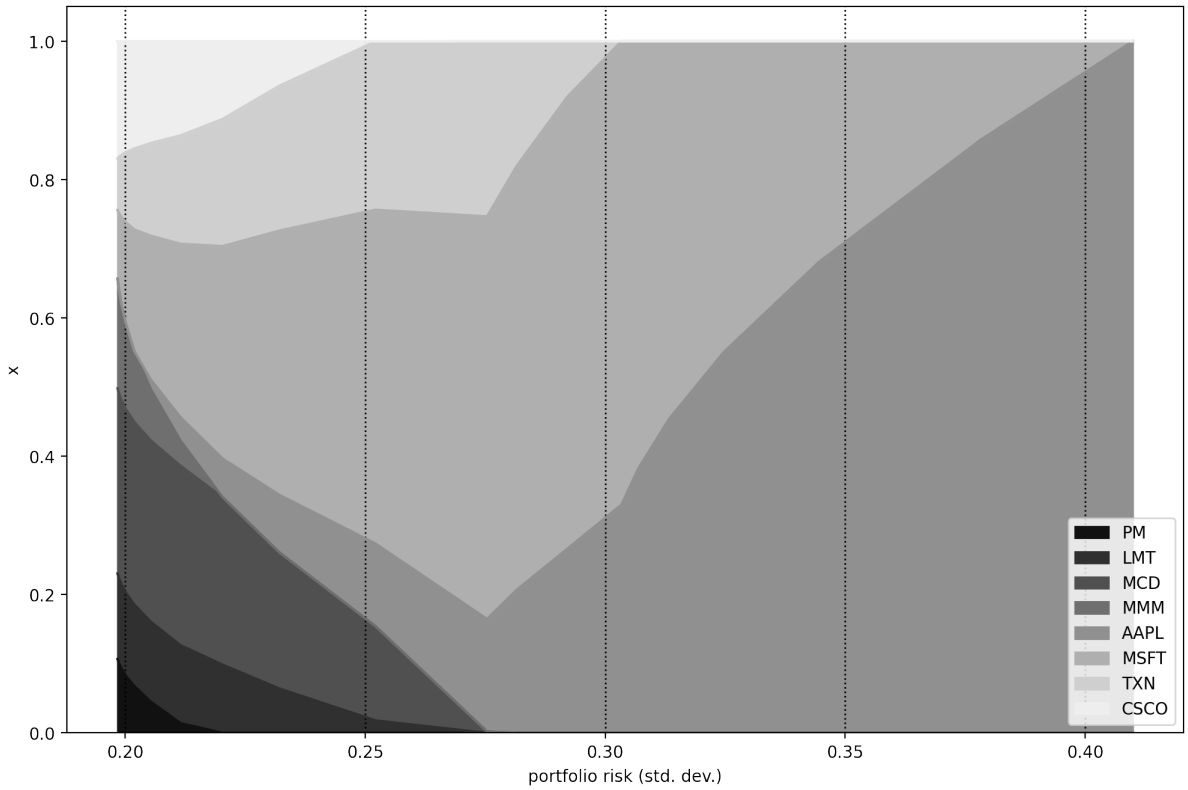

Fig. 4.2: Portfolio composition $\mathbf{x}$ with varying level if risk-aversion δ.

Chapter 5

Factor models

The purpose of *factor models* is to impose a structure on financial variables and their covariance matrix by explaining them through a small number of common factors. This can help to overcome estimation error by reducing the number of parameters, i. e., the dimensionality of the estimation problem, making portfolio optimization more robust against noise in the data. Factor models also provide a decomposition of financial risk to systematic and security specific components [CM13]. Factor models can be generalized in many ways; see in [BS11].

5.1 Definition

Let Z_t be an N-dimensional random vector representing a financial variable at time t. We can write Z_t as a linear function of components as

$$Z_t = \beta F_t + \theta_t,$$

where F_t is a K-dimensional random vector representing the *common factors* at time t, β is the $N \times K$ matrix of *factor exposures* (or *loadings*, *betas*), and $\theta_t = \alpha + \varepsilon_t$ is the *specific component* such that $\mathbb{E}(\theta_t) = \alpha$ and ε_t is white noise[1] with covariance Σ_θ. If Z_t represents security returns, then $\beta\mathbb{E}(F_t)$ is the expected explained return and α is the expected unexplained return.

Factor models typically work with the following assumptions:

- *Exact* factor model: All common movement between pairs of variables is modeled by the factors. This means that the covariances of Z_t is determined only by the covariances of F_t. This gives the conditions $\mathrm{Cov}(\varepsilon_{t_1}, F_{t_2}) = \mathbf{0}$ for all t_1, t_2 and that $\mathrm{Cov}(\varepsilon_t) = \Sigma_\varepsilon$ is diagonal.[2]
- *Static* factor model: F_t only has a contemporaneous effect on Z_t.[3]
- Weakly stationary factors: The sequence of random variables F_t are weakly stationary so that $\mathbb{E}(F_t) = \mu_F$ and $\mathrm{Cov}(F_t) = \Sigma_F$. This allows us to estimate the model from (historical) scenarios.

[1] For a white noise process ε_t we have $\mathbb{E}(\varepsilon_t) = \mathbf{0}$ and $\mathrm{Cov}(\varepsilon_{t_1}, \varepsilon_{t_2}) = \Omega$ if $t_1 = t_2$, where Ω does not depend on t, otherwise $\mathbf{0}$.

[2] If Σ_ε is not diagonal then we have an *approximate* factor model.

[3] Additional lags of F_t in the Z_t equations can be easily allowed, then we obtain a dynamic factor model.

With these assumptions, we can write the expected value of Z_t as

$$\mu_Z = \alpha + \beta\mu_F.$$

The covariance matrix will be given by

$$\Sigma_Z = \beta\Sigma_F\beta^\mathsf{T} + \Sigma_\theta.$$

These two equations allow the following properties:

- Number of parameters: The covariance matrix Σ_Z has only $NK + N + K(K+1)/2$ parameters, which is linear in N, in contrast to having $N(N+1)/2$ covariances, which is quadratic in N.
- Risk decomposition: Because the common factors and the specific components are uncorrelated, we can separate the risks associated with each. For a portfolio $\mathbf{x}$ the portfolio level factor exposures are given by $\mathbf{b} = \beta^\mathsf{T}\mathbf{x}$ and we can write the total portfolio variance as $\mathbf{b}^\mathsf{T}\Sigma_F\mathbf{b} + \mathbf{x}^\mathsf{T}\Sigma_\theta\mathbf{x}$.
- While the factor structure may introduce a bias on the covariance estimator, it will also lower its variance owing to the reduced number of parameters to estimate.

5.2 Classes of factor models

5.2.1 Explicit factor models

In these models the factors are predefined based on financial or economic theory, independently from the data. They are observed from e. g. macroeconomic quantities (inflation, economic growth, etc.) or from security characteristics (industry, dividend yield, market capitalization, etc.) [Con95]. The goal of these models is typically to explain security returns. A drawback of explicit factor models is the misspecification risk. The derivation of the covariance matrix Σ_Z strongly relies on the orthogonality between the factors F_t and the residuals θ_t, which might not be satisfied if relevant factors are missing from the model.

Macroeconomic models

In the case of *macroeconomic factor models* we observe the historical factor time-series $\mathbf{F}$ and apply time-series regression to get estimators for β, α, and ε. However, enough stationary data must be available for an accurate regression.

The simplest macroeconomic factor models use a single factor as common risk component. In Sharpe's single factor model this common factor is the market risk. In practice, we can approximate the market risk factor using the returns of a stock index, by either forming an equally-weighted or a market capitalization-weighted portfolio of the index constituents. The covariance matrix with single factor model structure has only $2N + 1$ parameters to estimate.

Fundamental models

There are two methods to estimate *fundamental factor models*:

- BARRA method: We compute the estimate $\boldsymbol{\beta}$ of the β from the observed security specific attributes (assumed to be time invariant). Then we do cross-sectional regression for each time instant t to obtain the factor time-series $\mathbf{F}$ and its estimated covariance matrix $\boldsymbol{\Sigma}_F$: $\mathbf{F}_t = (\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Sigma}_\theta^{-1}\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Sigma}_\theta^{-1}\mathbf{Z}_t$, where $\boldsymbol{\Sigma}_\theta$ is the diagonal matrix of estimated specific variances. It is there to take into account cross-sectional heteroskedasticity in the model.

- Fama–French method: For each time t we order the securities into quintiles by a given security attribute. Then we long the the top quintile and short the bottom quintile. The factor realization at time t corresponding to the chosen attribute will be the return of this hedged portfolio. After obtaining the factor time-series corresponding to each attribute, we can estimate β using time-series regression.

5.2.2 Implicit factor models

Implicit factor models derive both the factors and the factor exposures from the data, so these does not need any external input. However, the statistical estimation procedures involved are prone to discovering spurious correlations, and they can only work if we assume the factor exposures to be time invariant over the estimation period. Also, implicit factors do not have financial interpretation. Mainly two methods are used to derive the factors.

Factor analysis

This method assumes a more specific structure, a *normal factor model* that has the additional requirements $\mathbb{E}(F_t) = \mathbf{0}$ and $\mathrm{Cov}(F_t) = \mathbf{I}$.

- To satisfy the first condition, we redefine $\alpha' = \alpha + \beta\mathbb{E}(F_t)$.

- To satisfy the second condition, we observe that the variables β and $\mathbf{F}_t$ can only be determined up to an invertible matrix $\mathbf{H}$, because $\beta\mathbf{F}_t = \beta\mathbf{H}^{-1}\mathbf{H}\mathbf{F}_t$. Thus we decompose the factor covariance as $\Sigma_F = \mathbf{Q}\mathbf{Q}^\mathsf{T}$ and choose $\mathbf{H} = \mathbf{Q}^{-1}$ to arrive at the desired structure.

The covariance matrix will then become $\Sigma_Z = \beta\beta^\mathsf{T} + \Sigma_\theta$. The matrix $\mathbf{Q}$ is still determined only up to a rotation. This freedom can help in finding interpretation for the factors.

Further assuming that the variables Z_t are independent and identically normally distributed, we can compute estimates $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\Sigma}_\theta$ using maximum likelihood method. Then we use cross-sectional regression to estimate the factor time-series: $\mathbf{F}_t = (\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Sigma}_\theta^{-1}\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Sigma}_\theta^{-1}(\mathbf{z}_t - \boldsymbol{\alpha})$. Here we used $\boldsymbol{\Sigma}_\theta^{-1}$ as weighting to address cross-sectional heteroskedasticity in ε_t.

The number of factors can be determined e. g. using likelihood ratio test. The drawback of factor analysis is that it is not efficient for large problems.

Principal Component Analysis

Definition

Principal component analysis (PCA) is an affine dimension reduction method. Let us define factors (called *principal components*) through $F_t = \mathbf{d} + \mathbf{A}Z_t$, assuming $\mathbb{E}(F_t) = \mathbf{0}$, and $\dim(F_t) = K < N = \dim(Z_t)$. Then we can approximate Z_t with $\tilde{Z}_t = \alpha + \beta F_t$. The best such approximation is obtained by the eigendecomposition $\Sigma_Z = \mathbf{V}\Lambda\mathbf{V}^\mathsf{T}$, where the matrix Λ is diagonal with the eigenvalues $\lambda_1, \ldots, \lambda_N$ in it ordered from largest to smallest, and the matrix $\mathbf{V}$ contains the corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_N$ as columns.

If we partition the eigenvector matrix as $\mathbf{V} = [\mathbf{V}_K, \mathbf{V}_{N-K}]$, where $\mathbf{V}_K$ consists of the first K eigenvectors, we can construct the solution:

$$\begin{aligned} \beta &= \mathbf{V}_K, \\ \alpha &= \mathbb{E}(Z_t), \\ \mathbf{F}_t &= \mathbf{V}_K^\mathsf{T}(Z_t - \mathbb{E}(Z_t)), \\ \varepsilon_t &= Z_t - \tilde{Z}_t = \mathbf{V}_{N-K}\mathbf{V}_{N-K}^\mathsf{T}(Z_t - \mathbb{E}(Z_t)). \end{aligned}$$

Note that for ε_t we have $\mathbb{E}(\varepsilon_t) = \mathbf{0}$ and $\mathrm{Cov}(F_t, \varepsilon_t) = \mathbf{0}$, as required in the factor model definition. Also the factors will be uncorrelated: $\Sigma_F = \frac{1}{T}F_t F_t^\mathsf{T} = \Lambda_K$, containing the first K eigenvalues in the diagonal.

Intuitively PCA does an orthogonal projection of Z_t onto the hyperplane spanned by the K directions corresponding to the K largest eigenvalues. This way the projection $\tilde{Z}_t$ contains the most randomness of the original variable Z_t that an affine transformation can preserve. The eigenvalue λ_i measures the randomness of principal component $F_{t,i}$, the randomness of Z_t along the direction $\mathbf{v}_i$. The ratio $\sum_{i=1}^K \lambda_i / \sum_{i=1}^N \lambda_i$ is the fraction of randomness explained by the factor model approximation.

Advantages of PCA based factor models is that they are computationally simple, they need no external data, and that they avoid misspecification risk because the residuals and the factors are orthogonal by construction. Also they yield factors ranked according to their fraction of explained variance, which can help determining the number of factors K to use.

Number of factors

In practice, the number of factors K is unknown and has to be estimated. Too few factors reduce the explanatory power of the model, too many could lead to retaining insignificant factors. In [BN01] the authors propose an information criteria to estimate K:

$$K^* = \mathrm{argmin}_K \log\left(\varepsilon_K^T \varepsilon_K\right) + K\left(\frac{N+T}{NT}\right)\log\left(\frac{NT}{N+T}\right),$$

where ε_K is the vector of residuals in the case of K factors.

If we wish to keep the number of factors time invariant, then a heuristic method can help: $K = \lfloor \log(N) \rfloor$.

PCA on the correlation matrix

We can also do PCA on the correlation matrix:

- We get the correlation matrix Ω_Z from the covariance matrix Σ_Z by $\Omega_Z = \mathbf{D}^{-1/2}\Sigma_Z\mathbf{D}^{-1/2}$, where $\mathbf{D} = \text{Diag}(\Sigma_Z)$.
- Then we keep only the K largest eigenvalues: $\tilde{\Omega}_Z = \mathbf{V}_K\Lambda_K\mathbf{V}_K^\mathsf{T}$.
- To guarantee that this approximation will also be a correlation matrix, we add a diagonal correction term $\text{Diag}\left(\mathbf{I} - \tilde{\Omega}_Z\right)$.
- Finally, we transform the result back into a covariance matrix.

This method can give better performance when the scale of data variables is very different.

Practical considerations

In practice, PCA relies on the assumption that the number of observations T is large relative to the number of securities N. If this does not hold, i. e., $N > T$, then we can do PCA in a different way. Suppose that $\mathbf{Z}$ is the $N \times T$ data matrix, and $\boldsymbol{\mu}$ is the mean of the data. Then

- We do eigendecomposition on the $T \times T$ matrix $\frac{1}{N}(\mathbf{Z} - \boldsymbol{\mu}\mathbf{1}^\mathsf{T})^\mathsf{T}(\mathbf{Z} - \boldsymbol{\mu}\mathbf{1}^\mathsf{T}) = \mathbf{W}\Lambda\mathbf{W}^\mathsf{T}$.
- We get the principal component matrix as $\mathbf{F} = \Lambda\mathbf{W}^\mathsf{T}$.

The most efficient way to compute PCA is to use singular value decomposition (SVD) directly on the data matrix $\mathbf{Z}$. Then we get $\mathbf{Z} = \mathbf{V}\Lambda\mathbf{W}^\mathsf{T}$. Now it is easy to see the connection between the two PCA approaches:

$$\mathbf{F} = \mathbf{V}^\mathsf{T}\mathbf{Z} = \mathbf{V}^\mathsf{T}\mathbf{V}\Lambda\mathbf{W}^\mathsf{T} = \Lambda\mathbf{W}^\mathsf{T}.$$

5.3 Portfolio optimization with factor model

5.3.1 Sparsity

Suppose we have the decomposition of the covariance matrix $\boldsymbol{\Sigma}_Z = \boldsymbol{\beta}\boldsymbol{\Sigma}_F\boldsymbol{\beta}^\mathsf{T} + \boldsymbol{\Sigma}_\theta$, where $\boldsymbol{\Sigma}_F$ is the $K \times K$ sample factor covariance matrix and K is the number of factors. Typically we have only a few factors, so $K \ll N$ and thus $\boldsymbol{\Sigma}_F$ is much lower dimensional than the $N \times N$ matrix $\boldsymbol{\Sigma}_Z$. This makes it much cheaper to find the decomposition $\boldsymbol{\Sigma}_F = \mathbf{F}\mathbf{F}^\mathsf{T}$, and consequently the factorization $\boldsymbol{\Sigma}_Z = \mathbf{G}\mathbf{G}^\mathsf{T}$, where

$$\mathbf{G} = \begin{bmatrix} \boldsymbol{\beta}\mathbf{F} & \boldsymbol{\Sigma}_\theta^{1/2} \end{bmatrix}. \tag{5.1}$$

This form of $\mathbf{G}$ is typically very sparse. In practice sparsity is more important than the number of variables and constraints in determining the computational efficiency of the optimization problem. Thus it can be beneficial to focus on the number of nonzeros in the matrix $\mathbf{G}$ and try to reduce it. Indeed, the $N \times (N + K)$ dimensional $\mathbf{G}$ is larger than the

$N \times N$ dimensional Cholesky factor of $\mathbf{\Sigma}_Z$ would be, but in $\mathbf{G}$ there are only $N(K+1)$ nonzeros in contrast to the $N(N+1)/2$ nonzeros in the Cholesky factor. Because in practice K tends to be a small number independent of N, we can conclude that the usage of a factor model reduced the number of nonzeros, and thus the storage requirement by a factor of N. This will in most cases also lead to a significant reduction in the solution time.

5.3.2 Modeling risk terms

In the risk minimization setting (2.1) of the Markowitz problem, according to Sec. 13.1.1 we can model the factor risk term $\mathbf{x}^\mathsf{T}\boldsymbol{\beta}\mathbf{\Sigma}_F\boldsymbol{\beta}^\mathsf{T}\mathbf{x} \leq t_1$ as $(\frac{1}{2}, t_1, \mathbf{F}^\mathsf{T}\boldsymbol{\beta}^\mathsf{T}\mathbf{x}) \in \mathcal{Q}^{K+2}$, and the specific risk term $\mathbf{x}^\mathsf{T}\mathbf{\Sigma}_\theta\mathbf{x} \leq t_2$ as $(\frac{1}{2}, t_2, \mathbf{\Sigma}_\theta^{1/2}\mathbf{x}) \in \mathcal{Q}^{N+2}$. The latter can be written in a computationally more favorable way as $\left(\frac{1}{2}, t_2, \operatorname{diag}\left(\mathbf{\Sigma}_\theta^{1/2}\right) \circ \mathbf{x}\right) \in \mathcal{Q}^{N+2}$. The resulting risk minimization problem will then look like

$$\begin{array}{lrcl} \text{minimize} & t_1 + t_2 & & \\ \text{subject to} & (\frac{1}{2}, t_1, \mathbf{F}^\mathsf{T}\boldsymbol{\beta}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{K+2}, \\ & \left(\frac{1}{2}, t_2, \operatorname{diag}\left(\mathbf{\Sigma}_\theta^{1/2}\right) \circ \mathbf{x}\right) & \in & \mathcal{Q}_\mathrm{r}^{N+2}, \\ & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & \geq & r_{\min}, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{5.2}$$

We can also have prespecified limits γ_1 for the factor risk and γ_2 for the specific risk. Then we can use the return maximization setting:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & (\frac{1}{2}, \gamma_1^2, \mathbf{F}^\mathsf{T}\boldsymbol{\beta}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{K+2}, \\ & \left(\frac{1}{2}, \gamma_2^2, \operatorname{diag}\left(\mathbf{\Sigma}_\theta^{1/2}\right) \circ \mathbf{x}\right) & \in & \mathcal{Q}_\mathrm{r}^{N+2}, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{5.3}$$

5.4 Shrinkage as factor model

In section Sec. 4.2.1 we discussed that shrinkage is a regularization scheme, useful in handling the inaccuracy in the covariance matrix estimate (the sample covariance matrix can be singular and also its offdiagonal elements can be out-of-sample unstable). We can also look at the shrinkage of the covariance matrix as a factor model [Kak16]. This requires that the target matrix $\mathbf{B}$ is also written as factor model:

$$\mathbf{B} = \boldsymbol{\beta}\mathbf{\Sigma}_F\boldsymbol{\beta}^\mathsf{T} + \mathbf{\Sigma}_\theta,$$

where the number of factors K should be $K \ll N$ so that the factor covariance $\mathbf{\Sigma}_F$ is more stable than the sample covariance $\mathbf{\Sigma}_\mathrm{sam}$. Note that all three targets in section Sec. 4.2.1 are given in this form.

We also assume that the sample covariance is decomposed into principal components $\mathbf{\Sigma}_\mathrm{sam} = \mathbf{V}_L\Lambda_L\mathbf{V}_L^\mathsf{T}$, where some eigenvalues could be zero because of possible singularity, so in general we assume $L < N$ nonzero eigenvalues.

Then, applying (4.4) the shrunk matrix can be expressed as the factor model

$$\mathbf{\Sigma}_\mathrm{shrunk} = \tilde{\boldsymbol{\beta}}\tilde{\mathbf{\Sigma}}_F\tilde{\boldsymbol{\beta}}^\mathsf{T} + \tilde{\mathbf{\Sigma}}_\theta,$$

where the components are block matrices

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_\theta &= \alpha \boldsymbol{\Sigma}_\theta, \\ \tilde{\boldsymbol{\beta}} &= \begin{bmatrix} \boldsymbol{\beta} & \mathbf{V}_L \end{bmatrix}, \\ \tilde{\boldsymbol{\Sigma}}_F &= \begin{bmatrix} \alpha \boldsymbol{\Sigma}_F & \mathbf{0} \\ \mathbf{0} & (1-\alpha)\Lambda_L \end{bmatrix}. \end{aligned}$$

We can further improve this model by dropping some of the "noisy" principal components, and keep only $\hat{L} < L$ eigenvalues, thus making the shrunk covariance more stable. We consider the matrix

$$\boldsymbol{\Sigma}_{\text{shrunk}} = \mathbf{V}_{\hat{L}} \Lambda_{\hat{L}} \mathbf{V}_{\hat{L}}^{\mathsf{T}} + \mathbf{c}\mathbf{c}^{\mathsf{T}} \circ \mathbf{B}. \tag{5.4}$$

Instead of using a shrinkage coefficient, here we require that $\operatorname{diag}(\boldsymbol{\Sigma}_{\text{shrunk}}) = \operatorname{diag}(\boldsymbol{\Sigma}_{\text{sam}})$, which is a reasonable requirement, and determines $\mathbf{c}$: $c_i^2 = \frac{1}{b_{i,i}} \sum_{l=\hat{L}+1}^{L} \lambda_l v_{i,l}^2$, where $b_{i,i}$ is a diagonal element of $\mathbf{B}$ and $v_{i,l}$ is an element of $\mathbf{V}_L$. Note that (5.4) is a factor model, if $\mathbf{B}$ is a factor model.[4]

So we arrived at another regularization scheme, which improves stability of the covariance matrix estimate. This method is also useful for combining principal components with fundamental factors.

5.5 Example

In this chapter there are two examples. The first shows the application of macroeconomic factor models on the case study developed in the preceding chapters. The second example demonstrates the performance gain that factor models can yield on a large scale portfolio optimization problem.

5.5.1 Single factor model

In the example in Sec. 4.5 we have seen that the effective dimension $\tilde{N}$ is low, indicating that there are only one or two independent sources of risk for all securities. This suggests that a single factor model might give a good approximation of the covariance matrix. In this example we will use the return of the SPY ETF as market factor. The SPY tracks the S&P 500 index and thus is a good proxy for the U.S. stock market. Then our model will be:

$$R_t = \alpha + \beta R_{\text{M},t} + \varepsilon_t,$$

where R_{M} is the return of the market factor. To estimate this model using time-series regression, we need to obtain scenarios of linear returns on the investment time horizon ($h = 1$ year in this example), both for the individual securities and for SPY.

To get this, we add SPY as the ninth security, and proceed exactly the same way as in Sec. 3.4 up to the point where we have the expected yearly logarithmic return vector $\boldsymbol{\mu}_h^{\log}$ and covariance matrix $\boldsymbol{\Sigma}_h^{\log}$. Then we generate Monte Carlo scenarios using these parameters.

[4] This is because $\mathbf{c}\mathbf{c}^{\mathsf{T}} \circ (\boldsymbol{\beta}\boldsymbol{\Sigma}_F\boldsymbol{\beta}^{\mathsf{T}} + \boldsymbol{\Sigma}_\theta) = [\operatorname{Diag}(\mathbf{c})\boldsymbol{\beta}]\boldsymbol{\Sigma}_F[\operatorname{Diag}(\mathbf{c})\boldsymbol{\beta}]^{\mathsf{T}} + \operatorname{Diag}(\mathbf{c} \circ \mathbf{c})\boldsymbol{\Sigma}_\theta$.

```
scenarios_log = \
    np.random.default_rng().multivariate_normal(m_log, S_log, 100000)
```

Next, we convert the scenarios to linear returns:

```
scenarios_lin = np.exp(scenarios_log) - 1
```

Then we do the linear regression using the `OLS` class of the `statsmodels` package. The independent variable `X` will contain the scenarios for the market returns and a constant term. The dependent variable `y` will be the scenarios for one of the security returns.

```
params = []
resid = []
X = np.zeros((scenarios_lin.shape[0], 2))
X[:, 0] = scenarios_lin[:, -1]
X[:, 1] = 1
for k in range(N):
    y = scenarios_lin[:, k]
    model = sm.OLS(y, X, hasconst=True).fit()
    resid.append(model.resid)
    params.append(model.params)
resid = np.array(resid)
params = np.array(params)
```

After that we derive the estimates $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, s_{M}^2, and $\mathrm{diag}(\boldsymbol{\Sigma}_\theta)$ of the parameters α, β, σ_{M}^2, and $\mathrm{diag}(\Sigma_\theta)$ respectively.

```
a = params[:, 1]
B = params[:, 0]
s2_M = np.var(X[:, 0])
S_theta = np.cov(resid)
diag_S_theta = np.diag(S_theta)
```

At this point, we can compute the decomposition (5.1) of the covariance matrix:

```
G = np.block([[B[:, np.newaxis] * np.sqrt(s_M), np.sqrt(S_theta)]])
```

We can see that in this 8×9 matrix there are only 16 nonzero elements:

```
G = np.array([
    [0.174, 0.253, 0,     0,     0,     0,     0,     0,     0    ],
    [0.189, 0,     0.205, 0,     0,     0,     0,     0,     0    ],
    [0.171, 0,     0,     0.181, 0,     0,     0,     0,     0    ],
    [0.180, 0,     0,     0,     0.190, 0,     0,     0,     0    ],
    [0.274, 0,     0,     0,     0,     0.314, 0,     0,     0    ],
    [0.225, 0,     0,     0,     0,     0,     0.201, 0,     0    ],
    [0.221, 0,     0,     0,     0,     0,     0,     0.218, 0    ],
    [0.191, 0,     0,     0,     0,     0,     0,     0,     0.213]
])
```

This sparsity can considerably speed up the optimization. See the next example in Sec. 5.5.2 for a large scale problem where this speedup is apparent.

While **MOSEK** can handle the sparsity efficiently, we can exploit this structure also at the model creation phase, if we handle the factor risk and specific risk parts separately,

avoiding a large matrix-vector product:

```
G_factor = B[:, np.newaxis] * np.sqrt(s_M)
g_specific = np.sqrt(diag_S_theta)

factor_risk = G_factor.T @ x
specific_risk = Expr.mulElm(g_specific, x)
```

Then we can form the risk constraint in the usual way:

```
M.constraint('risk', Expr.vstack(gamma2, 0.5, factor_risk, specific_risk),
                     Domain.inRotatedQCone())
```

Finally, we compute the efficient frontier in the following points:

```
deltas = np.logspace(start=-1, stop=1.5, num=20)[::-1]
```

If we plot the efficient frontier on Fig. 5.1, and the portfolio composition on Fig. 5.2 we can compare the results obtained with and without using a factor model.

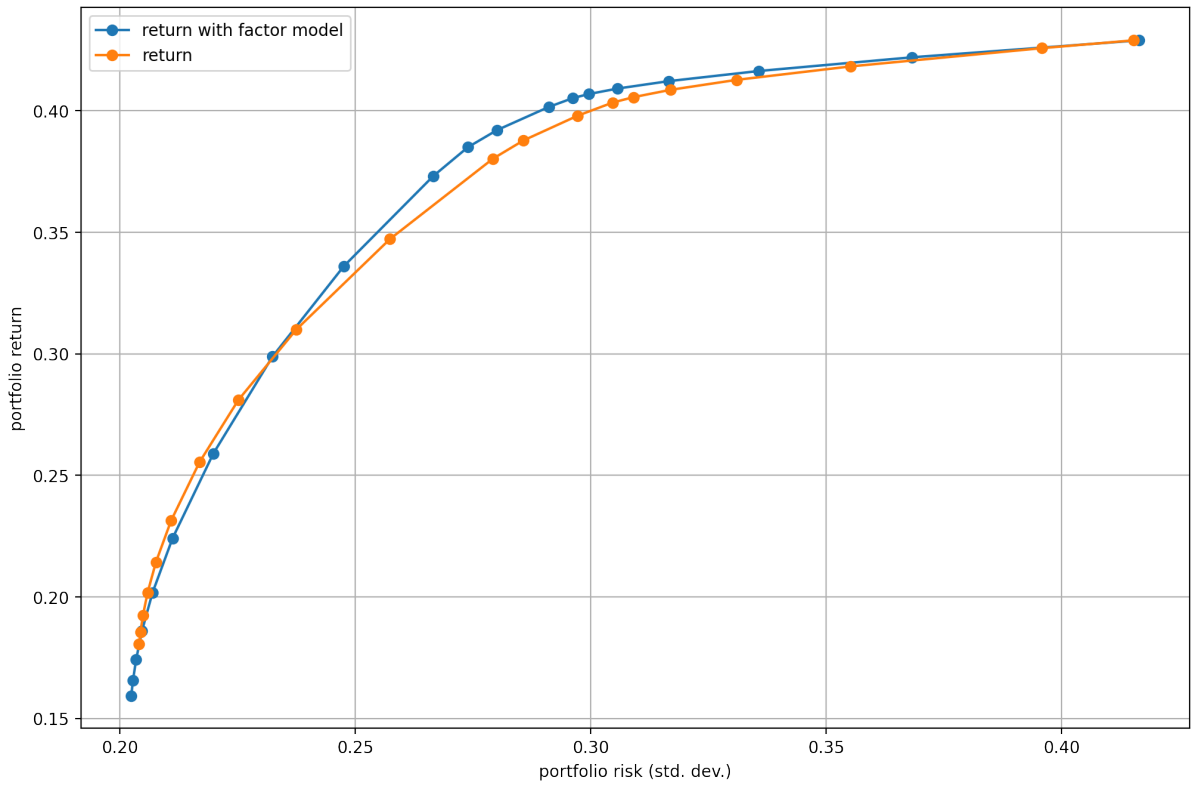

Fig. 5.1: The efficient frontier.

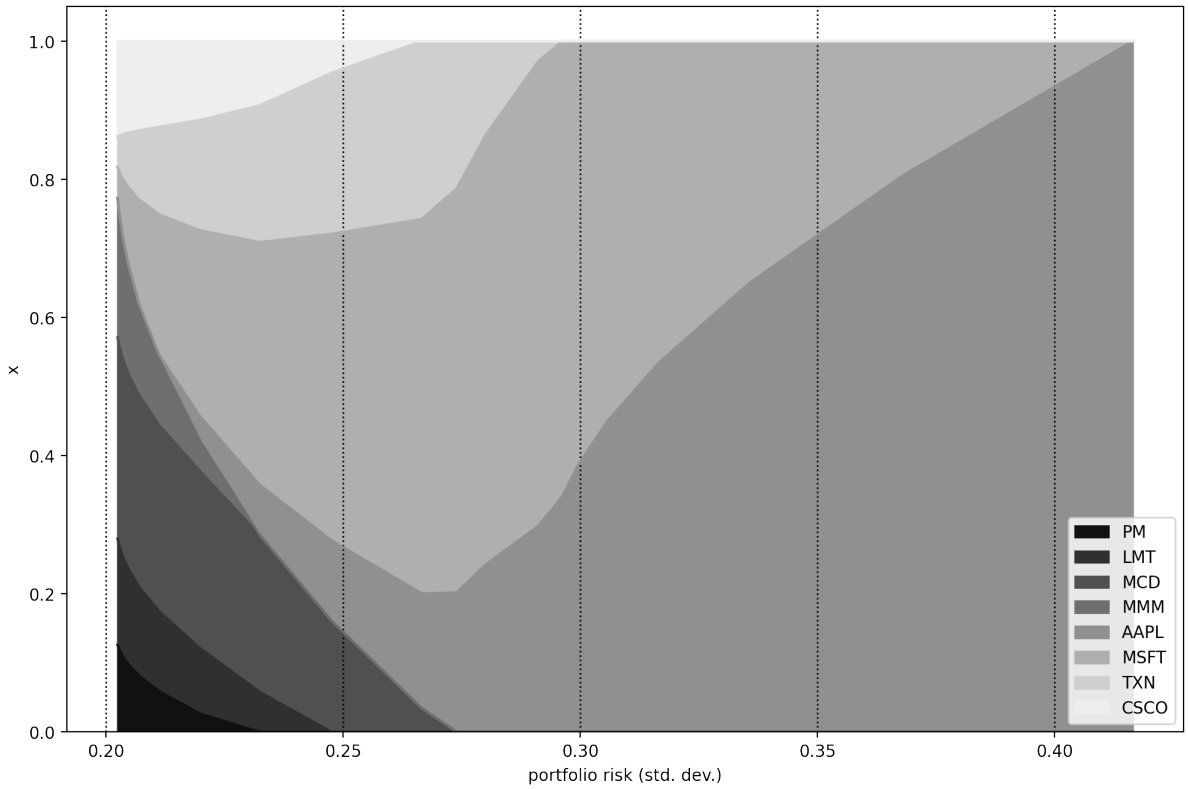

Fig. 5.2: Portfolio composition $\mathbf{x}$ with varying level if risk-aversion δ.

5.5.2 Large scale factor model

In this example we will generate Monte Carlo based return data and compare optimization runtime between the Cholesky decomposition based and factor model based representation of the covariance matrix.

The following function generates the data, and the factor model:

```
def random_factor_model(N, K, T):
    # Generate K uncorrelated, zero mean factors, with weighted variance
    S_F = np.diag(range(1, K + 1))
    rng = np.random.default_rng(seed=1)
    Z_F = rng.multivariate_normal(np.zeros(K), S_F, T).T

    # Generate random factor model parameters
    B = rng.normal(size=(N, K))
    a = rng.normal(loc=1, size=(N, 1))
    e = rng.multivariate_normal(np.zeros(N), np.eye(N), T).T

    # Generate N time-series from the factors
    Z = a + B @ Z_F + e

    # Residual covariance
    S_theta = np.cov(e)
    diag_S_theta = np.diag(S_theta)
```

(continues on next page)

(continued from previous page)

```
    # Optimization parameters
    m = np.mean(Z, axis=1)
    S = np.cov(Z)

    return m, S, B, S_F, diag_S_theta
```

The following code computes the comparison by increasing the number of securities N. The factor model in this example will have $K = 10$ common factors, which is kept constant, because it typically does not depend on N.

```
# Risk limit
gamma2 = 0.1

# Number of factors
K = 10

# Generate runtime data
list_runtimes_orig = []
list_runtimes_factor = []
for n in range(5, 15):
    N = 2**n
    T = N + 2**(n-1)
    m, S, B, S_F, diag_S_theta = random_factor_model(N, K, T)

    F = np.linalg.cholesky(S_F)
    G_factor = B @ F
    g_specific = np.sqrt(diag_S_theta)

    G_orig = np.linalg.cholesky(S)

    optimum_orig, runtime_orig = Markowitz(N, m, G_orig, gamma2)
    optimum_factor, runtime_factor = \
        Markowitz(N, m, (G_factor, g_specific), gamma2)
    list_runtimes_orig.append((N, runtime_orig))
    list_runtimes_factor.append((N, runtime_factor))

tup_N_orig, tup_time_orig = list(zip(*list_runtimes_orig))
tup_N_factor, tup_time_factor = list(zip(*list_runtimes_factor))
```

The function call `Markowitz(N, m, G, gamma2)` runs the Fusion model, the same way as in Sec. 2.4. If instead of the argument `G` we specify the tuple `(G_factor, g_specific)`, then the risk constraint in the Fusion model is created such that it exploits the risk factor structure:

```
# M is the Fusion model object
factor_risk = G_factor.T @ x
specific_risk = Expr.mulElm(g_specific, x)
total_risk = Expr.vstack(factor_risk, specific_risk)

M.constraint('risk', Expr.vstack(gamma2**0.5, total_risk),
```

(continues on next page)

(continued from previous page)

```
                    Domain.inQCone())
```

To get the runtime of the optimization, we add the following line to the model:

```
time = M.getSolverDoubleInfo("optimizerTime")
```

The results can be seen on Fig. 5.3, showing that the factor model based covariance matrix modeling can result in orders of magnitude faster solution times for large scale problems.

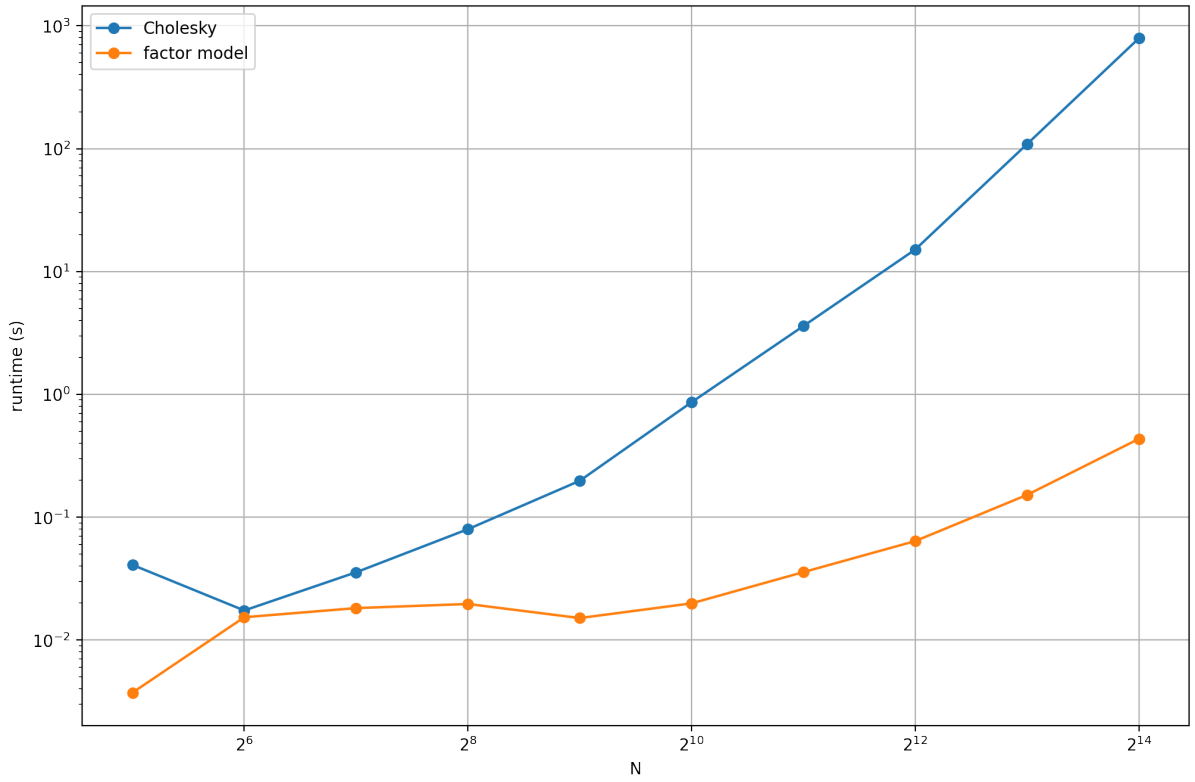

Fig. 5.3: Runtimes with and without using a factor model.

Chapter 6

Transaction costs

Rebalancing a portfolio generates turnover, i. e., buying and selling of securities to change the portfolio composition. The basic Markowitz model assumes that there are no costs associated with trading, but in reality, turnover incurs expenses. In this chapter we extend the basic model to take this into account in the form of transaction cost constraints. We also show some practical constraints, which can also limit turnover through limiting position sizes.

We can classify transaction costs into two types [WO11]:

- *Fixed costs* are independent of transaction volume. These include brokerage commissions and transfer fees.
- *Variable costs* depend on the transaction volume. These comprise execution costs such as market impact, bid/ask spread, or slippage; and opportunity costs of failed or incomplete execution.

Note that to be able to compare transaction costs with returns and risk, we need to aggregate them over the length of the investment time period.

In the optimization problem, let $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ denote the change in the portfolio with respect to the initial holdings $\mathbf{x}_0$. Then in general we can take into account transaction costs with the function C, where $C(\tilde{\mathbf{x}})$ is the total transaction cost incurred by the change $\tilde{\mathbf{x}}$ in the portfolio. Here we assume that transaction costs are separable, i.e., the total cost is the sum of the costs associated with each security: $C(\tilde{\mathbf{x}}) = \sum_{i=1}^{n} C_i(\tilde{x}_i)$, where the function $C_j(\tilde{x}_i)$ specifies the transaction costs incurred for the change in the holdings of security i. We can then write the MVO model with transaction cost in the following way:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} + \sum_{i=1}^{n} C_i(\tilde{x}_i) & = & \mathbf{1}^\mathsf{T}\mathbf{x}_0, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{6.1}$$

The constraint $\mathbf{1}^\mathsf{T}\mathbf{x} + \sum_{i=1}^{n} C_i(\tilde{x}_i) = \mathbf{1}^\mathsf{T}\mathbf{x}_0$ expresses the *self-financing* property of the portfolio. This means that no external cash is put into or taken out of the portfolio, we pay the costs from the existing portfolio components. We can e. g. assign one of the securities to be a cash account.

6.1 Variable transaction costs

The simplest model that handles variable costs makes the assumption that costs grow linearly with the trading volume [BBD+17, LMFB07]. We can use linear costs, for example, to model the cost related to the bid/ask spread, slippage, borrowing or shorting cost, or fund management fees. Let the transaction cost function for security i be given by

$$C_i(\tilde{x}_i) = \begin{cases} v_i^+ \tilde{x}_i, & \tilde{x}_i \geq 0, \\ -v_i^- \tilde{x}_i, & \tilde{x}_i < 0, \end{cases}$$

where v_i^+ and v_i^- are the cost rates associated with buying and selling security i. By introducing positive and negative part variables $\tilde{x}_i^+ = \max(\tilde{x}_i, 0)$ and $\tilde{x}_i^- = \max(-\tilde{x}_i, 0)$ we can linearize this constraint to $C_i(\tilde{x}_i) = v_i^+ \tilde{x}_i^+ + v_i^- \tilde{x}_i^-$. We can handle any piecewise linear convex transaction cost function in a similar way. After modeling the variables $\tilde{x}_i^+$ and $\tilde{x}_i^-$ as in Sec. 13.1.1, the optimization problem will then become

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} + \langle \mathbf{v}^+, \tilde{\mathbf{x}}^+ \rangle + \langle \mathbf{v}^-, \tilde{\mathbf{x}}^- \rangle & = & 1, \\ & \tilde{\mathbf{x}} & = & \tilde{\mathbf{x}}^+ - \tilde{\mathbf{x}}^-, \\ & \tilde{\mathbf{x}}^+, \tilde{\mathbf{x}}^- & \geq & 0, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{6.2}$$

In this model the budget constraint ensures that the variables $\tilde{\mathbf{x}}^+$ and $\tilde{\mathbf{x}}^-$ will not both become positive in any optimal solution.

6.2 Fixed transaction costs

We can extend the previous model with fixed transaction costs. Considering fixed costs is a way to discourage trading very small amounts, thus obtaining a sparse portfolio vector, i. e., one that has many zero entries.

Let f_i^+ and f_i^- be the fixed costs associated with buying and selling security i. The extended transaction cost function is given by

$$C_i(\tilde{x}_i) = \begin{cases} 0, & \tilde{x}_i = 0, \\ f_i^+ + v_i^+ \tilde{x}_i, & \tilde{x}_i > 0, \\ f_i^- - v_i^- \tilde{x}_i, & \tilde{x}_i < 0. \end{cases}$$

This function is not convex, but we can still formulate a mixed integer optimization problem based on Sec. 13.2.1 by introducing new variables. Let $\mathbf{y}^+$ and $\mathbf{y}^-$ be binary

vectors. Then the optimization problem with transaction costs will become

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} + \langle \mathbf{f}^+, \mathbf{y}^+ \rangle + \langle \mathbf{f}^-, \mathbf{y}^- \rangle + & & \\ & +\langle \mathbf{v}^+, \tilde{\mathbf{x}}^+ \rangle + \langle \mathbf{v}^-, \tilde{\mathbf{x}}^- \rangle & = & 1, \\ & \tilde{\mathbf{x}} & = & \tilde{\mathbf{x}}^+ - \tilde{\mathbf{x}}^-, \\ & \tilde{\mathbf{x}}^+, \tilde{\mathbf{x}}^- & \geq & 0, \\ & \tilde{\mathbf{x}}^+ & \leq & \mathbf{u}^+ \circ \mathbf{y}^+, \\ & \tilde{\mathbf{x}}^- & \leq & \mathbf{u}^- \circ \mathbf{y}^-, \\ & \mathbf{y}^+ + \mathbf{y}^- & \leq & \mathbf{1}, \\ & \mathbf{y}^+, \mathbf{y}^- & \in & \{0,1\}^N, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{6.3}$$

where $\mathbf{u}^+$ and $\mathbf{u}^-$ are vectors of upper bounds on the amounts of buying and selling in each security and $\circ$ is the elementwise product. The products $u_i^+ y_i^+$ and $u_i^- y_i^-$ ensure that if security i is traded ($y_i^+ = 1$ or $y_i^- = 1$), then both fixed and variable costs are incurred, otherwise ($y_i^+ = y_i^- = 0$) the transaction cost is zero. Finally, the constraint $\mathbf{y}^+ + \mathbf{y}^- \leq \mathbf{1}$ ensures that the transaction for each security is either a buy or a sell, and never both.

6.3 Market impact costs

In reality, each trade alters the price of the security. This effect is called *market impact*. If the traded quantity is small, the impact is negligible and we can assume that the security prices are independent of the amounts traded. However, for large traded volumes we should take market impact into account.

While there is no standard model for market impact, in practice an empirical power law is applied [GK00] [p. 452]. Let $\tilde{d}_i = d_i - d_{0,i}$ be the traded dollar amount for security i. Then the average relative price change is

$$\frac{\Delta p_i}{p_i} = \pm c_i \sigma_i \left(\frac{|\tilde{d}_i|}{q_i} \right)^{\beta - 1}, \tag{6.4}$$

where σ_i is the volatility of security i for a unit time period, q_i is the average dollar volume in a unit time period, and the sign depends on the direction of the trade. The number c_i has to be calibrated, but it is usually around one. Equation (6.4) is called the "square-root" law, because $\beta - 1$ is empirically shown to be around $1/2$ [TLD+11].

The relative price difference (6.4) is the impact cost rate, assuming $\tilde{d}_i$ dollar amount is traded. After actually trading this amount, we get the total market impact cost as

$$C_i(\tilde{d}_i) = \frac{\Delta p_i}{p_i} \tilde{d}_i = a_i |\tilde{d}_i|^\beta, \tag{6.5}$$

where $a_i = \pm c_i \sigma_i / q_i^{\beta - 1}$. Thus if $\beta - 1 = 1/2$, the market impact cost increases with $\beta = 3/2$ power of the traded dollar amount.

We can also express the market impact cost in terms of portfolio fraction $\tilde{x}_i$ instead of $\tilde{d}_i$ by normalizing q_i with the total portfolio value $\mathbf{v}^\mathsf{T}\mathbf{p}_0$.

Using Sec. 13.1.1 we can model $t_i \geq |\tilde{x}_i|^\beta$ with the power cone as $(t_i, 1, \tilde{x}_i) \in \mathcal{P}_3^{1/\beta,(\beta-1)/\beta}$. Hence, it follows that the total market impact cost term $\sum_{i=1}^N a_i|\tilde{x}_i|^\beta$ can be modeled by $\sum_{i=1}^N a_i t_i$ under the constraint $(t_i, 1, \tilde{x}_i) \in \mathcal{P}_3^{1/\beta,(\beta-1)/\beta}$.

Note however, that in this model nothing forces t_i to be small as possible to ensure $t_i = |\tilde{x}_i|^\beta$ holds at the optimal solution. This freedom allows the optimizer to try reducing portfolio risk by incorrectly treating $a_i t_i$ as a risk-free security. Then it would allocate more weight to $a_i t_i$ while reducing weight allocated to risky securities, basically throwing away money.

There are two solutions, which can prevent this unwanted behavior:

- Adding a penalty term $-\delta^\mathsf{T}\mathbf{t}$ to the objective function to prevent excess growth of the variables t_i. We have to calibrate the hyper-parameter vector δ so that the penalty would not become too dominant.
- Adding a risk-free security to the model. In this case the optimizer will prefer to allocate to the risk-free security, which has positive return (the risk-free rate), instead of allocating to $a_i t_i$.

Let us denote the weight of the risk-free security by x^f and the risk-free rate of return by r^f. Then the portfolio optimization problem accounting for market impact costs will be

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} + r^\mathrm{f}x^\mathrm{f} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} + \mathbf{a}^\mathsf{T}\mathbf{t} + x^\mathrm{f} & = & 1, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & (t_i, 1, \tilde{x}_i) & \in & \mathcal{P}_3^{1/\beta,(\beta-1)/\beta},\ i = 1, \ldots, N, \\ & \mathbf{x}, x^\mathrm{f} & \in & \mathcal{F}. \end{array} \tag{6.6}$$

Note that if we model using the quadratic cone instead of the rotated quadratic cone and a risk free security is present, then there will be no optimal portfolios for which $0 < x^\mathrm{f} < 1$. The solutions will be either $x^\mathrm{f} = 1$ or some risky portfolio with $x^\mathrm{f} = 0$. See a detailed discussion about this in Sec. 13.3.

6.4 Cardinality constraints

Investors often prefer portfolios with a limited number of securities. We do not need to use all of the N securities to achieve good diversification, and this way we can also reduce costs significantly. We can create explicit limits to constrain the number of securities.

Suppose that we allow at most K coordinates of the difference vector $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ to be non-zero, where K is (much) smaller than the total number of securities N.

We can again model this type of constraint based on Sec. 13.2.1 by introducing a binary vector $\mathbf{y}$ to indicate $|\tilde{\mathbf{x}}| \neq \mathbf{0}$, and by bounding the sum of $\mathbf{y}$. The basic Markowitz

model then gets updated as follows:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \tilde{\mathbf{x}} & = & \mathbf{x}-\mathbf{x}_0, \\ & \tilde{\mathbf{x}} & \leq & \mathbf{u}\circ\mathbf{y}, \\ & \tilde{\mathbf{x}} & \geq & -\mathbf{u}\circ\mathbf{y}, \\ & \mathbf{1}^\mathsf{T}\mathbf{y} & \leq & K, \\ & \mathbf{y} & \in & \{0,1\}^N, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{6.7}$$

where the vector $\mathbf{u}$ is some a priori chosen upper bound on the amount of trading in each security.

6.5 Buy-in threshold

In the above examples we assumed that trades can be arbitrarily small. In reality, however, it can be meaningful to place lower bounds on traded amounts to avoid unrealistically small trades and to control the transaction cost. These constraints are called *buy-in threshold*.

Let $\tilde{\mathbf{x}} = \mathbf{x}-\mathbf{x}_0$ be the traded amount. Let also $\tilde{\mathbf{x}}^+ = \max(\tilde{\mathbf{x}}, 0)$ and $\tilde{\mathbf{x}}^- = \max(-\tilde{\mathbf{x}}, 0)$ be the positive and negative part of $\tilde{\mathbf{x}}$. These we model according to Sec. 13.2.1. Then the buy-in threshold basically means that $\tilde{\mathbf{x}}^\pm \in \{0\} \cup [\ell^\pm, \mathbf{u}^\pm]$, where $\ell^\pm$ and $\mathbf{u}^\pm$ are vectors of lower and upper bounds on $\tilde{\mathbf{x}}^+$ and $\tilde{\mathbf{x}}^-$ respectively.

This is a semi-continuous variable, which we can model based on Sec. 13.2.1. We introduce binary variables $\mathbf{y}^\pm$ and constraints $\ell^\pm \circ \mathbf{y}^\pm \leq \tilde{\mathbf{x}}^\pm \leq \mathbf{u}^\pm \circ \mathbf{y}^\pm$. The optimization problem would then become a mixed integer problem of the form

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \mathbf{x}-\mathbf{x}_0 & = & \tilde{\mathbf{x}}^+ - \tilde{\mathbf{x}}^-, \\ & \tilde{\mathbf{x}}^+, \tilde{\mathbf{x}}^- & \geq & 0, \\ & \tilde{\mathbf{x}}^+ & \leq & \mathbf{u}^+ \circ \mathbf{y}^+, \\ & \tilde{\mathbf{x}}^+ & \geq & \ell^+ \circ \mathbf{y}^+, \\ & \tilde{\mathbf{x}}^- & \leq & \mathbf{u}^- \circ \mathbf{y}^-, \\ & \tilde{\mathbf{x}}^- & \geq & \ell^- \circ \mathbf{y}^-, \\ & \mathbf{y}^+ + \mathbf{y}^- & \leq & \mathbf{1}, \\ & \mathbf{y}^+, \mathbf{y}^- & \in & \{0,1\}^N, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{6.8}$$

This model is of course compatible with the fixed plus linear transaction cost model discussed in Sec. 6.2.

6.6 Example

In this chapter we show two examples. The first demonstrates the modeling of market impact through the use of the power cone, while the second example presents fixed and variable transaction costs and the buy-in threshold.

6.6.1 Market impact model

As a starting point, we refer back to problem (2.13). We will extend this problem with the market impact cost model. To compute the coefficients a_i in formula (6.5), we assume that daily volume data is also available in the dataframe `df_volumes`. We also compute the mean of the daily volumes, and the daily volatility for each security as the standard deviation of daily linear returns:

```
# Compute average daily volume and daily volatility (std. dev.)
df_lin_returns = df_prices.pct_change()
vty = df_lin_returns.std()
vol = (df_volumes * df_prices).mean()
```

According to the data, the average daily dollar volumes are $10^8 \cdot [3.9883, 4.2416, 6.0054, 4.2584, 30.4647, 34.5619, 5.0077, 8.4950]$, and the daily volatilities are $[0.0164, 0.0154, 0.0146, 0.0155, 0.0191, 0.0173, 0.0186, 0.0169]$. Thus in this example we will choose the size of our portfolio to be 10 billion dollars so that we can see a significant market impact.

Then we update the Fusion model introduced in Sec. 2.4.2 with new variables and constraints:

```
def EfficientFrontier(N, m, G, deltas, a, beta, rf):
   with Model("MarketImpact") as M:
      # Variables
      # The variable x is the fraction of holdings in each security.
      # x must be positive, this imposes the no short-selling constraint.
      x = M.variable("x", N, Domain.greaterThan(0.0))

      # Variable for risk-free security (cash account)
      xf = M.variable("xf", 1, Domain.greaterThan(0.0))

      # The variable s models the portfolio variance term in the objective.
      s = M.variable("s", 1, Domain.unbounded())

      # Auxiliary variable to model market impact
      t = M.variable("t", N, Domain.unbounded())

      # Budget constraint with transaction cost terms
      M.constraint('budget', Expr.sum(x) + xf + t.T @ a == 1.0)

      # Power cone to model market impact
      M.constraint('mkt_impact', Expr.hstack(t, Expr.constTerm(N, 1.0), x),
                   Domain.inPPowerCone(1.0 / beta))
```

(continues on next page)

(continued from previous page)

```
    # Objective (quadratic utility version)
    delta = M.parameter()
    M.objective('obj', ObjectiveSense.Maximize,
                       x.T @ m + rf * xf - delta * s)

    # Conic constraint for the portfolio variance
    M.constraint('risk', Expr.vstack(s, 1, G.T @ x),
                         Domain.inRotatedQCone())

    columns = ["delta", "obj", "return", "risk", "t_resid",
               "x_sum", "xf", "tcost"] + df_prices.columns.tolist()
    df_result = pd.DataFrame(columns=columns)
    for d in deltas:
        # Update parameter
        delta.setValue(d)

        # Solve optimization
        M.solve()

        # Save results
        portfolio_return = m @ x.level() + np.array([rf]) @ xf.level()
        portfolio_risk = np.sqrt(2 * s.level()[0])
        t_resid = t.level() - np.abs(x.level())**beta
        row = pd.Series([d, M.primalObjValue(), portfolio_return,
                         portfolio_risk, sum(t_resid), sum(x.level()),
                         sum(xf.level()), t.level() @ a] + \
                        list(x.level()), index=columns)
        df_result = df_result.append(row, ignore_index=True)

    return df_result
```

The new rows are:

- The row for the variable x^{f}, which represents the weight allocated to the cash account. The annual return on it is assumed to be $r^{\mathrm{f}} = 1\%$. We constrain x^{f} to be positive, meaning that borrowing money is not allowed.
- The row for the auxiliary variable $\mathbf{t}$.
- The row for the market impact constraint modeled using the power cone.

We modified the budget constraints to include x^{f} and the market impact cost $\mathbf{a}^{\mathsf{T}}\mathbf{t}$. The objective also contains the risk-free part of portfolio return $r^{\mathrm{f}}x^{\mathrm{f}}$.

In this example, we start with 100% cash, meaning that $x^{\mathrm{f}}_0 = 1$ and $\mathbf{x}_0 = \mathbf{0}$. Transaction cost is thus incurred for the total weight $\mathbf{x}$.

Next, we compute the efficient frontier with and without market impact costs. We select $\beta = 3/2$ and $c_i = 1$. The following code produces the results:

```
deltas = np.logspace(start=-0.5, stop=2, num=20)[::-1]
portfolio_value = 10**10
```

(continues on next page)

(continued from previous page)

```
rel_vol = vol / portfolio_value
a1 = np.zeros(N)
a2 = (c * vty / rel_vol**(beta - 1)).to_numpy()
ax = plt.gca()
for a in [a1, a2]:
    df_result = EfficientFrontier(N, m, G, deltas, a, beta, rf)
    mask = df_result < 0
    mask.iloc[:, :2] = False
    df_result[mask] = 0
    df_result.plot(ax=ax, x="risk", y="return", style="-o",
                   xlabel="portfolio risk (std. dev.)",
                   ylabel="portfolio return", grid=True)
ax.legend(["return without price impact", "return with price impact"])
```

On Fig. 6.1 we can see the return reducing effect of market impact costs. The left part of the efficient frontier (up to the so called *tangency portfolio*) is linear because a risk-free security was included. However, in this case borrowing is not allowed, so the right part remains the usual parabola shape.

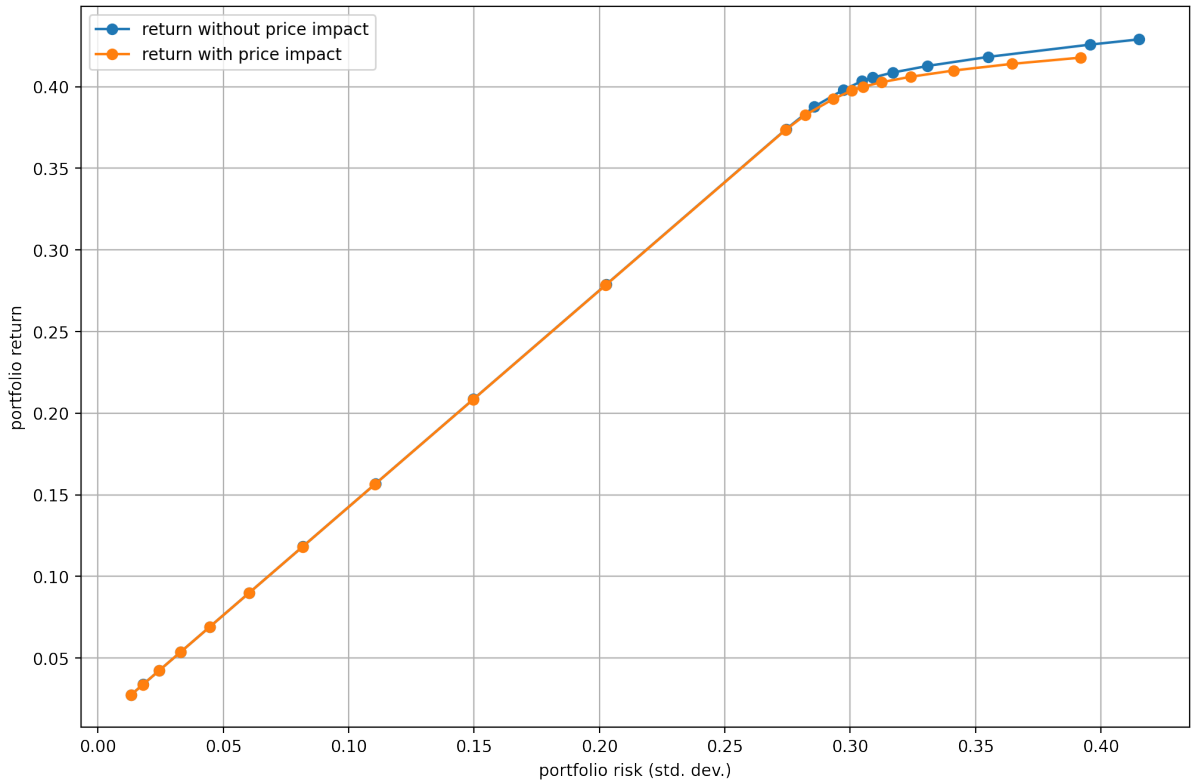

Fig. 6.1: The efficient frontier with risk-free security included, and market impact cost taken into account.

6.6.2 Transaction cost models

In this example we show a problem that models fixed and variable transaction costs and the buy-in threshold. Note that we do not model the market impact here.

We will assume now that $\mathbf{x}$ can take negative values too (short-selling is allowed), up to the limit of 30% portfolio size. This way we can see how to apply different costs to buy and sell trades. We also assume that $\mathbf{x}_0 = \mathbf{0}$, so $\tilde{\mathbf{x}} = \mathbf{x}$.

The following code defines variables used as the positive and negative part variables of $\mathbf{x}$ and the binary variables $\mathbf{y}^+, \mathbf{y}^-$ indicating whether there is buying or selling in a security:

```
# Real variables
xp = M.variable("xp", N, Domain.greaterThan(0.0))
xm = M.variable("xm", N, Domain.greaterThan(0.0))

# Binary variables
yp = M.variable("yp", N, Domain.binary())
ym = M.variable("ym", N, Domain.binary())
```

Next we add two constraints. The first links `xp` and `xm` to `x`, so that they represent the positive and negative parts. The second ensures that for each coordinate of `yp` and `ym` only one of the values can be 1.

```
# Constraint assigning xp and xm to the positive and negative part of x.
M.constraint('pos-neg-part', x == xp - xm)

# Exclusive buy-sell constraint
M.constraint('exclusion', yp + ym <= 1.0)
```

We update the budget constraint with the variable and fixed transaction cost terms. The fixed cost of buy and sell trades are held by the variables `fp` and `fm`. These are typically given in dollars, and have to be divided by the total portfolio value. The variable cost coefficients are `vp` and `vm`. If these are given as percentages, then we do not have to modify them.

```
# Budget constraint with transaction cost terms
fixcost_terms = yp.T @ fp + ym.T @ fm
varcost_terms = xp.T @ vp + xm.T @ vm
budget_terms = Expr.sum(x) + varcost_terms + fixcost_terms
M.constraint('budget', budget_terms == 1.0)
```

Next, the 130/30 leverage constraint is added. Note that the transaction cost terms from the budget constraint should also appear here, otherwise the two constraints combined would allow a little more leverage than intended. (The sum of $\mathbf{x}$ would not reach 1 because of the cost terms, leaving more space in the leverage constraint for negative positions.)

```
# Auxiliary variable for 130/30 leverage constraint
z = M.variable("z", N, Domain.unbounded())

# 130/30 leverage constraint
M.constraint('leverage-gt', z >= x)
```

(continues on next page)

(continued from previous page)

```
M.constraint('leverage-ls', z >= -x)
M.constraint('leverage-sum',
             Expr.sum(z) + varcost_terms + fixcost_terms == 1.6)
```

Finally, to be able to differentiate between zero allocation (not incurring fixed cost) and nonzero allocation (incurring fixed cost), and to implement buy-in threshold, we need bound constraint involving the binary variables:

```
# Bound constraints
M.constraint('ubound-p', xp <= up * yp)
M.constraint('ubound-m', xm <= um * ym)
M.constraint('lbound-p', xp >= lp * yp)
M.constraint('lbound-m', xm >= lm * ym)
```

The full updated model will then look like the following:

```
def EfficientFrontier(N, m, G, deltas, vp, vm, fp, fm, up, um,
                      lp, lm, pcoef):
    with Model("TransactionCost") as M:
        # Real variables
        # The variable x is the fraction of holdings in each security.
        x = M.variable("x", N, Domain.unbounded())
        xp = M.variable("xp", N, Domain.greaterThan(0.0))
        xm = M.variable("xm", N, Domain.greaterThan(0.0))

        # Binary variables
        yp = M.variable("yp", N, Domain.binary())
        ym = M.variable("ym", N, Domain.binary())

        # Constraint assigning xp and xm to the pos. and neg. part of x.
        M.constraint('pos-neg-part', x == xp - xm)

        # s models the portfolio variance term in the objective.
        s = M.variable("s", 1, Domain.unbounded())

        # Auxiliary variable for 130/30 leverage constraint
        z = M.variable("z", N, Domain.unbounded())

        # Bound constraints
        M.constraint('ubound-p', xp <= up * yp)
        M.constraint('ubound-m', xm <= um * ym)
        M.constraint('lbound-p', xp >= lp * yp)
        M.constraint('lbound-m', xm >= lm * ym)

        # Exclusive buy-sell constraint
        M.constraint('exclusion', yp + ym <= 1.0)

        # Budget constraint with transaction cost terms
        fixcost_terms = yp.T @ fp + ym.T @ fm
        varcost_terms = xp.T @ vp + xm.T @ vm
```

(continues on next page)

(continued from previous page)

```
        budget_terms = Expr.sum(x) + varcost_terms + fixcost_terms
        M.constraint('budget', budget_terms == 1.0)

        # 130/30 leverage constraint
        M.constraint('leverage-gt', z >= x)
        M.constraint('leverage-ls', z >= -x)
        M.constraint('leverage-sum',
                     Expr.sum(z) + varcost_terms + fixcost_terms == 1.6)

        # Objective (quadratic utility version)
        delta = M.parameter()
        penalty = pcoef * Expr.sum(xp + xm)
        M.objective('obj', ObjectiveSense.Maximize,
                           x.T @ m - penalty - delta * s)

        # Conic constraint for the portfolio variance
        M.constraint('risk', Expr.vstack(s, 1, G.T @ x),
                             Domain.inRotatedQCone())

        columns = ["delta", "obj", "return", "risk", "x_sum", "tcost"] + \
                  df_prices.columns.tolist()
        df_result = pd.DataFrame(columns=columns)
        for idx, d in enumerate(deltas):
            # Update parameter
            delta.setValue(d)

            # Solve optimization
            M.solve()

            # Save results
            portfolio_return = m @ x.level()
            portfolio_risk = np.sqrt(2 * s.level()[0])
            tcost = np.dot(vp, xp.level()) + np.dot(vm, xm.level()) + \
                    np.dot(fp, yp.level()) + np.dot(fm, ym.level())
            row = pd.Series([d, M.primalObjValue(), portfolio_return,
                             portfolio_risk, sum(x.level()), tcost] + \
                            list(x.level()), index=columns)
            df_result = df_result.append(row, ignore_index=True)

        return df_result
```

Here we also used a penalty term in the objective to prevent excess growth of the positive part and negative part variables. The coefficient of the penalty has to be calibrated so that we do not overpenalize.

We also have to mention that because of the binary variables, we can only solve this as a mixed integer optimization (MIO) problem. The solution of such a problem might not be as efficient as the solution of a problem with only continuous variables. See Sec. 13.2 for details regarding MIO problems.

We compute the efficient frontier with and without transaction costs. The following code produces the results:

```
deltas = np.logspace(start=-0.5, stop=2, num=20)[::-1]
ax = plt.gca()
for a in [0, 1]:
    pcoef = a * 0.03
    fp = a * 0.005 * np.ones(N) # Depends on portfolio value
    fm = a * 0.01 * np.ones(N) # Depends on portfolio value
    vp = a * 0.01 * np.ones(N)
    vm = a * 0.02 * np.ones(N)
    up = 2.0
    um = 2.0
    lp = a * 0.05
    lm = a * 0.05

    df_result = EfficientFrontier(N, m, G, deltas, vp, vm, fp, fm, up, um,
                                  lp, lm, pcoef)
    df_result.plot(ax=ax, x="risk", y="return", style="-o",
                   xlabel="portfolio risk (std. dev.)",
                   ylabel="portfolio return", grid=True)
ax.legend(["return without transaction cost",
           "return with transaction cost"])
```

On Fig. 6.2 we can see the return reducing effect of transaction costs. The overall return is higher because of the leverage.

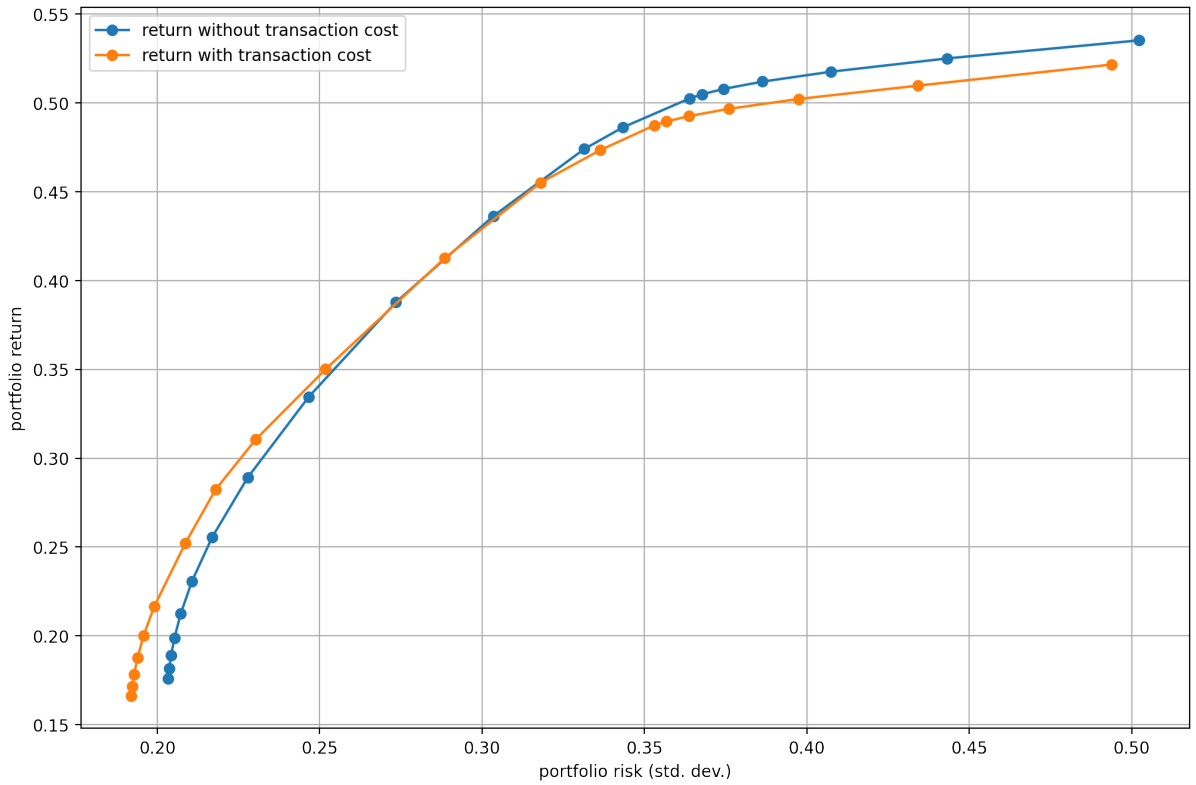

Fig. 6.2: The efficient frontier with transaction costs taken into account.

Chapter 7

Benchmark relative portfolio optimization

We often measure the performance of portfolios relative to a benchmark. The benchmark is typically a market *index* representing an asset class or some segment of the market, and its composition is assumed known. Benchmark relative portfolio management can have two types of strategies: *active* or *passive*. The passive strategy tries to replicate the benchmark, i. e. match its performance as closely as possible. In contrast, active management seeks to consistently beat (outperform) the benchmark (after management fees). This section is based on [CJPT18].

7.1 Active return

Let the portfolio and benchmark weights be $\mathbf{x}$ and $\mathbf{x}_{\text{bm}}$. In active management, quantities of interest are the return and the risk relative to the benchmark. The *active portfolio return* (return above the benchmark return) is

$$R_{\mathbf{x}_\text{a}} = \mathbf{x}_\text{a}^\mathsf{T} R = \mathbf{x}^\mathsf{T} R - \mathbf{x}_\text{bm}^\mathsf{T} R = R_\mathbf{x} - R_{\mathbf{x}_\text{bm}},$$

where $\mathbf{x}_\text{a} = \mathbf{x} - \mathbf{x}_\text{bm}$ is the *active holdings* of the portfolio.

The *active portfolio risk* or *tracking error* will be the standard deviation of active return:

$$\sigma_{\mathbf{x}_\text{a}}(R_\mathbf{x}, R_{\mathbf{x}_\text{bm}}) = \sqrt{\mathbf{x}_\text{a}^\mathsf{T} \Sigma \mathbf{x}_\text{a}},$$

where Σ is the covariance matrix of return. The tracking error measures how close the portfolio return is to the benchmark return. Outperforming the benchmark involves additional risk; if the tracking error is zero then there cannot be any active return at all.

In general we can construct a class of functions for the purpose of measuring tracking error differently, from any deviation measure. See in Sec. 8.

7.2 Factor model on active returns

In the active return there is still a systematic component attributable to the benchmark. We can account for that using a single factor model. First we create a factor model for security returns:

$$R = \beta R_{\mathbf{x}_{\text{bm}}} + \theta,$$

where $\beta_i = \text{Cov}(R_i, R_{\mathbf{x}_{\text{bm}}})/\text{Var}(R_{\mathbf{x}_{\text{bm}}})$ is the exposure of security i to the benchmark, $\text{Cov}(\theta) = \mathbf{0}$, $\text{Cov}(R_{\mathbf{x}_{\text{bm}}}, \theta_i) = 0$, and $\mathbb{E}(\theta_i) = \alpha_i$ is the expected residual return. Under this model, the covariance matrix of R will be

$$\Sigma = \beta^{\mathsf{T}} \beta \sigma^2_{\mathbf{x}_{\text{bm}}} + \Sigma_\theta,$$

where $\sigma^2_{\mathbf{x}_{\text{bm}}}$ is the benchmark variance and Σ_θ is the residual covariance matrix.

This factor model allows us to decompose the portfolio return into a systematic component, which is explained by the benchmark, and a residual component, which is specific to the portfolio:

$$R_{\mathbf{x}} = \beta_{\mathbf{x}} R_{\mathbf{x}_{\text{bm}}} + \theta_{\mathbf{x}},$$

where $\beta_{\mathbf{x}} = \beta^{\mathsf{T}}\mathbf{x}$, $\theta_{\mathbf{x}} = \theta^{\mathsf{T}}\mathbf{x}$, and $\alpha_{\mathbf{x}} = \alpha^{\mathsf{T}}\mathbf{x} = \mathbb{E}(\theta_{\mathbf{x}})$.

Finally, by subtracting the benchmark return from both sides we can write the active return as:

$$R_{\mathbf{x}_{\text{a}}} = (\beta_{\mathbf{x}} - 1) R_{\mathbf{x}_{\text{bm}}} + \theta_{\mathbf{x}},$$

where $\beta_{\mathbf{x}} - 1$ is the *active beta*. After computing the variance of the decomposed active return, we can also write the square of the tracking error as

$$\sigma^2_{\mathbf{x}_{\text{a}}}(R_{\mathbf{x}}, R_{\mathbf{x}_{\text{bm}}}) = (\beta_{\mathbf{x}} - 1)^2 \sigma^2_{\mathbf{x}_{\text{bm}}} + \omega^2_{\mathbf{x}},$$

where $\sigma^2_{\mathbf{x}_{\text{bm}}} = \text{Var}(R_{\mathbf{x}_{\text{bm}}})$, and $\omega^2_{\mathbf{x}} = \text{Var}(\theta_{\mathbf{x}})$ is the residual risk.

We must note that if a different factor model is used to forecast risk and alpha (which is often the case in practice), there could be factors included in one model but not included in the other model. The optimizer will interpret such misalignments as factor return without risk or factor risk without return and it will exploit them as false opportunities, leading to unintended biases in the results. Risk and alpha factors are considered *aligned* if alpha can be written as a linear combination of the risk factors. In case of a misalignment, there are different approaches to mitigate it [KS13].

7.3 Optimization

An active investment strategy would try to gain higher portfolio alpha $\alpha_{\mathbf{x}}$ at the cost of accepting a higher tracking error. It follows that portfolio optimization with respect to a benchmark will optimize the tradeoff between portfolio alpha and the square of the tracking error. Additional constrains specific to such problems can be bounds on the

portfolio active beta or on the active holdings. We can see examples in the following model:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\alpha}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & (\mathbf{x}-\mathbf{x}_{\text{bm}})^\mathsf{T}\boldsymbol{\Sigma}(\mathbf{x}-\mathbf{x}_{\text{bm}}) & \leq & \gamma_{\text{TE}}^2, \\ & \mathbf{x}-\mathbf{x}_{\text{bm}} & \geq & \mathbf{l}_{\text{h}}, \\ & \mathbf{x}-\mathbf{x}_{\text{bm}} & \leq & \mathbf{u}_{\text{h}}, \\ & \boldsymbol{\beta}_{\mathbf{x}}-1 & \geq & \mathbf{l}_{\beta}, \\ & \boldsymbol{\beta}_{\mathbf{x}}-1 & \leq & \mathbf{u}_{\beta}, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{7.1}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are estimates of α and β. To compute these estimates, we can do a linear regression. However, this tends to overestimate the betas of stocks with high benchmark exposure and underestimate the betas of the stocks with low benchmark exposure. To improve the estimation, shrinkage towards one (the beta of the benchmark) can be helpful:

$$\boldsymbol{\beta}_{\text{shrunk}} = (1-c)\boldsymbol{\beta} + c\mathbf{1},$$

where we must estimate the optimal shrinkage factor c.

7.4 Extensions

7.4.1 Variance constraint

Optimization of alpha with constraining only the tracking error can increase total portfolio risk. According to [Jor04] it can be helpful to constrain also the total portfolio variance, especially in cases when the benchmark portfolio is relatively inefficient:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\alpha}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & (\mathbf{x}-\mathbf{x}_{\text{bm}})^\mathsf{T}\boldsymbol{\Sigma}(\mathbf{x}-\mathbf{x}_{\text{bm}}) & \leq & \gamma_{\text{TE}}^2, \\ & \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x} & \leq & \gamma^2, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{7.2}$$

7.4.2 Passive management

In the case of passive management, the goal of optimization is to construct a benchmark tracking portfolio, i. e., to have a tracking error as small as possible, given the constraints. This usually also means that $\alpha_{\mathbf{x}} = 0$. Of course without constraints or trading costs, the optimal solution is the benchmark index. Therefore optimization in passive management may only be useful when constraints are needed or liquidity issues are important [MM08].

7.4.3 Linear tracking error measures

The tracking error is one possible measure of closeness between the portfolio and the benchmark. However, there can be other measures [RWZ99]. For example, we can use the *mean absolute deviation* (MAD) measure, i. e., $\mathbb{E}(|R_{\mathbf{x}}-R_{\mathbf{x}_{\text{bm}}}|)$. Advantage of this measure is its better robustness against outliers in the data. Applying scenario approximation to the expectation we get $\frac{1}{T}\sum_{k=1}^{T}\left|\mathbf{r}_k^\mathsf{T}(\mathbf{x}-\mathbf{x}_{\text{bm}})\right|$. After modeling the absolute value function

(see Sec. 13.1.1), we can formulate the benchmark tracking problem with this measure as an LO model:

$$\begin{array}{lrcl} \text{maximize}_{\mathbf{x},\mathbf{t}} & \alpha^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \frac{1}{T}\sum_{k=1}^{T} t_k & \leq & \gamma, \\ & \mathbf{t} + \mathbf{R}^\mathsf{T}(\mathbf{x} - \mathbf{x}_\text{bm}) & \geq & 0, \\ & \mathbf{t} - \mathbf{R}^\mathsf{T}(\mathbf{x} - \mathbf{x}_\text{bm}) & \geq & 0, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{7.3}$$

where $\mathbf{R}$ is the return data matrix with one observation $\mathbf{r}_k,\ k = 1, \ldots, T$ in each column.

If the investor perceives risk as portfolio return being below the benchmark return, we can also use downside deviation measures. One example is the *lower partial moment* LPM_1 measure: $\mathbb{E}\left(\max(-R_\mathbf{x} + R_{\mathbf{x}_\text{bm}}, 0)\right)$. The scenario approximation of this expectation will be $\frac{1}{T}\sum_{k=1}^{T} \max\left(-\mathbf{r}_k^\mathsf{T}(\mathbf{x} - \mathbf{x}_\text{bm}), 0\right)$. After modeling the maximum (see Sec. 13.1.1) by defining a new variable $t_k^- = \max\left(-\mathbf{r}_k^\mathsf{T}(\mathbf{x} - \mathbf{x}_\text{bm}), 0\right)$, we can solve the problem as an LO:

$$\begin{array}{lrcl} \text{maximize} & \alpha^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & \frac{1}{T}\sum_{k=1}^{T} t_k^- & \leq & \gamma, \\ & \mathbf{t}^- & \geq & -\mathbf{R}^\mathsf{T}(\mathbf{x} - \mathbf{x}_\text{bm}), \\ & \mathbf{t}^- & \geq & 0, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{7.4}$$

Note that for a symmetric portfolio return distribution, this will be equivalent to the MAD model.

Linear models might be also preferable because of their more intuitive interpretation. By measuring the tracking error according to a linear function, the measurement unit of the objective function is percentage instead of squared percentage.

7.5 Example

Here we show an example of a benchmark relative optimization problem. The benchmark will be the equally weighted portfolio of the eight stocks from the previous examples, therefore $\mathbf{x}_\text{bm} = \mathbf{1}/N$. The benchmark is created by the following code by aggregating the price data of the eight stocks:

```
# Create benchmark
df_prices['bm'] = df_prices.iloc[:-2, 0:8].mean(axis=1)
```

Then we follow the same Monte Carlo procedure as in Sec. 5.5.1, just with the benchmark instead of the market factor. This will yield scenarios of linear returns on the investment time horizon of $h = 1$ year, so that we can compute estimates $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of α and β using time-series regression.

In the Fusion model, we make the following modifications:

- We define the active holdings variable $\mathbf{x}_\text{a} = \mathbf{x} - \mathbf{x}_\text{bm}$ by

  ```
  # Active holdings
  xa = x - xbm
  ```

- We modify the constraint on risk to be the constraint on tracking error:

```
# Conic constraint for the portfolio variance
M.constraint('risk', Expr.vstack(s, 1, G.T @ xa),
             Domain.inRotatedQCone())
```

- We also specify bounds on the active holdings and on the portfolio active beta:

```
# Constraint on active holdings
M.constraint('bound-h', xa, Domain.inRange(lh, uh))

# Constraint on portfolio active beta
port_act_beta = x.T @ B - 1
M.constraint('bound-b', port_act_beta, Domain.inRange(lb, ub))
```

- Finally, we modify the objective to maximize the portfolio alpha:

```
# Objective (quadratic utility version)
delta = M.parameter()
M.objective('obj', ObjectiveSense.Maximize, x.T @ a - delta * s)
```

The complete Fusion model of the optimization problem (7.1) will then be

```
def EfficientFrontier(N, a, B, G, xbm, deltas, uh, ub, lh, lb):
    with Model("Frontier") as M:
        # Variables
        # The variable x is the fraction of holdings in each security.
        # Non-negative, no short-selling
        x = M.variable("x", N, Domain.greaterThan(0.0))

        # Active holdings
        xa = x - xbm

        # Portfolio variance term in the objective.
        s = M.variable("s", 1, Domain.unbounded())

        # Budget constraint
        M.constraint('budget_x', Expr.sum(x) == 1.0)

        # Constraint on active holdings
        M.constraint('bound-h', xa, Domain.inRange(lh, uh))

        # Constraint on portfolio active beta
        port_act_beta = x.T @ B - 1
        M.constraint('bound-b', port_act_beta, Domain.inRange(lb, ub))

        # Conic constraint for the portfolio variance
        M.constraint('risk', Expr.vstack(s, 1, G.T @ xa),
                             Domain.inRotatedQCone())

        # Objective (quadratic utility version)
        delta = M.parameter()
```

(continues on next page)

(continued from previous page)

```
        M.objective('obj', ObjectiveSense.Maximize, x.T @ a - delta * s)

        # Create DataFrame to store the results. SPY is removed.
        columns = ["delta", "obj", "return", "risk"] + \
                  df_prices.columns[:-1].tolist()
        df_result = pd.DataFrame(columns=columns)
        for d in deltas:
            # Update parameter
            delta.setValue(d);

            # Solve optimization
            M.solve()

            # Save results
            portfolio_return = a @ x.level()
            portfolio_risk = np.sqrt(2 * s.level()[0])
            row = pd.Series([d, M.primalObjValue(), portfolio_return,
                             portfolio_risk] + \
                             list(x.level()), index=columns)
            df_result = pd.concat([df_result, row.to_frame().T],
                                  ignore_index=True)

        return df_result
```

We give the input parameters and compute the efficient frontier using the following code:

```
deltas = np.logspace(start=-0.5, stop=2, num=20)[::-1]
xbm = np.ones(N) / N
uh = np.ones(N) * 0.5
lh = -np.ones(N) * 0.5
ub = 0.5
lb = -0.5
df_result = EfficientFrontier(N, a, B, G, xbm, deltas, uh, ub, lh, lb)
mask = df_result < 0
mask.iloc[:, :2] = False
df_result[mask] = 0
df_result
```

On Fig. 7.1 we plot the efficient frontier and on Fig. 7.2 the portfolio composition. On the latter we see that as the tracking error decreases, the portfolio gets closer to the benchmark, i. e., the equal-weighted portfolio.

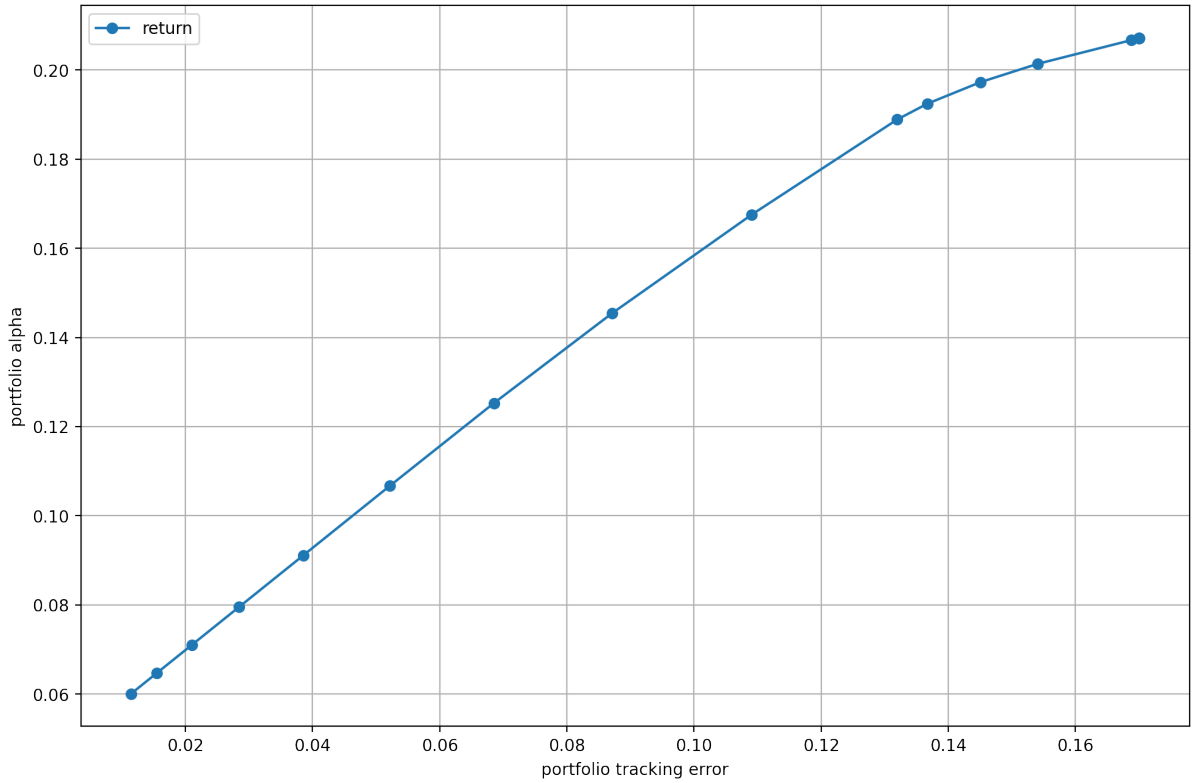

Fig. 7.1: The benchmark relative efficient frontier.

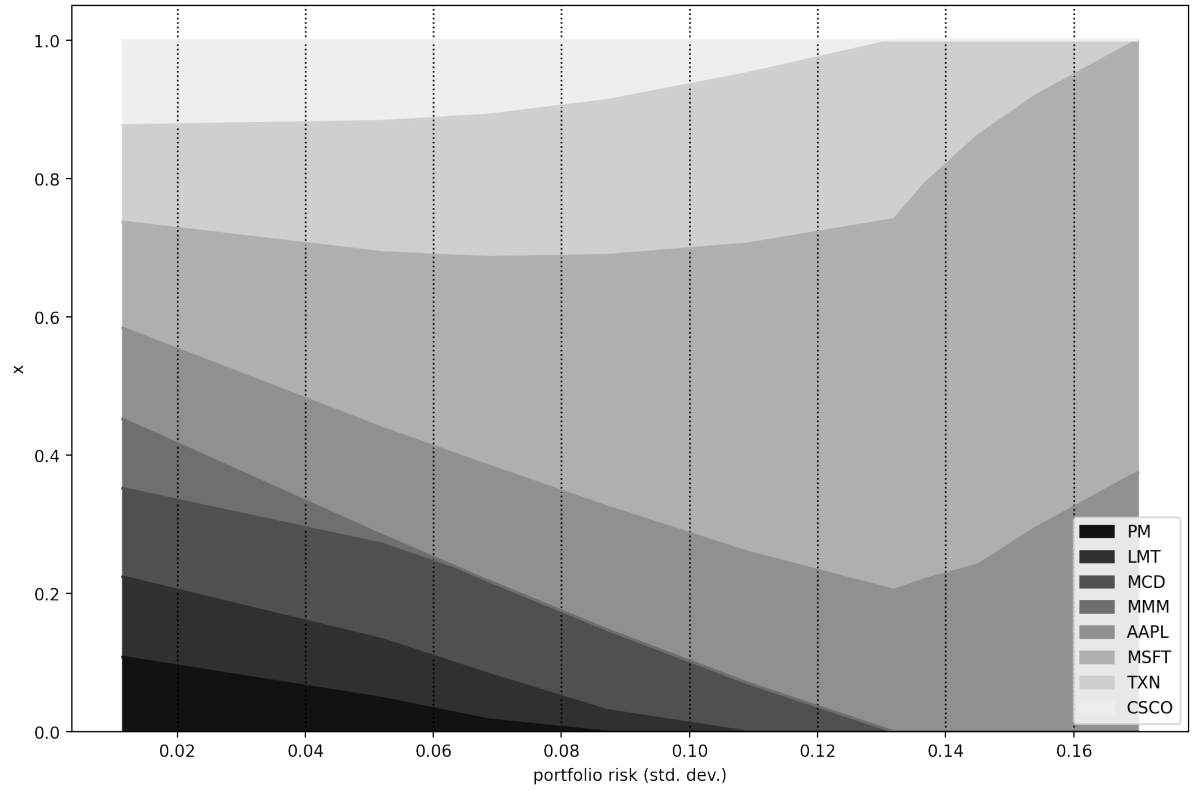

Fig. 7.2: Portfolio composition $\mathbf{x}$ with varying level if risk-aversion δ.

Chapter 8

Other risk measures

In the definitions (2.1)-(2.4) of the classical MVO problem the variance (or standard deviation) of portfolio return is used as the measure of risk, making computation easy and convenient. If we assume that the portfolio return is normally distributed, then the variance is in fact the optimal risk measure, because then MVO can take into account all information about return. Moreover, if T is large compared to N, the sample mean and covariance matrix are maximum likelihood estimates (MLE), implying that the optimal portfolio resulting from estimated inputs will also be a MLE of the true optimal portfolio [Lau01].

Empirical observations suggest however, that the distribution of linear return is often skewed, and even if it is elliptically symmetric, it can have fat tails and can exhibit tail dependence[1]. As the distribution moves away from normality, the performance of a variance based portfolio estimator can quickly degrade.

No perfect risk measure exists, though. Depending on the distribution of return, different measures can be more appropriate in different situations, capturing different characteristics of risk. Return of diversified equity portfolios are often approximately symmetric over periods of institutional interest. Options, swaps, hedge funds, and private equity have return distributions that are unlikely to be symmetric. Also less symmetric are distributions of fixed-income and real estate index returns, and diversified equity portfolios over long time horizons [MM08].

The measures presented here are expectations, meaning that optimizing them would lead to stochastic optimization problems in general. As a simplification, we consider only their sample average approximation by assuming that a set of linear return scenarios are given.

[1] Tail dependence between two random variables means that their correlation is greater for more extreme events. Typical in financial markets, where extreme events are likely to happen together for multiple securities.

8.1 Deviation measures

This class of measures quantify the variability around the mean of the return distribution, similar to variance. If we associate investment risk with the uncertainty of return, then such measures could be ideal.

Let X, Y be random variables of investment returns, and let D be a mapping. The general properties that deviation measures should satisfy are [RUZ06]:

1. Positivity: $D(X) \geq 0$ with $D(X) = 0$ only if $X = c \in \mathbb{R}$,
2. Translation invariance: $D(X + c) = D(X),\ c \in \mathbb{R}$,
3. Subadditivity: $D(X + Y) \leq D(X) + D(Y)$,
4. Positive homogeneity: $D(cX) = cD(X),\ c \geq 0$

Note that positive homegeneity and subadditivity imply the *convexity* of D.

8.1.1 Robust statistics

A group of deviation measures are robust statistics. Maximum likelihood estimators are highly sensitive to deviations from the assumed distribution. Thus if the return is not normally distributed, robust portfolio estimators can be an alternative. These are less efficient than MLE for normal distribution, but their performance degrades less quickly under departures from normality. One suitable class of estimators to express portfolio risk are M-estimators [VD09]:

$$D_\delta(R_\mathbf{x}) = \min_q \mathbb{E}\left[\delta(R_\mathbf{x} - q)\right],$$

where δ is a convex loss function with unique minimum at zero. It has the sample average approximation $\min_q \frac{1}{T}\sum_{k=1}^{T} \delta(\mathbf{x}^\mathsf{T}\mathbf{r}_k - q)$.

Variance

Choosing $\delta_{\text{var}}(y) = \frac{1}{2}y^2$ we get $D_{\delta_{\text{var}}}$, a different expression for the portfolio variance. The optimal q in this case is the portfolio mean return $\mu_\mathbf{x}$. This gives us a way to perform standard mean–variance optimization directly using scenario data, without the need for a sample covariance matrix:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2T}\sum_{k=1}^{T}(\mathbf{x}^\mathsf{T}\mathbf{r}_k - q)^2 \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. \end{array} \tag{8.1}$$

By defining new variables $t_k^+ = \max\left(\mathbf{x}^\mathsf{T}\mathbf{r}_k - q, 0\right)$ and $t_k^- = \max\left(-\mathbf{x}^\mathsf{T}\mathbf{r}_k + q, 0\right)$, then using Sec. 13.1.1 we arrive at a QO model:

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{2T}\|\mathbf{t}^+\|^2 + \frac{1}{2T}\|\mathbf{t}^-\|^2 & & \\ & \mathbf{x}^\mathsf{T}\mathbf{R} - q & = & \mathbf{t}^+ - \mathbf{t}^-, \\ & \mathbf{t}^+, \mathbf{t}^- & \geq & 0, \\ \text{subject to} & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{8.2}$$

where $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_T]$ is the scenario matrix, where each column is a scenario. The norms can be further modeled by the rotated quadratic cone; see in Sec. 13.1.1. While this formulation does not need covariance estimation, it can still suffer from computational inefficiency because the number of variables and the number of constraints depend on the sample size T.

α-shortfall

Choosing $\delta_{\text{sf},\alpha}(y) = \alpha y - \min(y, 0)$ or equivalently $\delta_{\text{sf},\alpha}(y) = \alpha\max(y, 0) + (1 - \alpha)\max(-y, 0)$ for $\alpha \in (0, 1)$ will give the *α-shortfall* $D_{\delta_{\text{sf},\alpha}}$ studied in [Lau01]. The optimal q will be the α-quantile of the portfolio return. α-shortfall is a deviation measure with favorable robustness properties when the portfolio return has an asymmetric distribution, fat tailed symmetric distribution, or exhibits multivariate tail-dependence. The α parameter adjusts the level of asymmetry in the δ function, allowing us to penalize upside and downside returns differently.

Portfolio optimization using sample α-shortfall can be formulated as

$$\begin{array}{lll} \text{minimize} & \frac{\alpha}{T}\sum_{k=1}^{T}(\mathbf{x}^\mathsf{T}\mathbf{r}_k - q) - \frac{1}{T}\sum_{k=1}^{T}\min(\mathbf{x}^\mathsf{T}\mathbf{r}_k - q, 0) & \\ \text{subject to} & \mathbf{x} \in \mathcal{F} & \end{array} \tag{8.3}$$

By defining new variables $t_k^- = \max(-\mathbf{x}^\mathsf{T}\mathbf{r}_k + q, 0)$, and model accoring to Sec. 13.1.1 we arrive at an LO model:

$$\begin{array}{lll} \text{minimize} & \alpha(\mathbf{x}^\mathsf{T}\boldsymbol{\mu} - q) + \frac{1}{T}\sum_{k=1}^{T} t_k^- & \\ & \mathbf{t}^- \geq -\mathbf{x}^\mathsf{T}\mathbf{r}_k + q, & \\ & \mathbf{t}^- \geq 0, & \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. & \end{array} \tag{8.4}$$

A drawback of this model is that the number of constraints depends on the sample size T, resulting in computational inefficiency for large number of samples.

For elliptically symmetric return distributions the α-shortfall is equivalent to the variance in the sense that the corresponding sample portfolio estimators will be estimating the same portfolio. In fact for normal distribution, the *α-shortfall* is proportional to the portfolio variance: $D_{\delta_{\text{sf},\alpha}} = \frac{\phi(q_\alpha)}{\alpha}\sqrt{D_{\delta_{\text{var}}}}$, where $\phi(q_\alpha)$ is the value of the standard normal density function at its α-quantile.

However when return is symmetric but not normally distributed, the sample portfolio estimators for α-shortfall can have less estimation error than sample portfolio estimators for variance.

MAD

A special case of α-shortfall is for $\alpha = 0.5$. The function $\delta_{\text{MAD}}(y) = \frac{1}{2}|y|$ gives us $D_{\delta_{\text{MAD}}}$, which is called the *mean absolute deviation* (MAD) measure or the *L_1-risk*. For this case the optimal q will be the median of the portfolio return, and the sample MAD portfolio optimization problem can be formulated as

$$\begin{array}{lll} \text{minimize} & \frac{1}{2T}\sum_{k=1}^{T}|\mathbf{x}^\mathsf{T}\mathbf{r}_k - q| & \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. & \end{array} \tag{8.5}$$

After modeling the absolute value based on Sec. 13.1.1 we arrive at the following LO:

$$\begin{array}{lll} \text{minimize} & \frac{1}{2T}\sum_{k=1}^{T} t_k & \\ \text{subject to} & \mathbf{x}^\mathsf{T}\mathbf{R} - q \geq -\mathbf{t}, & \\ & \mathbf{x}^\mathsf{T}\mathbf{R} - q \leq \mathbf{t}, & \\ & \mathbf{x} \in \mathcal{F}, & \end{array} \tag{8.6}$$

where $\mathbf{R}$ is the return data matrix with one observation $\mathbf{r}_k,\ k = 1, \ldots, T$ in each column. Note that the number of constraints in this LO problem again depends on the sample size T.

For normally distributed returns, the MAD is proportional to the variance: $D_{\delta_{MAD}} = \sqrt{\frac{2}{\pi}}\sqrt{D_{\delta_{\text{var}}}}$.

The L_1-risk can also be applied without minimizing over q. We can just let q to be the sample portfolio mean instead, i. e., $q = \mathbf{x}^\mathsf{T}\boldsymbol{\mu}$ [KY91].

Risk measure from the Huber function

Another robust portfolio estimator can be obtained for the case of symmetric return distributions using the Huber function

$$\delta_{\text{H}}(y) = \begin{cases} y^2, & |y| \leq c \\ 2c|y| - c^2, & |y| > c \end{cases},$$

yielding the risk measure $D_{\delta_{\text{H}}}$. A different form of the Huber function is $\delta_{\text{H}}(y) = \min_u u^2 + 2c|y-u|$, which leads to a QO formulation:

$$\begin{array}{lll} \text{minimize} & \frac{1}{T}\sum_{k=1}^{T} u_k^2 + \frac{1}{T}\sum_{k=1}^{T} 2c|\mathbf{x}^\mathsf{T}\mathbf{r}_k - q - u_k| & \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. & \end{array} \tag{8.7}$$

Modeling the absolute value based on Sec. 13.1.1 we get

$$\begin{array}{lll} \text{minimize} & \frac{1}{T}\sum_{k=1}^{T} u_k^2 + \frac{1}{T}\sum_{k=1}^{T} 2ct_k & \\ \text{subject to} & \mathbf{x}^\mathsf{T}\mathbf{R} - q - \mathbf{u} \geq -\mathbf{t}, & \\ & \mathbf{x}^\mathsf{T}\mathbf{R} - q - \mathbf{u} \leq \mathbf{t}, & \\ & \mathbf{x} \in \mathcal{F}. & \end{array} \tag{8.8}$$

Note that the size of this problem depends on the number of samples T.

8.1.2 Downside deviation measures

In financial context many distributions are skewed. Investors might only be concerned with negative deviations (losses) relative to the expected portfolio return $\mu_{\mathbf{x}}$, or in general with falling short of some benchmark return r_{bm}. In these cases downside deviation measures are more appropriate.

A class of downside deviation measures are *lower partial moments* of order n:

$$\text{LPM}_n(r_{\text{bm}}) = \mathbb{E}(\max(r_{\text{bm}} - R_{\mathbf{x}}, 0)^n),$$

where r_{bm} is some given target return. The discrete version of this measure is $\sum_{k=1}^{T} p_k \max(r_{\text{bm}} - \mathbf{x}^\mathsf{T}\mathbf{r}_k, 0)^n$, where $\mathbf{r}_k$ is a scenario of the portfolio return $R_{\mathbf{x}}$ occurring with probability p_k.

We have the following special cases:

- LPM_0 is the probability of loss relative to the target return r_{bm}. LPM_0 is an incomplete measure of risk, because it does not provide any indication of how severe the shortfall can be, should it occur. Therefore it is best used as a constraint while optimizing for a different risk measure. This way LPM_0 can still provide information about the risk tolerance of the investor.

- LPM_1 is the *expected loss*, also called *target shortfall*.

- LPM_2 is called *target semi-variance*.

While lower partial moments only consider outcomes below the target r_{bm}, the optimization still uses the entire distribution of $R_{\mathbf{x}}$. The right tail of the distribution (representing the outcomes above the target) is captured in the mean $\mu_{\mathbf{x}}$ of the distribution.

The LPM_n optimization problem can be formulated as [Lau01]

$$\begin{array}{lll} \text{minimize} & \sum_{k=1}^{T} p_k \max(r_{\text{bm}} - \mathbf{x}^{\mathsf{T}}\mathbf{r}_k, 0)^n & \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. & \end{array} \tag{8.9}$$

If we define the new variables $t_k^- = \max(r_{\text{bm}} - \mathbf{x}^{\mathsf{T}}\mathbf{r}_k, 0)$, then for $n = 1$ we arrive at an LO and for $n = 2$ we arrive at a QO problem:

$$\begin{array}{lllll} \text{minimize} & \sum_{k=1}^{T} p_k (t_k^-)^n & & & \\ \text{subject to} & \mathbf{t}^- & \geq & r_{\text{bm}} - \mathbf{x}^{\mathsf{T}}\mathbf{R}, & \\ & \mathbf{t}^- & \geq & 0, & \\ & \mathbf{x} & \in & \mathcal{F}, & \end{array} \tag{8.10}$$

where $\mathbf{t}^-$ is T dimensional vector. Thus there will be $N + T$ variables, and the number of constraints will also depend on the sample size T.

Contrary to the variance, LPM_n measures are consistent with more general classes of investor utility functions, and assume less about the return distribution. Thus they better reflect investor preferences and are valid under a broader range of conditions [Har91].

However, LPMs are only useful for skewed distributions. If the return distribution is symmetric, LPM_1 and LPM_2 based portfolio estimates will be equivalent to MAD and variance based ones. In particular the MAD for a random variable with density $f(x)$ will be the same as LPM_1 of a random variable with density $f(x) + f(-x)$. Also, the mean-LPM optimization model is very sensitive to the changes in the target value.

8.2 Tail risk measures

Tail risk measures try to capture the risk of extreme events. The measures described below are commonly defined for random variables treating loss as positive number, so to apply them on security returns R we have to flip the sign and define loss as $L = -R$.

Let X, Y be random variables of investment losses, and let τ be a mapping. The formal properties that a reasonable tail risk measure should satisfy are the following [FS02]:

1. Monotonicity: If $X \leq Y$, then $\tau(X) \leq \tau(Y)$,
2. Translation invariance: $\tau(X + m) = \tau(X) + m,\ m \in \mathbb{R}$,
3. Subadditivity: $\tau(X + Y) \leq \tau(X) + \tau(Y)$,
4. Positive homogeneity: $\tau(cX) = c\tau(X),\ c \geq 0$

If τ satisfies these properties then it is called *coherent risk measure*. If we relax condition 3 and 4 to the property of *convexity*, then we get the more general class of *convex risk measures*. Not all risk measures used in practice satisfy these properties, violation of condition 1 and/or 2 is typical. However, in the context of portfolio selection these are less critical, the property of convexity is most important to achieve risk diversification [Lau01].

8.2.1 Value-at-Risk

Denote the α-quantile of a random variable L by $q_\alpha(L) = \inf\{x|\mathbb{P}(L \leq x) \geq \alpha\}$. If L is the loss over a given time horizon, then the *value-at-risk* (VaR) of L with confidence level α (or risk level $1-\alpha$) is defined to be

$$\mathrm{VaR}_\alpha(L) = q_\alpha(L).$$

This is the amount of loss (a positive number) over the given time horizon that will not be exceeded with probability α. However, VaR does not give information about the magnitude of loss in case of the $1-\alpha$ probability event when losses are greater than $q_\alpha(L)$. Also it is not a coherent risk measure (it does not respect convexity property), meaning it can discourage diversification.[2] In other words, portfolio VaR may not be lower than the sum of the VaRs of the individual securities.

8.2.2 Conditional Value-at-Risk

A modification of VaR that is a coherent risk measure is *conditional value-at-risk* (CVaR). CVaR is the average of the $1-\alpha$ fraction of worst case losses, i. e., the losses equal to or exceeding $\mathrm{VaR}_\alpha(L)$. Its most general definition (applicable also for loss distributions with possible discontinuities) can be found in [RU02]. Let $\lambda_\alpha = \frac{1}{1-\alpha}(\mathbb{P}(L \leq q_\alpha(L)) - \alpha)$. Then the CVaR of L with confidence level α will be

$$\mathrm{CVaR}_\alpha(L) = \lambda_\alpha q_\alpha(L) + (1-\lambda_\alpha)\mathbb{E}(L|L > q_\alpha(L)),$$

which is a linear combination of VaR and the quantity called *mean shortfall*. The latter is also not coherent on its own.

If the distribution function $\mathbb{P}(L \leq q)$ is continuous at $q_\alpha(L)$ then $\lambda_\alpha = 0$. If there is discontinuity and we have to account for a probability mass concentrated at $q_\alpha(L)$, then λ_α is nonzero in general. This is often the case in practice, when the loss distribution is discrete, for example for scenario based approximations.

CVaR for discrete distribution

Suppose that the loss distribution is described by points $q_1 < \cdots < q_T$ with nonzero probabilities $p_1, \ldots, p_T$, and $i_\alpha \in \{1, \ldots, T\}$ is the index such that $\sum_{i=1}^{i_\alpha - 1} p_i < \alpha \leq \sum_{i=1}^{i_\alpha} p_i$. Then $\mathrm{VaR}_\alpha(L) = q_{i_\alpha}$ and

$$\begin{aligned} \mathrm{CVaR}_\alpha(L) &= \lambda_\alpha q_{i_\alpha} + (1-\lambda_\alpha)\tfrac{1}{\sum_{i>i_\alpha} p_i}\textstyle\sum_{i>i_\alpha} p_i q_i \quad = \\ &= \tfrac{1}{1-\alpha}\left((\textstyle\sum_{i=1}^{i_\alpha} p_i - \alpha)q_{i_\alpha} + \sum_{i>i_\alpha} p_i q_i\right), \end{aligned} \tag{8.11}$$

where $\lambda_\alpha = \frac{1}{1-\alpha}(\sum_{i=1}^{i_\alpha} p_i - \alpha)$. As a special case, if we assume that $p_i = \frac{1}{T}$, i. e., we have a sample average approximation, then the above formula simplifies with $i_\alpha = \lceil \alpha T \rceil$.

It can be seen that $\mathrm{VaR}_\alpha(L) \leq \mathrm{CVaR}_\alpha(L)$ always holds. CVaR is also consistent with second-order stochastic dominance (SSD), i. e., the CVaR efficient portfolios are the ones actually preferred by some wealth-seeking risk-averse investors.

[2] VaR only becomes coherent when return is normally distributed, but in this case it is proportional to the standard deviation.

Portfolio optimization with CVaR

If we substitute portfolio loss scenarios into formula (8.11), we can see that the quantile $(-\mathbf{R}^\mathsf{T}\mathbf{x})_{i_\alpha}$ will depend on $\mathbf{x}$. It follows that the ordering of the scenarios and the index i_α will also depend on $\mathbf{x}$, making it difficult to optimize. However, note that formula (8.11) is actually the linear combination of largest elements of the vector $\mathbf{q}$. We can thus apply Sec. 13.1.1 to get the dual form of $\mathrm{CVaR}_\alpha(L)$, which is an LO problem:

$$\begin{array}{lrcl} \text{minimize} & t + \frac{1}{1-\alpha}\mathbf{p}^\mathsf{T}\mathbf{u} & & \\ \text{subject to} & \mathbf{u} + t & \geq & \mathbf{q}, \\ & \mathbf{u} & \geq & \mathbf{0}. \end{array} \tag{8.12}$$

Note that problem (8.12) is equivalent to

$$\min_t \; t + \tfrac{1}{1-\alpha}\textstyle\sum_i p_i \max(0, q_i - t). \tag{8.13}$$

This convex function (8.13) is exactly the one found in [RU00], where it is also proven to be valid for continuous probability distributions as well.

Now we can substitute the portfolio return into $\mathbf{q}$, and optimize over $\mathbf{x}$ to find the portfolio that minimizes CVaR:

$$\begin{array}{lrcl} \text{minimize} & t + \frac{1}{1-\alpha}\mathbf{p}^\mathsf{T}\mathbf{u} & & \\ \text{subject to} & \mathbf{u} + t & \geq & -\mathbf{R}^\mathsf{T}\mathbf{x}, \\ & \mathbf{u} & \geq & \mathbf{0}, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{8.14}$$

Because CVaR is represented as a convex function in formula (8.13), we can also formulate an LO to maximize expected return, while limiting risk in terms of CVaR:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & t + \frac{1}{1-\alpha}\mathbf{p}^\mathsf{T}\mathbf{u} & \leq & \gamma, \\ & \mathbf{u} + t & \geq & -\mathbf{R}^\mathsf{T}\mathbf{x}, \\ & \mathbf{u} & \geq & \mathbf{0}, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{8.15}$$

The drawback of optimizing CVaR using problems (8.14) or (8.15) is that both the number of variables and the number of constraints depend on the number of scenarios T. This can make the LO model computationally expensive for very large number of samples. For example if the distribution of return is not known, we might need to obtain or simulate a substantial amount of samples, depending on the confidence level α.

8.2.3 Entropic Value-at-Risk

A more general risk measure in this class is the *entropic value-at-risk* (EVaR). EVaR is also a coherent risk measure, with additional favorable monotonicity properties; see in [AJ12]. It is defined as the tightest upper bound on VaR obtained from the Chernoff inequality:

$$\mathrm{EVaR}_\alpha(L) = \inf_{s>0} \frac{1}{s} \log\left(\frac{M_L(s)}{1-\alpha}\right), \tag{8.16}$$

where $M_L(s) = \mathbb{E}(\mathrm{e}^{sL})$ is the moment generating function of L. The EVaR incorporates losses both less and greater than the VaR simultaneously, thus it always holds that $\mathrm{CVaR}_\alpha(L) \leq \mathrm{EVaR}_\alpha(L)$.

EVaR for discrete distribution

Based on the definition (8.16), the discrete version of the EVaR will be

$$\mathrm{EVaR}_\alpha(L) = \inf_{s>0} \frac{1}{s} \log\left(\frac{\sum_i p_i \mathrm{e}^{sq_i}}{1-\alpha}\right), \tag{8.17}$$

We can make formula (8.17) convex by substituting a new variable $t = \frac{1}{s}$:

$$\mathrm{EVaR}_\alpha(L) = \inf_{t>0} t\log\left(\sum_i p_i \mathrm{e}^{q_i/t}\right) - t\log(1-\alpha). \tag{8.18}$$

We can transform formula (8.18) into a conic optimization problem by substituting the first term of the objective with a new variable z and adding the new constraint $z \geq \mathrm{EVaR}_\alpha(L)$. Then we apply the rule Sec. 13.1.1:

$$\begin{array}{lrcl} \text{minimize} & z - t\log(1-\alpha), & & \\ \text{subject to} & \sum_{i=1}^T p_i u_i & \leq & t, \\ & (u_i, t, q_i - z) & \in & K_{\exp},\ i = 1, \ldots, T, \\ & t & \geq & 0. \end{array} \tag{8.19}$$

Portfolio optimization with EVaR

Now we can substitute the portfolio return into $\mathbf{q}$, and optimize over $\mathbf{x}$ to find the portfolio that minimizes EVaR:

$$\begin{array}{lrcl} \text{minimize} & z - t\log(1-\alpha) & & \\ \text{subject to} & \sum_{i=1}^T p_i u_i & \leq & t, \\ & (u_i, t, -\mathbf{r}_i^\mathsf{T}\mathbf{x} - z) & \in & K_{\exp},\ i = 1, \ldots, T, \\ & t & \geq & 0, \\ & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & \geq & r_{\min}, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{8.20}$$

Because EVaR is represented as a convex function (8.18), we can also formulate a conic problem to maximize expected return, while limiting risk in terms of EVaR:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & & \\ \text{subject to} & z - t\log(1-\alpha), & \leq & \gamma, \\ & \sum_{i=1}^T p_i u_i & \leq & t, \\ & (u_i, t, -\mathbf{r}_i^\mathsf{T}\mathbf{x} - z) & \in & K_{\exp},\ i = 1, \ldots, T, \\ & t & \geq & 0, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{8.21}$$

A disadvantage of the EVaR conic model is that it still depends on T in the number of exponential cone contraints.

Note that if we assume the return distribution to be a Gaussian mixture, we can find a different approach to computing EVaR. See in Sec. 8.3.2.

8.2.4 Relationship between risk measures

Suppose that $\tau(X)$ is a coherent tail risk measure that additionally satisfies $\tau(X) > -\mathbb{E}(X)$. Suppose also that $D(X)$ is a deviation measure that additionally satisfies $D(X) \leq \mathbb{E}(X)$ for all $X \geq 0$. Then these subclasses of risk measures can be related through the following identities:

$$\begin{aligned} D(X) &= \tau(X - \mathbb{E}(X)) \\ \tau(X) &= D(X) - \mathbb{E}(X) \end{aligned} \tag{8.22}$$

For example, α-shortfall and CVaR is related this way. Details can be found in [RUZ06].

8.2.5 Practical considerations

Many risk measures in this chapter can be computed in practice through scenario based approximations. The relevant optimization problems can then be conveniently formulated as LO or QO problems. The advantage of these models is that they do not need a covariance matrix estimate. The simplification of the problem structure, however, comes at the cost of increasing the problem dimension by introducing a number of new variables and constraints proportional to the number of scenarios T.

If T is large, this can lead to computational burden not only because of the increased number of variables and constraints, but also because the scenario matrix $\mathbf{R}$ also depending on T becomes very non-sparse, making the solution process less efficient.

If T is too small, specifically $T < N$, then there will be fewer scenarios then parameters to estimate. This makes the portfolio optimization problem very imprecise, and prone to overfit the data. Also depending on other constraints, solution of LO problems can have computational difficulties. To mitigate this, we can use regularization methods, such as L_1 (lasso) or L_2 (ridge) penalty terms in the objective:

$$\begin{aligned} \text{minimize} \quad & \mathcal{R}_{\mathbf{x}} + \lambda \|\mathbf{x} - \mathbf{x}_{\text{pri}}\|_p^p \\ \text{subject to} \quad & \mu^{\mathsf{T}}\mathbf{x} \geq R, \\ & \mathbf{1}^{\mathsf{T}}\mathbf{x} = 1, \end{aligned} \tag{8.23}$$

where $\mathcal{R}_{\mathbf{x}}$ is the portfolio risk, $\mathbf{x}_{\text{pri}}$ is a prior portfolio, and λ is the regularization strength parameter. The penalty term will ensure that the extreme deviations from the prior are unlikely. Thus λ basically sets the investor confidence in the portfolio $\mathbf{x}_{\text{pri}}$. Using $p = 1, 2$ we arrive at LO and QO problems respectively. See details in [Lau01].

8.3 Expected utility maximization

Apart from selecting different risk measures, we can also approach portfolio optimization through the use of utility functions. We specify a concave and increasing utility function that measures the investors preference for each specific outcome. Then the objective is to maximize the expected utility under the return distribution.

Expected utility maximization can take into account any type of return distribution, while MVO works only with the first two moments, therefore it is accurate only for normally distributed return. MVO is also equivalent with expected utility maximization if the utility function is quadratic, because it models the investors indifference about higher moments of return. The only advantage of MVO in this comparison is that it works without having to discretize the return distribution and work with scenarios.

8.3.1 Optimal portfolio using gaussian mixture return

In [LB22] an expected utility maximization approach is taken, assuming that the return distribution is a Gaussian mixture (GM). The benefits of a GM distribution are that it can approximate any continuous distribution, including skewed and fat-tailed ones. Also its components can be interpreted as return distributions given a specific market regime. Moreover, the expected utility maximization using this return model can be formulated as a convex optimization problem without needing return scenarios, making this approach as efficient and scalable as MVO.

We denote security returns having Gaussian mixture (GM) distribution with K components as

$$R \sim \mathcal{GM}(\{\mu_i, \Sigma_i, \pi_i\}_{i=1}^K), \tag{8.24}$$

where μ_i is the mean, Σ_i is the covariance matrix, and π_i is the probability of component i. As special cases, for $K = 1$ we get the normal distribution $\mathcal{N}(\mu_1, \Sigma_1)$, and for $\Sigma_i = 0,\ i = 1, \ldots, K$ we get a scenario distribution with K return scenarios $\{\mu_1, \ldots, \mu_K\}$.

Using definition (8.24), the distribution of portfolio return $R_{\mathbf{x}}$ will also be GM:

$$R_{\mathbf{x}} \sim \mathcal{GM}(\{\mu_{\mathbf{x},i}, \sigma_{\mathbf{x},i}^2, \pi_i\}_{i=1}^K), \tag{8.25}$$

where $\mu_{\mathbf{x},i} = \mu_i^\mathsf{T}\mathbf{x}$, and $\sigma_{\mathbf{x},i}^2 = \mathbf{x}^\mathsf{T}\Sigma_i\mathbf{x}$.

To select the optimal portfolio we use the exponential utility function $U_\delta(x) = 1 - e^{-\delta x}$, where $\delta > 0$ is the risk aversion parameter. This choice allows us to write the expected utility of the portfolio return $R_{\mathbf{x}}$ using the moment generating function:

$$\mathbb{E}(1 - e^{-\delta R_{\mathbf{x}}}) = 1 - M_{R_{\mathbf{x}}}(-\delta). \tag{8.26}$$

Thus maximizing the function (8.26) is the same as minimizing the moment generating function of the portfolio return, or equivalently, we can minimize its logarithm, the cumulant generating function:

$$\begin{array}{lrcl} \text{minimize} & K_{R_{\mathbf{x}}}(-\delta) & & \\ \text{subject to} & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{8.27}$$

If $R_{\mathbf{x}}$ is assumed to be a GM random variable, then its cumulant generating function will be the following:

$$K_{R_{\mathbf{x}}}(-\delta) = \log\left(\sum_{i=1}^K \pi_i \exp\left(-\delta\mu_i^\mathsf{T}\mathbf{x} + \frac{\delta^2}{2}\mathbf{x}^\mathsf{T}\Sigma_i\mathbf{x}\right)\right) \tag{8.28}$$

The function (8.28) is convex in $\mathbf{x}$ because it is a composition of the convex and increasing log-sum-exp function and a convex quadratic function. Note that for $K = 1$, we get back the same quadratic utility objective as in version (2.3) of the MVO problem.

Assuming we have the GM distribution parameter estimates $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i,\ i = 1, \ldots, K$, we can apply Sec. 13.1.1 and Sec. 13.1.1 to arrive at the conic model of the utility maximization problem (8.27):

$$\begin{array}{lrcl} \text{minimize} & z & & \\ \text{subject to} & \sum_{i=1}^K \pi_i u_i & \leq & 1, \\ & (u_i, 1, -\delta\boldsymbol{\mu}_i^\mathsf{T}\mathbf{x} + \delta^2 q_i - z) & \in & K_{\exp},\ i = 1, \ldots, K, \\ & (q_i, 1, \mathbf{G}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_{\mathrm{r}}^{N+2},\ i = 1, \ldots, K, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{8.29}$$

where q_i is an upper bound for the quadratic term $\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{\Sigma}_i\mathbf{x}$, and $\mathbf{G}_i$ is such that $\mathbf{\Sigma}_i = \mathbf{G}_i\mathbf{G}_i^\mathsf{T}$.

8.3.2 EVaR using Gaussian mixture return

We introduced Entropic Value-at-Risk (EVaR) in Sec. 8.2.3. EVaR can also be expressed using the cumulant generating function $K_L(s) = \log(M_L(s))$ of the loss L:

$$\mathrm{EVaR}_\alpha(L) = \inf_{s>0} \frac{1}{s} K_L(s) - \frac{1}{s}\log(1-\alpha). \tag{8.30}$$

After substituting the return $R = -L$ instead of the loss, we see that $K_{-R}(s) = K_R(-s)$. Assuming now that $R = R_\mathbf{x}$ is the portfolio return, we get the following optimization problem:

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{s}K_{R_\mathbf{x}}(-s) - \frac{1}{s}\log(1-\alpha) & & \\ \text{subject to} & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{8.31}$$

We can find a connection between EVaR computation (8.31) and maximization of expected exponential utility (8.27). Suppose that the pair $(\mathbf{x}^*, s^*)$ is optimal for problem (8.31). Then $\mathbf{x}^*$ also optimal for problem (8.27), with risk aversion parameter $\delta = s^*$.

By assuming a GM distribution for security return, we can optimize problem (8.31) without needing a scenario distribution. First, to formulate it as a convex optimization problem, define the new variable $t = \frac{1}{s}$:

$$\mathrm{EVaR}_\alpha(L) = \inf_{t>0} tK_{R_\mathbf{x}}\left(-\frac{1}{t}\right) - t\log(1-\alpha). \tag{8.32}$$

We can observe that the first term in formula (8.32) is the perspective of $K_{R_\mathbf{x}}(-1)$. Therefore by substituting the cumulant generating function (8.28) of the GM portfolio return, we get a convex function in the portfolio $\mathbf{x}$:

$$\mathrm{EVaR}_\alpha(L) = \inf_{t>0} t\log\left(\sum_{i=1}^K \pi_i \exp\left(-\frac{1}{t}\mu_i^\mathsf{T}\mathbf{x} + \frac{1}{2t^2}\mathbf{x}^\mathsf{T}\Sigma_i\mathbf{x}\right)\right) - t\log(1-\alpha). \tag{8.33}$$

Assuming we have the GM distribution parameter estimates $\boldsymbol{\mu}_i, \mathbf{\Sigma}_i,\ i = 1,\ldots,K$, we can apply Sec. 13.1.1 and Sec. 13.1.1 to arrive at the conic model of the EVaR minimization problem (8.31):

$$\begin{array}{lrcl} \text{minimize} & z - t\log(1-\alpha) & & \\ \text{subject to} & \sum_{i=1}^K \pi_i u_i & \leq & t, \\ & (u_i, t, -\boldsymbol{\mu}_i^\mathsf{T}\mathbf{x} + q_i - z) & \in & K_{\exp},\ i = 1,\ldots,K, \\ & (q_i, 1, \mathbf{G}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{N+2},\ i = 1,\ldots,K, \\ & t & \geq & 0, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{8.34}$$

where q_i is an upper bound for the quadratic term $\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{\Sigma}_i\mathbf{x}$, and $\mathbf{G}_i$ is such that $\mathbf{\Sigma}_i = \mathbf{G}_i\mathbf{G}_i^\mathsf{T}$.

The huge benefit of this EVaR formulation is that its size does not depend on the number of scenarios, because it is derived without using a scenario distribution. It depends only on the number of GM components K.

8.4 Example

This example shows how can we compute the CVaR efficient frontier using the dual form of CVaR in **MOSEK** Fusion.

8.4.1 Scenario generation

The input data is again obtained the same way as detailed in Sec. 3.4.2, but we do only the steps until Sec. 3.4.3. This way we get the expected return estimate $\boldsymbol{\mu}^{\log}$ and the covariance matrix estimate $\boldsymbol{\Sigma}^{\log}$ of yearly logarithmic returns. These returns have approximately normal distribution, so we can easily generate T number of return scenarios from them, using the numpy default random number generator. Here we choose $T = 99999$. This number ensures to have a nonzero λ_α in formula (8.11), which is the most general case.

```
# Number of scenarios
T = 99999
# Generate logarithmic return scenarios assuming normal distribution
R_log = np.random.default_rng().multivariate_normal(m_log, S_log, T)
```

Next, we convert the received logarithmic return scenarios to linear return scenarios using the inverse of formula (3.2).

```
# Convert logarithmic return scenarios to linear return scenarios
R = np.exp(scenarios_log) - 1
R = R.T
```

We transpose the resulting matrix just to remain consistent with the notation in this chapter, namely that each column of $\mathbf{R}$ is a scenario. We also set the scenario probabilities to be $\frac{1}{T}$:

```
# Scenario probabilities
p = np.ones(T) / T
```

8.4.2 The optimization problem

The optimization problem we solve here resembles problem (2.12), but we will change the risk measure from portfolio standard deviation to portfolio CVaR:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \delta \cdot \mathrm{CVaR}_\alpha(-\mathbf{R}^\mathsf{T}\mathbf{x}) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{8.35}$$

Applying the dual CVaR formula (8.13), we get:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} - \delta \left(t + \frac{1}{1-\alpha}\sum_i p_i \max(0, -\mathbf{r}_i^\mathsf{T}\mathbf{x} - t)\right) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{8.36}$$

We know that formula (8.13) equals CVaR only if it is minimal in t. This will be satisfied in the optimal solution of problem (8.36), because the CVaR term is implicitly forced to become smaller, and t is independent from $\mathbf{x}$.

Now we model the maximum function, and arrive at the following LO model of the mean-CVaR efficient frontier:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^{\mathsf{T}}\mathbf{x} - \delta t - \frac{\delta}{1-\alpha}\sum_i p_i u_i & & \\ \text{subject to} & \mathbf{u} & \geq & -\mathbf{R}^{\mathsf{T}}\mathbf{x} - t, \\ & \mathbf{u} & \geq & \mathbf{0}, \\ & \mathbf{1}^{\mathsf{T}}\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{8.37}$$

8.4.3 The Fusion model

Here we show the Fusion model of problem (8.37).

```
def EfficientFrontier(N, T, m, R, p, alpha, deltas):
    with Model("CVaRFrontier") as M:
        # Variables
        # x - fraction of holdings relative to the initial capital.
        # It is constrained to take only positive values.
        x = M.variable("x", N, Domain.greaterThan(0.0))

        # Budget constraint
        M.constraint('budget', Expr.sum(x) == 1)

        # Auxiliary variables.
        t = M.variable("t", 1, Domain.unbounded())
        u = M.variable("u", T, Domain.unbounded())

        # Constraint modeling maximum
        M.constraint(u >= - R * x - Var.repeat(t, T))
        M.constraint(u >= 0)

        # Objective
        delta = M.parameter()
        cvar_term = t + u.T @ p / (1-alpha)
        M.objective('obj', ObjectiveSense.Maximize,
                           x.T @ m - delta * cvar_term)

        # Create DataFrame to store the results.
        columns = ["delta", "obj", "return", "risk"] + \
                  df_prices.columns.tolist()
        df_result = pd.DataFrame(columns=columns)
        for d in deltas:
            # Update parameter
            delta.setValue(d)

            # Solve optimization
            M.solve()

            # Save results
            portfolio_return = m @ x.level()
```

(continues on next page)

(continued from previous page)

```
            portfolio_risk = t.level()[0] + \
                1/(1-alpha) * np.dot(p, u.level())
            row = pd.Series([d, M.primalObjValue(),
                             portfolio_return, portfolio_risk] + \
                            list(x.level()), index=columns)
            df_result = df_result.append(row, ignore_index=True)

        return df_result
```

Next, we compute the efficient frontier. We select the confidence level $\alpha = 0.95$. The following code produces the optimization results:

```
alpha = 0.95

# Compute efficient frontier with and without shrinkage
deltas = np.logspace(start=-1, stop=2, num=20)[::-1]
df_result = EfficientFrontier(N, T, m, R, p, alpha, deltas)
```

On Fig. 8.1 we can see the risk-return plane, and on Fig. 8.2 the portfolio composition for different levels of risk.

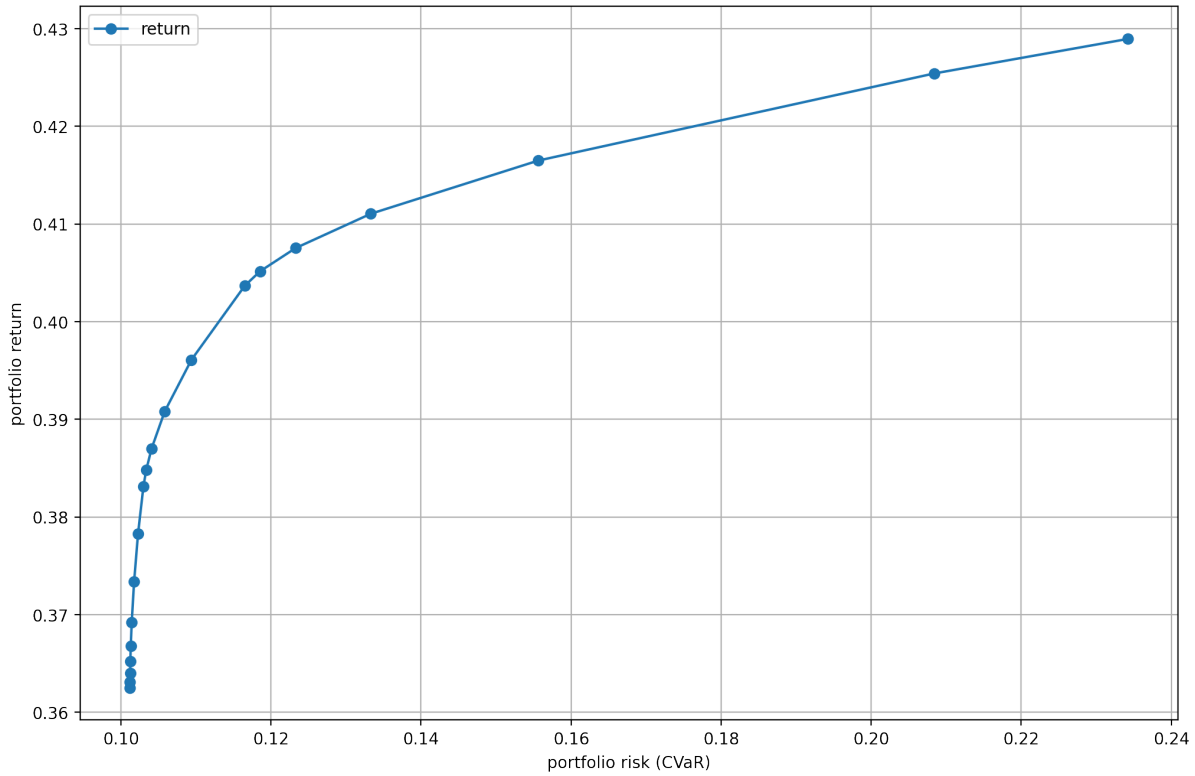

Fig. 8.1: The CVaR efficient frontier.

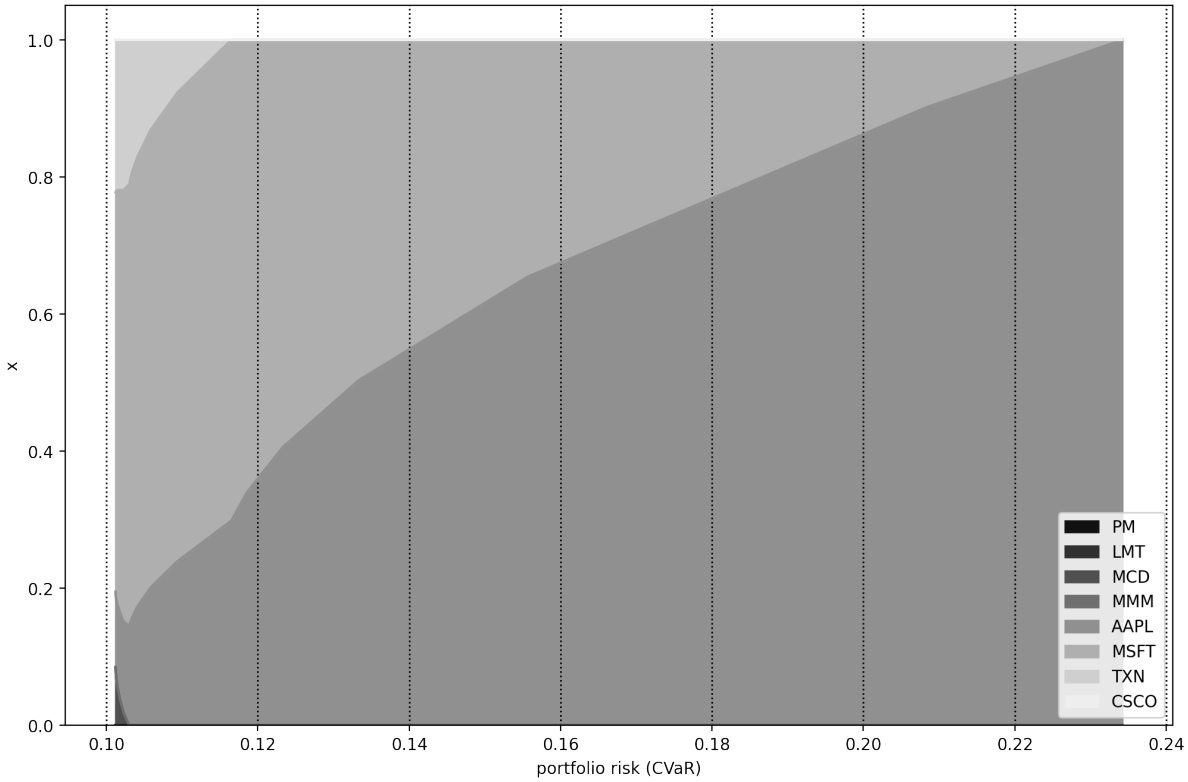

Fig. 8.2: Portfolio composition $\mathbf{x}$ with varying level of CVaR risk.

Chapter 9

Risk budgeting

Traditional MVO only considers the risk of the portfolio and ignores how the risk is distributed among individual securities. This might result in risk being concentrated into only a few securities, when these gain too much weight in the portfolio.

Risk budgeting techniques were developed to give portfolio managers more control over the amount of risk each security contributes to the total portfolio risk. They allocate the risk instead of the capital, thus diversify risk more directly. Risk budgeting also makes the resulting portfolio less sensitive to estimation errors, because it excludes the use of expected return estimates [CK20, Pal20].

9.1 Risk contribution

If the risk measure is a homogeneous function of degree one, then we can use Euler decomposition write the total portfolio risk $\mathcal{R}_{\mathbf{x}}$ as the sum of risk contributions $\mathrm{rc}_{\mathbf{x},i}$ of each security:

$$\mathcal{R}_{\mathbf{x}} = \sum_{i=1}^{N} \mathrm{rc}_{\mathbf{x},i} = \sum_{i=1}^{N} x_i \mathrm{mrc}_{\mathbf{x},i} = \sum_{i=1}^{N} x_i \frac{\partial \mathcal{R}_{\mathbf{x}}}{\partial x_i} = \mathbf{x}^{\mathsf{T}} \nabla \mathcal{R}_{\mathbf{x}} \tag{9.1}$$

where $\mathrm{mrc}_{\mathbf{x},i} = \frac{\partial \mathcal{R}_{\mathbf{x}}}{\partial x_i}$ is called *marginal risk contribution*, because it measures the sensitivity of the portfolio standard deviation to changes in x_i.

9.2 Risk budgeting portfolio

With the above definitions, the *risk budgeting portfolio* aims to allocate the risk contributions according to a predetermined *risk profile* $\mathbf{b}$. In other words, it aims to find the portfolio $\mathbf{x}$ that solves the system

$$\mathbf{x} \circ \nabla \mathcal{R}_{\mathbf{x}} = \mathbf{b} \circ \mathcal{R}_{\mathbf{x}} = \mathbf{b} \circ \mathbf{x}^{\mathsf{T}} \nabla \mathcal{R}_{\mathbf{x}} \tag{9.2}$$

where $\mathbf{1}^{\mathsf{T}}\mathbf{b} = 1$ and $\mathbf{b} \geq \mathbf{0}$. A special case of this is the *risk parity portfolio*, for which $b_i = \frac{1}{N}$.

Using that $\nabla \mathcal{R}_{c\mathbf{x}} = \nabla \mathcal{R}_{\mathbf{x}}$ for homogeneous $\mathcal{R}_{\mathbf{x}}$ of degree one and $c \in \mathbb{R}$, we can see that if $\mathbf{x}_{\mathrm{rb}}$ is a risk budgeting portfolio, then so is $c\mathbf{x}_{\mathrm{rb}}$. In some solution approaches it is useful to define a risk-normalized variable $\mathbf{y} = \mathbf{x}/\mathcal{R}_{\mathbf{x}}$, and get

$$\mathbf{y} \circ \nabla \mathcal{R}_{\mathbf{y}} = \mathbf{b} \tag{9.3}$$

Working with this form is typically easier. After solving these equations, we can recover the portfolio $\mathbf{x}$ by normalizing again.

9.3 Risk budgeting with variance

If we use portfolio variance as risk measure, i. e., set $\mathcal{R}_{\mathbf{x}} = \mathbf{x}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{x}$, then (9.2) will become

$$\mathbf{x} \circ \mathbf{\Sigma}\mathbf{x} = \mathbf{b} \circ \mathbf{x}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{x}, \tag{9.4}$$

and (9.3) will become

$$\mathbf{y} \circ \mathbf{\Sigma}\mathbf{y} = \mathbf{b} \tag{9.5}$$

Solving this system is inherently a non-convex problem because it has quadratic equality constraints. However, we can still formulate convex optimization problems to find the solution.

In general, however, we have to be aware that the conditions defining the risk budgeting portfolio are very restrictive. Therefore we have very limited possibility to further constrain a risk budgeting problem, without making the risk budgeting portfolio infeasible.

9.3.1 Convex formulation using quadratic cone

Suppose we allocate the risk budget b_m for each security i in a group of securities $\mathcal{G}_m$, and we have M such groups. As special cases, we have $M = N$ when each security has a unique risk budget, thus each $\mathcal{G}_m$ contains only one security. In contrast, we have $M = 1$ when all risk budgets $b_i = 1/N$ are the same, leading to the risk parity portfolio.

Let us now focus on group $\mathcal{G}_m$. We can define new variables $\gamma_{\mathrm{u},m}$ and $\gamma_{\mathrm{l},m}$ to bound the risk contributions in this group from above and from below. First we express the risk contributions as the fraction of the total risk $b_m\mathbf{x}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{x}$. We can bound this from above to get the constraint

$$b_m\mathbf{x}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{x} \leq \gamma_{\mathrm{u},m}^2,$$

and model it using the quadratic cone.

Next, we bound the risk contributions from below as

$$x_i(\mathbf{\Sigma}\mathbf{x})_i \geq \gamma_{\mathrm{l},m}^2.$$

Assuming $x_i \geq 0$ and $(\mathbf{\Sigma}\mathbf{x})_i \geq 0$, we can model this using the power cone as $(x_i, (\mathbf{\Sigma}\mathbf{x})_i, \gamma_{\mathrm{l},m}) \in \mathcal{P}_3^{1/2,1/2}$, or equivalently using the rotated quadratic cone as $(x_i/\sqrt{2}, (\mathbf{\Sigma}\mathbf{x})_i/\sqrt{2}, \gamma_{\mathrm{l},m}) \in \mathcal{Q}_{\mathrm{r}}^3$.

Finally, minimizing the difference between the upper and lower bounds, we get the risk budgeting portfolio as the optimal solution. The full optimization problem will look like

$$\begin{array}{lrcl}
\text{minimize} & \sum_{m=1}^{M} \gamma_{\mathrm{u},m} - \gamma_{\mathrm{l},m} & & \\
\text{subject to} & \mathbf{1}^{\mathsf{T}}\mathbf{x} & = & 1, \\
& (\gamma_{\mathrm{u},m}/\sqrt{b_m}, \mathbf{G}^{\mathsf{T}}\mathbf{x}) & \in & \mathcal{Q}^{k+1}, \quad m = 1, \ldots, M, \\
& (x_i, (\mathbf{\Sigma}\mathbf{x})_i, \gamma_{\mathrm{l},m}) & \in & \mathcal{P}_3^{1/2,1/2}, \quad i \in \mathcal{G}_m,\ m = 1, \ldots, M, \\
& \mathbf{x}, \mathbf{\Sigma}\mathbf{x} & \geq & \mathbf{0}, \\
& \gamma_{\mathrm{u},m} & \geq & \gamma_{\mathrm{l},m}, \quad m = 1, \ldots, M,
\end{array} \tag{9.6}$$

where $\mathbf{\Sigma} = \mathbf{G}\mathbf{G}^\mathsf{T}$ for $\mathbf{G} \in \mathbb{R}^{N \times k}$.

Note that we have to be careful adding further constraints to problem (9.6), because these might make the risk budgeting portfolio infeasible.

9.3.2 Convex formulation using exponential cone

There also exists a different convex formulation of the risk budgeting problem, that works not only in the positive orthant, but in any (prespecified) orthant of the portfolio space. This allows us to work with long-short portfolios as well.

Observe that the risk budgeting equations (9.4) are the first-order optimality conditions of the optimization problem

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{\Sigma}\mathbf{x} - c\mathbf{b}^\mathsf{T}\log(\mathbf{z} \circ \mathbf{x}) & \\ \text{subject to} & \mathbf{z} \circ \mathbf{x} \geq \mathbf{0}. & \end{array} \tag{9.7}$$

We can use the parameter c to scale the sum of vector $\mathbf{b}$, leading to different magnitude of the optimal solution. Thus we can directly find a solution that satisfies for example $\sum_i x_i = 1$ or in the long-short case $\sum_i |x_i| = 1$.

Problem (9.7) can be modeled using the exponential cone in the following way:

$$\begin{array}{lrcll} \text{minimize} & s - c\mathbf{b}^\mathsf{T}\mathbf{t} & & & \\ \text{subject to} & (s, 1, \mathbf{G}^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{k+2}, & \\ & (z_i x_i, 1, t_i) & \in & K_\mathrm{exp}, & i = 1, \ldots, N, \\ & \mathbf{z} \circ \mathbf{x} & \geq & \mathbf{0}, & \end{array} \tag{9.8}$$

where $\mathbf{\Sigma} = \mathbf{G}\mathbf{G}^\mathsf{T}$ for $\mathbf{G} \in \mathbb{R}^{N \times k}$.

While this convex approach works with any sign configuration $\mathbf{z} \in \{+1, -1\}^N$ of $\mathbf{x}$, we have to specify $\mathbf{z}$ as parameter, meaning we can still only compute the optimal solution for a given orthant. The reason is that by design, there are solutions to (9.4) in each orthant. There are 2^N (normalized) solutions, one for each possible $\mathbf{z} \circ \mathbf{x} > 0$ [CK20], i. e., one in each orthant.[1] So there is a local optimal solution in each orthant, meaning that the problem is non-convex on the full space.

Also note that in this formulation we cannot add further constraints, because these could alter the optimality conditions (9.4), making the solution invalid as a risk budgeting portfolio.

9.3.3 Mixed integer formulation

In each orthant, the solution to (9.7) can have a different total risk. Therefore we can try to find a risk budgeting portfolio that has low total risk. In order to do this, we can extend problem (9.7) to be mixed integer. This will allow us to optimize over the full space, including all long-short portfolios, without needing to prespecify the signs. After defining separate variables based on Sec. 13.2.1 for the long and the short part of the

[1] Of course the solutions in opposite orthants differ only by global sign, so these can be considered the same solution.

portfolio, we get:

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}\mathbf{x} - c\mathbf{b}^T \log(\mathbf{x}^+ + \mathbf{x}^-) & & \\ \text{subject to} & \mathbf{x}^+, \mathbf{x}^- & \geq & 0, \\ & \mathbf{x} & = & \mathbf{x}^+ - \mathbf{x}^-, \\ & \mathbf{x}^+ & \leq & M\mathbf{z}^+, \\ & \mathbf{x}^- & \leq & M\mathbf{z}^-, \\ & \mathbf{z}^+ + \mathbf{z}^- & \leq & \mathbf{1}, \\ & \mathbf{z}^+, \mathbf{z}^- & \in & \{0,1\}^N. \end{array} \tag{9.9}$$

What makes sure that problem (9.9) will find a low risk solution? The first term of the objective function is the sum of risk budgets, thus in any optimal solution, we must have $\mathbf{x}^T\mathbf{\Sigma}\mathbf{x} = \sqrt{c}$, because the risk budgeting conditions are satisfied. It follows that the second term will decide the optimal objective value. This logarithmic term will become the lowest if the monomial $|\mathbf{x}|^{c\mathbf{b}} = \left(\|\mathbf{x}\|_1 |\hat{\mathbf{x}}|^{\mathbf{b}}\right)^c$ is largest, where $\hat{\mathbf{x}}$ denotes a unit vector. Depending on $\mathbf{b}$, this implies an optimal $\mathbf{x}$ with large 1-norm. It follows that after normalization, this $\mathbf{x}$ will yield a low value for the total risk $\mathbf{x}^T\mathbf{\Sigma}\mathbf{x}$. Note however, that it is not guaranteed to get the lowest risk, because $|\hat{\mathbf{x}}|^{\mathbf{b}}$ can also vary.

A further tradeoff with the mixed integer approach is that we have no performance guarantee, finding the optimal solution can take a lot of time. Most likely we will have to settle with suboptimal portfolios in terms of risk.

9.4 Example

In this example we show how can we find a risk parity portfolio by solving a convex optimization problem in **MOSEK** Fusion.

The input data is again obtained the same way as detailed in Sec. 3.4.2. Here we assume that the covariance matrix estimate $\mathbf{\Sigma}$ of yearly returns is available.

The optimization problem we solve here is (9.8), which we repeat here:

$$\begin{array}{lrcll} \text{minimize} & s - c\mathbf{b}^T\mathbf{t} & & & \\ \text{subject to} & (s, 1, \mathbf{G}^T\mathbf{x}) & \in & \mathcal{Q}_r^{k+2}, & \\ & (z_i x_i, 1, t_i) & \in & K_{\exp}, & i = 1, \ldots, N, \\ & \mathbf{z} \circ \mathbf{x} & \geq & \mathbf{0}, & \end{array} \tag{9.10}$$

We search for a solution in the positive orthant, so we set $\mathbf{z} = \mathbf{1}$. We choose the risk budget vector to be $\mathbf{b} = 1/N$, so that all securities contribute the same amount of risk.

The Fusion model of (9.10):

```
def RiskBudgeting(N, G, b, z, a):
    with Model("RiskBudgeting") as M:
        # Portfolio weights
        x = M.variable("x", N, Domain.unbounded())

        # Auxiliary variables
        t = M.variable("t", N, Domain.unbounded())
        s = M.variable("s", 1, Domain.unbounded())

        # Objective: 1/2 * x'Sx - a * b' @ log(z*x) becomes s - a * t' @ b
```

(continues on next page)

(continued from previous page)

```
        M.objective(ObjectiveSense.Minimize, s - a * (t.T @ b))

        # Bound on risk term
        M.constraint(Expr.vstack(s, 1, G.T @ x), Domain.inRotatedQCone())

        # Bound on log term t <= log(z*x) becomes (z*x, 1, t) in K_exp
        M.constraint(Expr.hstack(Expr.mulElm(z, x),
                                 Expr.constTerm(N, 1.0), t),
                     Domain.inPExpCone())

        # Create DataFrame to store the results.
        columns = ["obj", "risk", "xsum", "bsum"] + \
                  df_prices.columns.tolist()
        df_result = pd.DataFrame(columns=columns)

        # Solve optimization
        M.solve()

        # Save results
        xv = x.level()

        # Check solution quality
        risk_budgets = xv * np.dot(G @ G.T, xv)

        # Renormalize to gross exposure = 1
        xv = xv / np.abs(xv).sum()

        # Compute portfolio metrics
        Gx = np.dot(G.T, xv)
        portfolio_risk = np.sqrt(np.dot(Gx, Gx))

        row = pd.Series([M.primalObjValue(), portfolio_risk,
                         np.sum(z * xv), np.sum(risk_budgets)] + \
                        list(xv), index=columns)
        df_result = df_result.append(row, ignore_index=True)
        row = pd.Series([None] * 4 + list(risk_budgets), index=columns)
        df_result = df_result.append(row, ignore_index=True)

        return df_result
```

The following code defines the parameters, including the matrix $\mathbf{G}$ such that $\mathbf{\Sigma} = \mathbf{G}\mathbf{G}^T$.

```
# Number of securities
N = 8
# Risk budget
b = np.ones(N) / N
# Orthant selector
z = np.ones(N)
```

(continues on next page)

(continued from previous page)

```
# Global setting for sum of b
c = 1
# Cholesky factor of the covariance matrix S
G = np.linalg.cholesky(S)
```

Finally, we produce the optimization results:

```
df_result = RiskBudgeting(N, G, b, z, c)
```

On Fig. 9.1 we can see the portfolio composition, and on Fig. 9.2 the risk contributions of each security.

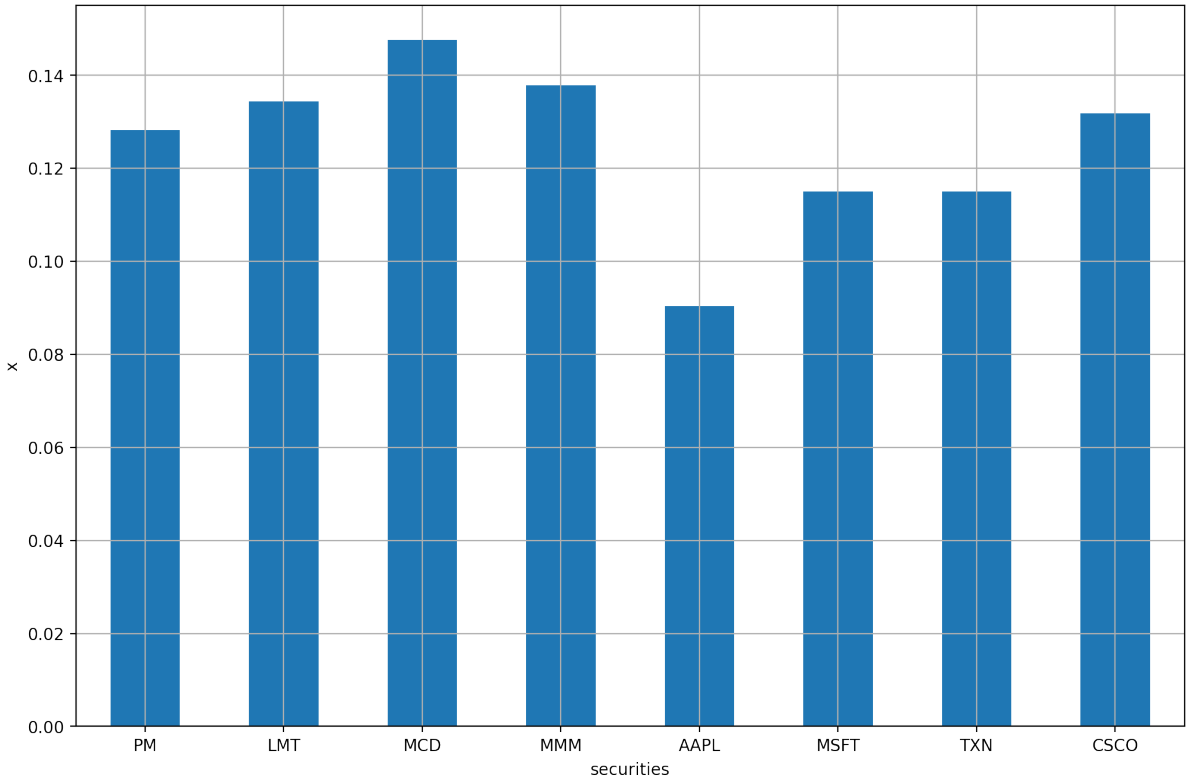

Fig. 9.1: The risk budgeting portfolio.

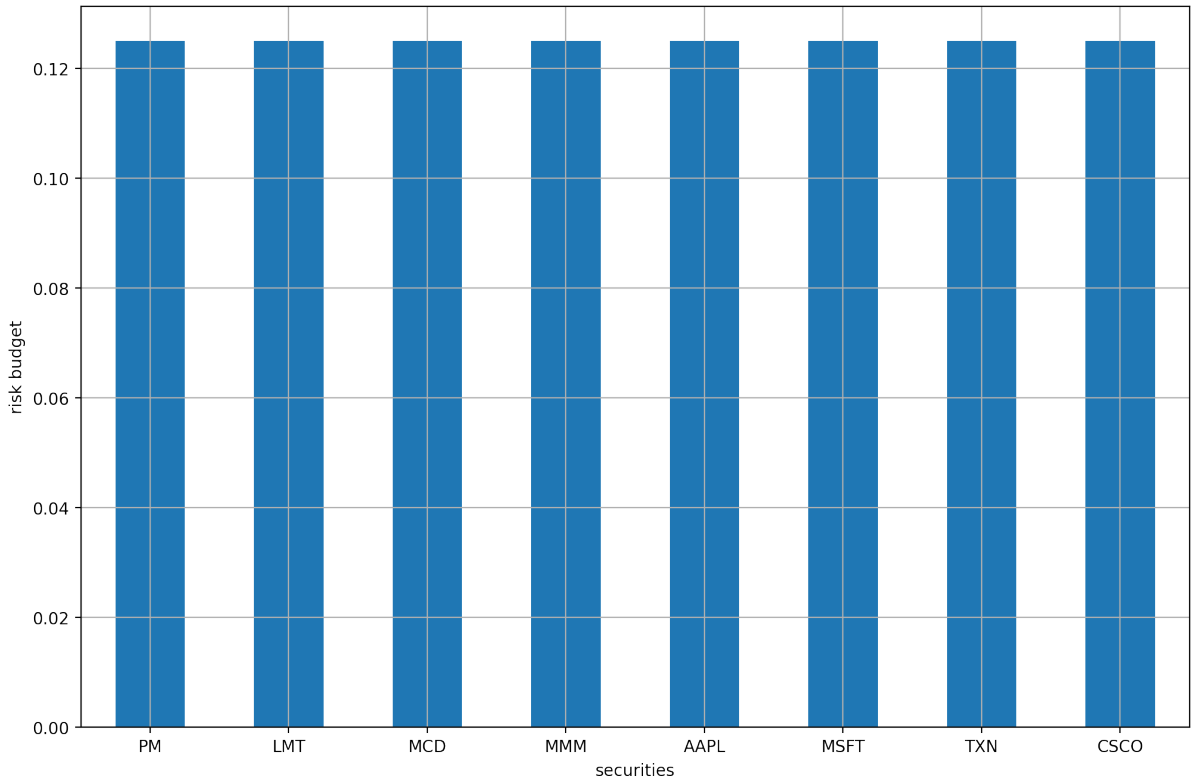

Fig. 9.2: The risk contributions of each security.

Chapter 10

Robust optimization

In chapter Sec. 4 we have discussed in detail, that the inaccurate or uncertain input parameters of a portfolio optimization problem can result in wrong optimal solutions. In other words, the solution is very input sensitive. Robust optimization is another possible modeling tool to overcome this sensitivity. It is a way of handling estimation errors in the optimization problem instead of the input preparation phase. In the most common setup, the parameters are the estimated mean and estimated covariance matrix of the security returns. In robust optimization, we do not compute point estimates of these, but rather an uncertainty set, where the true values lie with certain confidence. A robust portfolio thus optimizes the worst-case performance with respect to all possible parameter values within their corresponding uncertainty sets. [CJPT18, VD09]

10.1 Types of uncertainty

We can form different types of uncertainty sets around the unknown parameters, depending on the nature of the uncertainty, the sensitivity of the solution, and the available information.

- Polytope: If we have a finite number of scenarios, we can form a polytope uncertainty set by taking the convex hull of the scenarios.
- Interval: We can compute a confidence interval for each of the parameters.
- Ellipsoidal region: For vector variables, we can compute confidence regions.

We can model all of these types of uncertainty sets in conic optimization, allowing us to solve robust optimization problems efficiently.

The size of the uncertainty sets reflects the desired level of robustness. For confidence intervals, it is controlled by the confidence level. There is of course a tradeoff, if the uncertainty sets are chosen to be very large, then the resulting portfolios will be very conservative, and they will perform much worse for any given parameter set, than the portfolio designed for that set of parameters. On the other hand, if the size is chosen too small, the resulting portfolio will not be robust enough.

10.2 Uncertainty in security returns

Consider problem (2.3), which we restate here:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^{\mathsf{T}}\mathbf{x} - \frac{\delta}{2}\mathbf{x}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\mathbf{x} & = & 1. \end{array} \tag{10.1}$$

Assume that the vector of mean returns $\boldsymbol{\mu}$ belongs to the elliptical uncertainty set

$$\mathcal{U}_\mu = \{\boldsymbol{\mu} \mid (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^{\mathsf{T}} Q^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}_0) \leq \gamma^2\}, \tag{10.2}$$

where Q is a known positive semidefinite matrix. Then the worst case expected portfolio return will be

$$\min_{\boldsymbol{\mu} \in \mathcal{U}_\mu} \boldsymbol{\mu}^{\mathsf{T}}\mathbf{x} = \boldsymbol{\mu}_0^{\mathsf{T}}\mathbf{x} - \gamma\sqrt{\mathbf{x}^{\mathsf{T}} Q \mathbf{x}}.$$

It follows that the robust version of (10.1) becomes

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}_0^{\mathsf{T}}\mathbf{x} - \gamma\sqrt{\mathbf{x}^{\mathsf{T}} Q \mathbf{x}} - \frac{\delta}{2}\mathbf{x}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\mathbf{x} & = & 1. \end{array} \tag{10.3}$$

10.3 Uncertainty in the factor model

In the article [GI03], the authors consider a factor model on security returns, treat its parameters as uncertain using ellipsoidal uncertainty sets defined as confidence regions, and formulate robust portfolio optimization problems.

Assume that for a random security return vector R_t and factor return vector F_t at time t, the factor model takes the form

$$R_t = \mu + \beta F_t + \theta_t, \tag{10.4}$$

where μ is the vector of mean security returns, β is the factor loading matrix, and θ_t is the vector of residual returns. We also assume an exact factor model (see Sec. 5), meaning that the factor returns and residual returns are assumed to be independent, and the residual covariance matrix D of θ_t is diagonal. Moreover, $\mathbb{E}(F_t) = \mathbf{0}$, so the factors carry no information on the mean returns. A final assumption in this section is that the factor covariance matrix $Q \in \mathbb{R}^{K \times K}$ is known exactly. This requirement will be relaxed later.

The portfolio mean return and portfolio variance will be $\mathbb{E}(R_{\mathbf{x}}) = \mu^{\mathsf{T}}\mathbf{x}$ and $\mathrm{Var}(R_{\mathbf{x}}) = \mathbf{x}^{\mathsf{T}}(\beta Q \beta^{\mathsf{T}} + D)\mathbf{x}$. If we compute estimates $\boldsymbol{\mu}$, $\boldsymbol{\beta}$, and $\mathbf{D}$ of the above quantities, we can reformulate problem (2.1):

$$\begin{array}{lrcl} \text{minimize} & t_1 + t_2 & & \\ \text{subject to} & \boldsymbol{\mu}^{\mathsf{T}}\mathbf{x} & \geq & r_{\min}, \\ & \mathbf{x}^{\mathsf{T}}\mathbf{D}\mathbf{x} & \leq & t_2, \\ & \mathbf{x}^{\mathsf{T}}\boldsymbol{\beta} Q \boldsymbol{\beta}^{\mathsf{T}}\mathbf{x} & \leq & t_1, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{10.5}$$

10.3.1 Uncertainty sets

Instead of computing (10.5) using estimates, we assume variables $\boldsymbol{\mu}$, $\boldsymbol{\beta}$, and $\mathbf{D}$ to lie in the following uncertainty sets:

- The diagonal elements σ_θ^2 of the matrix $\mathbf{D}$ can take values in an uncertainty interval $[\underline{\sigma}_\theta^2, \overline{\sigma}_\theta^2]$:

$$\mathcal{U}_D = \{\mathbf{D} \mid \operatorname{diag}(\mathbf{D}) \in [\underline{\sigma}_\theta^2, \overline{\sigma}_\theta^2]\} \tag{10.6}$$

- The factor loadings matrix $\boldsymbol{\beta}$ belongs to the elliptical uncertainty set

$$\mathcal{U}_\beta = \{\boldsymbol{\beta} \mid \boldsymbol{\beta} = \boldsymbol{\beta}_0 + \mathbf{B}, \|\mathbf{B}_i\|_{\mathbf{G}} \leq \rho_i, i = 1, \ldots, N\} \tag{10.7}$$

  where $\mathbf{B}_i$ is row i of the matrix $\mathbf{B}$, and $\|\mathbf{b}\|_{\mathbf{G}} = \sqrt{\mathbf{b}^\mathsf{T}\mathbf{G}\mathbf{b}}$ denotes the elliptic norm of $\mathbf{b}$ with respect to the positive definite matrix $\mathbf{G} \in \mathbb{R}^{K \times K}$.

- The vector $\boldsymbol{\mu}$ of mean returns is assumed to lie in the uncertainty interval

$$\mathcal{U}_\mu = \{\boldsymbol{\mu} \mid \boldsymbol{\mu} = \boldsymbol{\mu}_0 + \mathbf{m}, |\mathbf{m}| \leq \gamma\} \tag{10.8}$$

10.3.2 Robust problem formulation

The goal of robust portfolio selection is then to select portfolios that perform well for all parameter values that constitute these sets of uncertainty. In other words, we are looking for worst case optimal solutions, formulated as minmax optimization problems. Then we can state the robust mean-variance optimization problem:

$$\begin{array}{lrcl} \text{minimize} & t_1 + t_2 & & \\ \text{subject to} & \min\limits_{\boldsymbol{\mu} \in \mathcal{U}_\mu} \boldsymbol{\mu}^\mathsf{T}\mathbf{x} & \geq & r_{\min}, \\ & \max\limits_{\mathbf{D} \in \mathcal{U}_D} \mathbf{x}^\mathsf{T}\mathbf{D}\mathbf{x} & \leq & t_2, \\ & \max\limits_{\boldsymbol{\beta} \in \mathcal{U}_\beta} \mathbf{x}^\mathsf{T}\boldsymbol{\beta}Q\boldsymbol{\beta}^\mathsf{T}\mathbf{x} & \leq & t_1, \\ & \mathbf{x} & \in & \mathcal{F}. \end{array} \tag{10.9}$$

To convert the minmax problem (10.9) into a conic optimization problem, we first need to represent the worst case expected portfolio return and portfolio variance using conic constraints:

- We can evaluate the worst case expected portfolio return:

$$\min_{\boldsymbol{\mu} \in \mathcal{U}_\mu} \boldsymbol{\mu}^\mathsf{T}\mathbf{x} = \boldsymbol{\mu}_0^\mathsf{T}\mathbf{x} - \gamma^\mathsf{T}|\mathbf{x}|.$$

- We can evaluate the worst case residual portfolio variance:

$$\max_{\mathbf{D} \in \mathcal{U}_D} \mathbf{x}^\mathsf{T}\mathbf{D}\mathbf{x} = \mathbf{x}^\mathsf{T}\operatorname{Diag}(\overline{\sigma}_\theta^2)\mathbf{x}.$$

- We cannot easily evaluate the worst case factor portfolio variance. But we can show that the constraint

$$\max_{\boldsymbol{\beta}\in\mathcal{U}_\beta} \mathbf{x}^\mathsf{T}\boldsymbol{\beta}Q\boldsymbol{\beta}^\mathsf{T}\mathbf{x} \leq t_1$$

is equivalent to

$$(\rho^\mathsf{T}|\mathbf{x}|, t_1, \mathbf{x}) \in \mathcal{H}(\boldsymbol{\beta}_0, Q, \mathbf{G}). \tag{10.10}$$

Relation (10.10) is a shorthand notation for the following: Define $\mathbf{H} = \mathbf{G}^{-1/2}Q\mathbf{G}^{-1/2}$, with spectral decomposition $\mathbf{H} = \mathbf{V}\Lambda\mathbf{V}^\mathsf{T}$, and define $\mathbf{w} = \mathbf{V}^\mathsf{T}\mathbf{H}^{1/2}\mathbf{G}^{1/2}\boldsymbol{\beta}_0^\mathsf{T}\mathbf{x}$. Then there exist $\tau, s, \mathbf{u} \geq 0$ that satisfy the set of conic constraints

$$\begin{array}{rcll} \tau + \mathbf{1}^\mathsf{T}\mathbf{u} & \leq & t_1, & \\ s & \leq & 1/\lambda_{\max}(\mathbf{H}), & \\ (\rho^\mathsf{T}|\mathbf{x}|)^2 & \leq & s\tau, & \\ w_i^2 & \leq & (1 - s\lambda_i)u_i, & i = 1, \ldots, K \end{array}$$

There is also a different but equivalent version of this statement, see [GI03].

10.3.3 Robust conic model

Now we can formulate the robust optimization problem (10.9):

$$\begin{array}{lrcl} \text{minimize} & t_1 + t_2 & & \\ \text{subject to} & \boldsymbol{\mu}_0^\mathsf{T}\mathbf{x} - \gamma^\mathsf{T}|\mathbf{x}| & \geq & r_{\min}, \\ & \mathbf{x}^\mathsf{T}\mathrm{Diag}(\overline{\sigma}_\theta^2)\mathbf{x} & \leq & t_2, \\ & (\rho^\mathsf{T}|\mathbf{x}|, t_1, \mathbf{x}) & \in & \mathcal{H}(\boldsymbol{\beta}_0, Q, \mathbf{G}), \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{10.11}$$

Finally, we can convert (10.11) into conic form by modeling the absolute value based on Sec. 13.1.1 and the quadratic cone based on Sec. 13.1.1:

$$\begin{array}{lrcl} \text{minimize} & t_1 + t_2 & & \\ \text{subject to} & \boldsymbol{\mu}_0^\mathsf{T}\mathbf{x} - \gamma^\mathsf{T}\mathbf{z} & \geq & r_{\min}, \\ & (t_2, \frac{1}{2}, \sqrt{\overline{\sigma}_\theta^2} \circ \mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{N+2}, \\ & (\rho^\mathsf{T}\mathbf{z}, t_1, \mathbf{x}) & \in & \mathcal{H}(\boldsymbol{\beta}_0, Q, \mathbf{G}), \\ & \mathbf{x} & \leq & \mathbf{z}, \\ & \mathbf{x} & \geq & -\mathbf{z}, \\ & \mathbf{x} & \in & \mathcal{F}, \end{array} \tag{10.12}$$

and the constraint $(\rho^\mathsf{T}\mathbf{z}, t_1, \mathbf{x}) \in \mathcal{H}(\boldsymbol{\beta}_0, Q, \mathbf{G})$ can be modeled using the hyperbolic constraint in Sec. 13.1.1:

$$\begin{array}{rcll} \tau + \mathbf{1}^\mathsf{T}\mathbf{u} & \leq & t_1, & \\ s & \leq & 1/\lambda_{\max}(\mathbf{H}), & \\ (s, \tau, \rho^\mathsf{T}\mathbf{z}) & \in & \mathcal{Q}_\mathrm{r}^3, & \\ (1 - s\lambda_i, u_i, w_i) & \in & \mathcal{Q}_\mathrm{r}^3, & i = 1, \ldots, K \end{array} \tag{10.13}$$

where $\tau, s, \mathbf{u} \geq 0$ are new variables, $\mathbf{w} = \mathbf{V}^\mathsf{T}\mathbf{H}^{1/2}\mathbf{G}^{1/2}\boldsymbol{\beta}_0^\mathsf{T}\mathbf{x}$, and λ_i are eigenvalues of $\mathbf{H} = \mathbf{G}^{-1/2}Q\mathbf{G}^{-1/2} = \mathbf{V}\Lambda\mathbf{V}^\mathsf{T}$.

10.3.4 Case of unknown factor covariance

In this section we cover the case when the factor covariance matrix Q is also uncertain, and has the estimate $\mathbf{Q}$. In this case, the robust portfolio optimization problem can still be converted into a conic form, and solved efficiently.

We can give an uncertainty structure to either the factor covariance matrix $\mathbf{Q}$ or its inverse $\mathbf{Q}^{-1}$. Both choices lead to the same worst case portfolio variance constraint. The choice of $\mathbf{Q}^{-1}$ is a bit more restrictive, but allows us to accomodate prior information about the structure of $\mathbf{Q}$. See the details in [GI03]. Here we describe the case of $\mathbf{Q}$.

The matrix estimate $\mathbf{Q}$ has the uncertainty structure

$$\mathcal{U}_Q = \{\mathbf{Q} \mid \mathbf{Q} = \mathbf{Q}_0 + \Delta \geq 0, \Delta = \Delta^\mathsf{T}, \|\mathbf{N}^{-1/2}\Delta\mathbf{N}^{-1/2}\| \leq \zeta\}, \tag{10.14}$$

where $\mathbf{Q}_0 \geq 0$, and the norm is the spectral norm or the Frobenius norm.

Then the worst case factor portfolio variance constraint

$$\max_{\boldsymbol{\beta}\in\mathcal{U}_\beta, \mathbf{Q}\in\mathcal{U}_Q} \mathbf{x}^\mathsf{T}\boldsymbol{\beta}\mathbf{Q}\boldsymbol{\beta}^\mathsf{T}\mathbf{x} \leq t_1$$

is equivalent to

$$(\rho^\mathsf{T}|\mathbf{x}|, t_1, \mathbf{x}) \in \mathcal{H}(\boldsymbol{\beta}_0, \mathbf{Q}_0 + \zeta\mathbf{N}, \mathbf{G}). \tag{10.15}$$

10.4 Parameters

In this section we discuss how the parameters of the uncertainty sets introduced in Sec. 10.3.1 can be computed from market data. Typically the parameters $\boldsymbol{\mu}$, $\boldsymbol{\beta}$, $\mathbf{D}$ are estimated from the security return and factor return data, using multivariate linear regression. We can also compute multidimensional confidence regions with any desired confidence level around the least-squares estimates. These confidence regions become the uncertainty sets in the robust portfolio optimization problem. The regression procedure also yields natural choices for the matrix $\mathbf{G}$ defining the elliptic norm and the bounds ρ, γ, $\overline{\sigma}_\theta^2$. In [GI03], there are two methods discussed to construct the uncertainty sets from data, here we detail only one of them.

In (10.4) the factor model is written for all security at one time instant t. Now we write the model for one security i at all time instants:

$$R_i = \mu_i + \beta_i\mathbf{F} + \theta_i, \quad i = 1, \ldots, N$$

where β_i is row number i of β. We can write this in a shorter form as

$$R_i = \mathbf{A}Y_i + \theta_i, \quad i = 1, \ldots, N$$

where $Y_i = [\mu_i, \beta_i]^\mathsf{T} \in \mathbb{R}^{(K+1)\times 1}$, and $\mathbf{A} = [\mathbf{1}, \mathbf{F}^\mathsf{T}] \in \mathbb{R}^{N\times(K+1)}$.

Assuming that the matrix $\mathbf{A}$ is rank $K+1$, and we have market return series $\mathbf{r}_i$, the theory of ordinary least squares leads to the estimate

$$\bar{\mathbf{y}}_i = [\bar{\boldsymbol{\mu}}_i, \bar{\boldsymbol{\beta}}_i]^\mathsf{T} = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\mathbf{r}_i$$

The ω elliptical confidence region around $\mathbf{y}_i$ is then

$$\mathcal{U}_{y_i}(\omega) = \{\mathbf{y}_i \mid (\bar{\mathbf{y}}_i - \mathbf{y}_i)^\mathsf{T}(\mathbf{A}^\mathsf{T}\mathbf{A})(\bar{\mathbf{y}}_i - \mathbf{y}_i) \leq (K+1)(s_\theta^2)_i c_{K+1}(\omega)\},$$

where $(s_\theta^2)_i = \|\mathbf{r}_i - \mathbf{A}\bar{\mathbf{y}}_i\|^2/(T-K-1)$ is estimate of the error variance $(\sigma_\theta^2)_i$, $c_{K+1}(\omega)$ is the ω critical value, the solution of $\mathcal{F}_F(c_{K+1}) = \omega$, and $\mathcal{F}_F$ is the CDF of the F-distribution with degrees of freedom $(K+1, T-K-1)$.

Then the full ω^N confidence set for $\mathbf{y}$ will be $\mathcal{U}_y(\omega) = \mathcal{U}_{y_1}(\omega) \times \cdots \times \mathcal{U}_{y_N}(\omega)$.

10.4.1 The parameters of $\mathcal{U}_\mu$

If we project $\mathcal{U}_y(\omega)$ along the vector $\boldsymbol{\mu}$, we get the ω^N confidence set (10.8), where

$$\begin{aligned} (\boldsymbol{\mu}_0)_i &= \bar{\boldsymbol{\mu}}_i, \\ \gamma_i &= \sqrt{(K+1)(\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}_{1,1}(s_\theta^2)_i c_{K+1}(\omega)} \end{aligned}$$

10.4.2 The parameters of $\mathcal{U}_\beta$

Let $\mathbf{P} = [\mathbf{0}, \mathbf{I}] \in \mathbb{R}^{K\times(K+1)}$ be the matrix projecting $\mathbf{y}_i$ along $\boldsymbol{\beta}_i$. If we project $\mathcal{U}_y(\omega)$ along $\boldsymbol{\beta}$, then we get the ω^N confidence set (10.7), where

$$\begin{aligned} (\boldsymbol{\beta}_0)_i &= \bar{\boldsymbol{\beta}}_i, \\ \mathbf{G} &= (\mathbf{P}(\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{P}^\mathsf{T})^{-1} \\ &= \mathbf{F}\mathbf{F}^\mathsf{T} - \tfrac{1}{T}(\mathbf{F}\mathbf{1})(\mathbf{F}\mathbf{1})^\mathsf{T}, \\ \rho_i &= \sqrt{(K+1)(s_\theta^2)_i c_{K+1}(\omega)} \end{aligned}$$

10.4.3 The parameters of $\mathcal{U}_D$

It would be natural to choose the confidence interval around $(s_\theta^2)_i$, the estimate of the error variance $(\sigma_\theta^2)_i$, but we only have a single value. It would be possible to use bootstrapping to construnct an upper bound this way, but it can be computationally expensive.

Since we only require an estimate of the worst case error variance $(\overline{\sigma}_\theta^2)_i$ for the robust optimization problem, it is cheaper to use any reasonable estimate for this purpose.

10.4.4 The parameters of $\mathcal{U}_Q$

In case the factor covariance matrix is not known, we can construct its uncerainty region also from data. According to [GI03], the ω^K confidence set (10.14) can be parameterized the following way:

$$\begin{aligned} \mathbf{Q}_0 &= \mathbf{Q}_{\mathrm{ML}}, \\ \mathbf{N} &= \mathbf{Q}_{\mathrm{ML}}, \\ \zeta &= \eta/(1-\eta), \end{aligned}$$

where $\mathbf{Q}_{\mathrm{ML}} = \mathbf{G}/(T-1)$ is the maximum likelihood estimate (MLE) of Q computed from factor return data, and η is the unique solution of

$$\mathcal{F}_\Gamma(1+\eta) - \mathcal{F}_\Gamma(1-\eta) = \omega, \tag{10.16}$$

where $\mathcal{F}_\Gamma$ is the CDF of a $\Gamma(\frac{T+1}{2}, \frac{T-1}{2})$ random variable[1], and ω is the desired confidence level.[2] Note that equation (10.16) restricts ω^K to be at most $\mathcal{F}_\Gamma(2)^K$, which depends

[1] In the distribution $\Gamma(a, b)$, a is the *shape* parameter and b is the *rate* parameter.

[2] There is a unique solution because the right-hand side of (10.16) is monotonic on $[0, 1]$, and takes values also in $[0, 1]$.

on the number of data samples T. However, this limitation is not very restrictive in practice.[3]

10.5 Example

Here we show a code example of the robust optimization problem (10.3), that we restate here:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}_0^T\mathbf{x} - \gamma\sqrt{\mathbf{x}^T Q\mathbf{x}} - \frac{\delta}{2}\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x} & & \\ \text{subject to} & \mathbf{1}^T\mathbf{x} & = & 1. \end{array} \tag{10.17}$$

We start at the point where data is already prepared, and we show the optimization model. This example considers an elliptical uncertainty region around the expected return vector. If we compute the worst case portfolio return in this case, there will be two terms with quadratic expressions. The first will be $\gamma\sqrt{\mathbf{x}^T Q\mathbf{x}}$, where γ controls the size of the uncertainty region. If $\gamma = 0$, then we get back the original, non-robust MVO problem. The second quadratic expression $\frac{\delta}{2}\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x}$ models the portfolio risk, and δ is the risk aversion coefficient.

We can model both terms using the second-order cones. For the term with square-root, the quadratic cone is more appropriate, while the portfolio variance term can be modeled using the rotated quadratic cone. We substitute the square-root term with the new variable $s_q = \sqrt{\mathbf{x}^T Q\mathbf{x}}$, then the objective of the problem will be

```
# Objective
delta = M.parameter()
wc_return = x.T @ mu0 - gamma * sq
M.objective('obj', ObjectiveSense.Maximize, wc_return - delta * s)
```

Assuming that $Q = \mathbf{G}_Q\mathbf{G}_Q^T$, the square root term can be modeled as

```
# Robustness
M.constraint('robustness', Expr.vstack(sq, GQ.T @ x), Domain.inQCone())
```

Similarly, we substitute the risk term with $s = \frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}\mathbf{x}$, and assuming $\boldsymbol{\Sigma} = \mathbf{G}\mathbf{G}^T$, we model the risk as

```
# Risk constraint
M.constraint('risk', Expr.vstack(s, 1, G.T @ x),
                     Domain.inRotatedQCone())
```

The full model would look like the following:

```
with Model("Robust") as M:
    # Variables
    # The variable x is the fraction of holdings in each security.
    # It is restricted to be positive, which imposes no short-selling.
    x = M.variable("x", N, Domain.greaterThan(0.0))

    # The variable s models the portfolio risk term.
```

(continues on next page)

[3] Assuming $K = 40$ factors, having $T \geq 50$ data samples allows for a confidence level of $\omega^K \geq 0.995$.

(continued from previous page)

```
    s = M.variable("s", 1, Domain.greaterThan(0.0))

    # The variable sq models the robustness term.
    sq = M.variable("sq", 1, Domain.greaterThan(0.0))

    # Budget constraint
    M.constraint('budget', Expr.sum(x) == 1.0)

    # Objective
    delta = M.parameter()
    wc_return = x.T @ mu0 - gamma * sq
    M.objective('obj', ObjectiveSense.Maximize, wc_return - delta * s)

    # Robustness
    M.constraint('robustness', Expr.vstack(sq, GQ.T @ x), Domain.inQCone())

    # Risk constraint
    M.constraint('risk', Expr.vstack(s, 1, G.T @ x),
                         Domain.inRotatedQCone())

    # Create DataFrame to store the results. Last security names
    # (the factors) are removed.
    columns = ["delta", "obj", "return", "risk"] + \
              df_prices.columns.tolist()
    df_result = pd.DataFrame(columns=columns)
    for d in deltas:
        # Update parameter
        delta.setValue(d)

        # Solve optimization
        M.solve()

        # Save results
        portfolio_return = mu0 @ x.level() - gamma * sq.level()[0]
        portfolio_risk = np.sqrt(2 * s.level()[0])
        row = pd.Series([d, M.primalObjValue(),
                         portfolio_return, portfolio_risk] + \
                        list(x.level()), index=columns)
        df_result = df_result.append(row, ignore_index=True)
```

Finally, we compute the efficient frontier in the following points:

```
deltas = np.logspace(start=-1, stop=2, num=20)[::-1]
```

If we plot the efficient frontier on Fig. 10.1, and the portfolio composition on Fig. 10.2 we can compare the results obtained with and without using robust optimization.

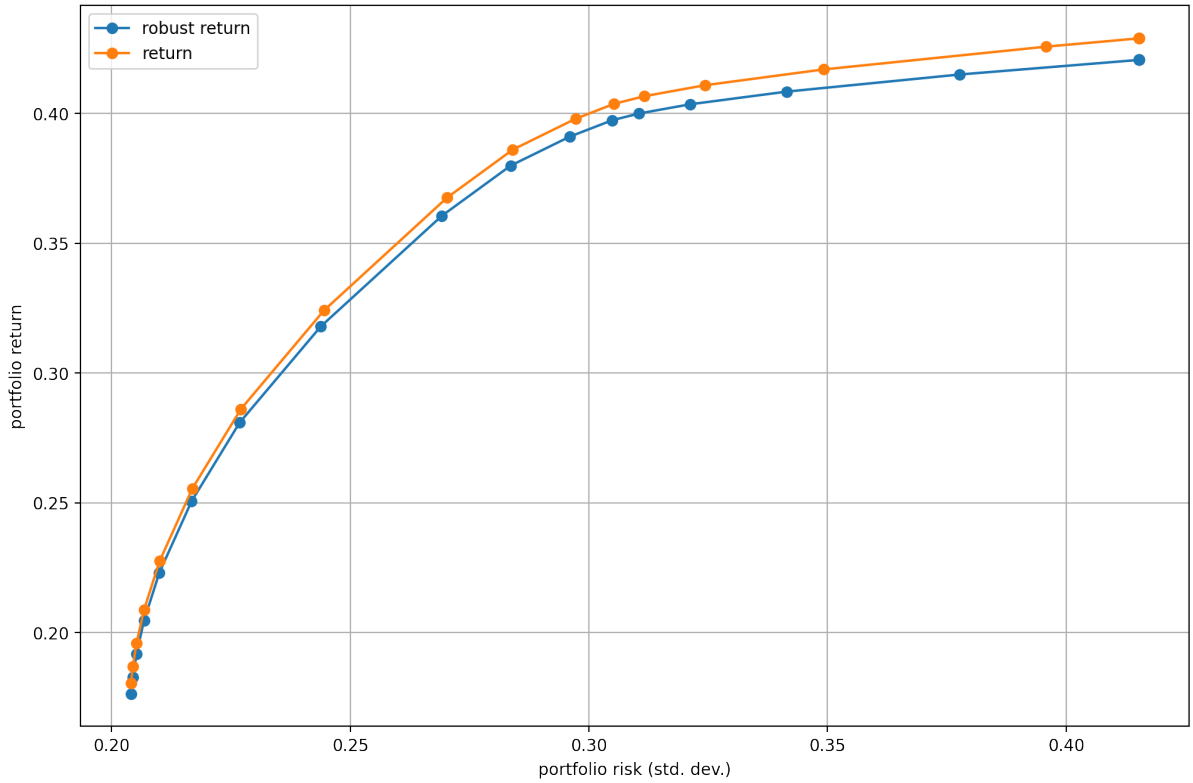

Fig. 10.1: The efficient frontier.

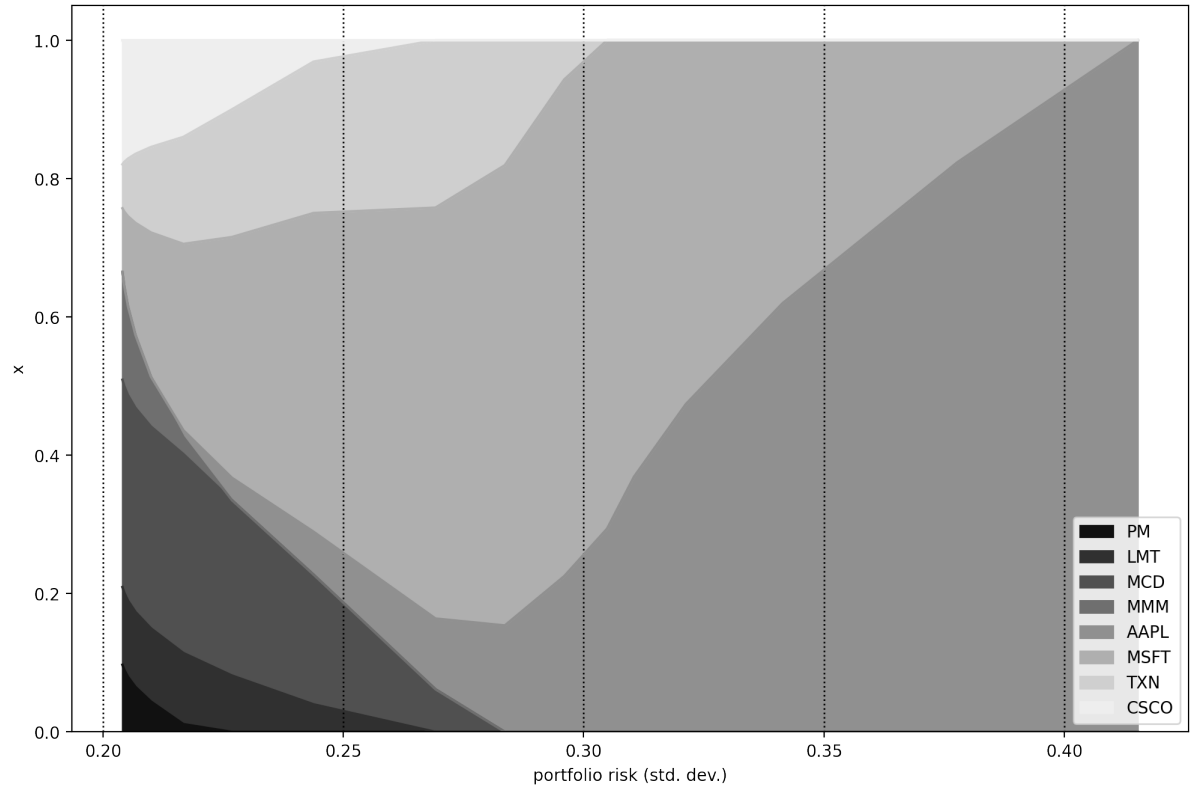

Fig. 10.2: Portfolio composition $\mathbf{x}$ with varying level if risk-aversion δ.

Chapter 11

Multi-period portfolio optimization

Multi-period portfolio optimization is an extension of the single-period MVO problem. Its objective is to select a sequence of trades over a series of time periods. In contrast to repeated single period optimizations, it takes into account recourse and updated information while planning trades for the subsequent period.

While multi period models are more complex, they can help us to naturally handle various inter-temporal effects:

- Different time scales: We can model multiple, possibly conflicting return predictions on different time scales.
- Transaction costs: Considers whether our trades in the current period put us in favourable position to trade in future periods.
- Time-varying constraints: For example, reducing the leverage bound will likely incur lower trading cost if done over several period.
- Time-varying return forecasts: For example, we may assign an exponential decay-rate to the return predictions.
- Future events: We can act on or exploit known future events that will change risk, liquidity, or volume, etc.

Dynamic programming is a common method to deal with multi-period problems, however it is challenging due to the "curse of dimensionality". Therefore in practice, we use approximation methods, like approximate dynamic programming. Fortunately, the resulting performance loss is typically negligible in practical scenarios.

In this chapter we describe a practical multi-period portfolio optimization problem based on [BBD+17].

11.1 Single-period model

First we show the optimization problem for a single period to introduce the model elements and notation.

11.1.1 Definitions

Suppose we have N securities and a cash account (labeled as security $N+1$), and we are at the beginning of period t. Let $\mathbf{h}_t$ be the *portfolio in dollars* in period t, and let $v_t = \mathbf{1}^\mathsf{T}\mathbf{h}_t$ be its total value. Then we can normalize $\mathbf{h}_t$ to get the *portfolio in weights* as $\mathbf{x}_t = \mathbf{h}_t/v_t$.

Let $\mathbf{u}_t$ be the *dollar trade vector* and $\mathbf{z}_t$ be the *normalized trade vector* in period t. Then the *post-trade portfolio* will be $\mathbf{h}_t + \mathbf{u}_t$ or $\mathbf{x}_t + \mathbf{z}_t$. In the following, we will use only the normalized quantities.

11.1.2 Problem statement

At period t, we wish to determine the trade vector $\mathbf{z}_t$. The starting portfolio $\mathbf{x}_t$ is known. All other parameters (future quantities) have to be estimated. These are the mean return $\boldsymbol{\mu}_t$, the risk function $\boldsymbol{\mathcal{R}}_t$, the trading cost function $\boldsymbol{\phi}_t^{\text{trade}}$, and the holding cost function $\boldsymbol{\phi}_t^{\text{hold}}$.

The single-period optimization problem will be

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}_t^\mathsf{T}(\mathbf{x}_t+\mathbf{z}_t) - \delta_t\boldsymbol{\mathcal{R}}_t(\mathbf{x}_t+\mathbf{z}_t) - & & \\ & -\boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t) - \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{z}_t) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{z}_t + \boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t) + \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{z}_t) & = & 0, \\ & \mathbf{z}_t & \in & \mathcal{Z}_t, \\ & \mathbf{x}_t+\mathbf{z}_t & \in & \mathcal{X}_t, \end{array} \tag{11.1}$$

where δ_t is the risk aversion parameter, and $\mathcal{Z}_t$ and $\mathcal{X}_t$ represent the set of trading and the set of holding constraints respectively.

11.1.3 Post-trade computations

After (11.1) is solved, we obtain the optimal solution $\mathbf{z}_t^*$ together with the optimal cash component $z_{t,N+1}^*$. However, the latter might be inaccurate, because it is determined using estimated costs. After knowing the realized cost functions ϕ_t^{hold} and ϕ_t^{trade}, we can use them to determine the accurate cash component from the self-financing condition:

$$z_{t,N+1} = -\mathbf{1}^\mathsf{T}\mathbf{z}_{t,1..N}^* - \phi_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t^*) - \phi_t^{\text{trade}}(\mathbf{z}_t^*).$$

However, this more precise cash quantity may result in small constraint violations.

Finally, we hold the post-trade portfolio until the end of period t. Then we get the realized return μ_t, and can compute the new total portfolio value v_{t+1}. This allows us to compute the realized portfolio return using the self financing condition:

$$R_{\mathbf{x},t} = \frac{v_{t+1}-v_t}{v_t} = \mu^\mathsf{T}(\mathbf{x}_t+\mathbf{z}_t) - \phi_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t) - \phi_t^{\text{trade}}(\mathbf{z}_t) \tag{11.2}$$

The portfolio composition at the end of period t can be computed using the realized return μ_t:

$$\mathbf{x}_{t+1} = \frac{1}{1 + R_{\mathbf{x},t}}(\mathbf{1} + \mu_t) \circ (\mathbf{x}_t + \mathbf{z}_t). \tag{11.3}$$

This dynamics equation ensures that if $\mathbf{1}^\mathsf{T}\mathbf{x}_t = 1$ then we have $\mathbf{1}^\mathsf{T}\mathbf{x}_{t+1} = 1$.

11.1.4 Simplified problem

We can make two approximations to simplify problem (11.1).

- Cost terms: We can omit the cost terms from the self-financing condition, reducing it to $\mathbf{1}^\mathsf{T}\mathbf{z}_t = 0$. In this case the costs are still taken into account in the objective as penalty terms. The benefit will be that the equation $\mathbf{1}^\mathsf{T}(\mathbf{x}_t + \mathbf{z}_t) = 1$ will hold exactly in all cases.

- Returns: If returns are very small in any period, then we can assume them to be zero. Then we can approximate the dynamics equation (11.3) to be $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{z}_t$.

11.1.5 Holding constraints

Holding constraints are imposed on the post-trade portfolio $\mathbf{x}_t + \mathbf{z}_t$, that is held until the end of period t. Such constraints can be for example the long only constraint, leverage, position size limit, concentration limit, minimum cash balance, factor neutrality, etc.

Some of these constraints can actually be approximations of constraints on $\mathbf{x}_{t+1}$. When we assume small returns, these approximations will also become more accurate.

11.1.6 Trading constraint

Trading constraints are imposed on the trade vector $\mathbf{z}_t$. Such constraints can be for example the turnover, trading limits, restrictions on the direction of trade, etc.

Here we assume the trading cost function ϕ_t^{trade} to be separable, i. e., $\phi_t^{\text{trade}}(\mathbf{x}) = \sum_{i=1}^N \phi_{t,i}^{\text{trade}}(x_i)$. We also assume that there are no transaction costs associated with the cash account.

11.2 Multi-period model

Here we extend the model (11.1) further by considering a planning horizon of H periods. In this context, we would like to find the trade vector $\mathbf{z}_0$ for the first period by optimizing $\mathbf{z}_t$ for all periods $t = 0, \ldots, H-1$.[1] The initial portfolio weight vector $\mathbf{x}_0$ is known. The rest of the (future) input quantities $\boldsymbol{\mu}_t$, $\boldsymbol{\mathcal{R}}_t$, $\boldsymbol{\phi}_t^{\text{trade}}$, and $\boldsymbol{\phi}_t^{\text{hold}}$ have to be estimated for all

[1] Only the first period trades are executed. However it is still useful to compute the trades for all periods, to be sure not to carry out trades now that will put us in a bad position in the future.

periods. Then we have to solve the following problem:

$$\begin{array}{lrcl} \text{maximize} & \sum_{t=0}^{H-1} \boldsymbol{\mu}_t^\mathsf{T}(\mathbf{x}_t+\mathbf{z}_t) - \delta_t \boldsymbol{\mathcal{R}}_t(\mathbf{x}_t+\mathbf{z}_t) - & & \\ & -\boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t) - \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{z}_t) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{z}_t + \boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t) + \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{z}_t) & = & 0, \\ & \mathbf{z}_t & \in & \mathcal{Z}_t, \\ & \mathbf{x}_t+\mathbf{z}_t & \in & \mathcal{X}_t, \\ & \boldsymbol{\mu}^\mathsf{T}(\mathbf{x}_t+\mathbf{z}_t) - \boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_t+\mathbf{z}_t) - \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{z}_t) & = & R_{\mathbf{x},t}, \\ & 1/(1+R_{\mathbf{x},t})(\mathbf{1}+\mu_t)\circ(\mathbf{x}_t+\mathbf{z}_t) & = & \mathbf{x}_{t+1}, \end{array} \tag{11.4}$$

where we repeat all constraints for all $t = 0, \ldots, H-1$.

The multi-period objective is the sum of the single-period objectives for each period t. Here we implicitly assumed that the returns are independent random variables, so the variance of their sum is the sum of the variances.

Unfortunately (11.4) is not a convex problem, because of the dynamic equation. Therefore we have to make a simplification.

11.2.1 Simplified model

We can apply the simplification of returns discussed in section Sec. 11.1.4 and use the approximate dynamics equation $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{z}_t$. This also allows us to eliminate the trade variables $\mathbf{z}_t$ from the model, and arrive at a simpler form:

$$\begin{array}{lrcl} \text{maximize} & \sum_{t=0}^{H-1} \boldsymbol{\mu}_t^\mathsf{T}\mathbf{x}_{t+1} - \delta_t \boldsymbol{\mathcal{R}}_t(\mathbf{x}_{t+1}) - & & \\ & -\boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_{t+1}) - \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{x}_{t+1}-\mathbf{x}_t) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x}_{t+1} + \boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_{t+1}) + \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{x}_{t+1}-\mathbf{x}_t) & = & \mathbf{1}^\mathsf{T}\mathbf{x}_t, \\ & \mathbf{x}_{t+1}-\mathbf{x}_t & \in & \mathcal{Z}_t, \\ & \mathbf{x}_{t+1} & \in & \mathcal{X}_t, \end{array} \tag{11.5}$$

where we repeat all constraints for all $t = 0, \ldots, H-1$.

Note that this simplification does not mean we disregard returns, we only approximate the portfolio dynamics between periods. The returns are still taken into account in the objective.

11.2.2 Extension

An extension of (11.5) is to specify the terminal portfolio after the last period. Choice for the terminal portfolio could depend on the use case. Some reasonable choices could be

- The predicted benchmark portfolio, which ensures that the optimal portfolio we get will not deviate much from the benchmark.
- 100% cash, which ensures that the optimal portfolio we get will not be expensive to liquidate.

Since we never execute the optimal trades for all periods, the purpose of multiperiod planning with a terminal constraint is to avoid moving towards undesired positions, not to actually end up holding the terminal portfolio.

Problem (11.5) with the terminal constraint added will look like:

$$\begin{array}{lrcl} \text{maximize} & \sum_{t=0}^{H-1} \boldsymbol{\mu}_t^\mathsf{T}\mathbf{x}_{t+1} - \delta_t\boldsymbol{\mathcal{R}}_t(\mathbf{x}_{t+1}) - & & \\ & -\boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_{t+1}) - \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{x}_{t+1}-\mathbf{x}_t) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x}_{t+1} + \boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_{t+1}) + \boldsymbol{\phi}_t^{\text{trade}}(\mathbf{x}_{t+1}-\mathbf{x}_t) & = & \mathbf{1}^\mathsf{T}\mathbf{x}_t, \\ & \mathbf{x}_{t+1}-\mathbf{x}_t & \in & \mathcal{Z}_t, \\ & \mathbf{x}_{t+1} & \in & \mathcal{X}_t, \\ & \mathbf{x}_H & = & \mathbf{x}^{\text{term}}, \end{array} \tag{11.6}$$

where $\mathbf{x}^{\text{term}}$ is the desired terminal portfolio, and we repeat all constraints for all $t = 0, \ldots, H-1$.

11.3 Example

Here we show a code example of the multiperiod portfolio optimization problem (11.5). In this setup, we will use the following specification:

- The risk function will be the variance: $\boldsymbol{\mathcal{R}}_t(\mathbf{x}_{t+1}) = \mathbf{x}_{t+1}^\mathsf{T}\boldsymbol{\Sigma}_t\mathbf{x}_{t+1}$.
- The trading cost function will include a linear term, and a market impact term: $\boldsymbol{\phi}_t^{\text{trade}}(\mathbf{x}_{t+1}-\mathbf{x}_t) = \sum_{i=1}^N a_{t,i}|x_{t+1,i}-x_{t,i}| + \tilde{b}_{t,i}|x_{t+1,i}-x_{t,i}|^{3/2}$. The $a_{t,i}$ are the coefficients of the linear cost term, and the $b_{t,i}$ are the coefficients of the market impact cost term. $b_{t,i} = \tilde{b}_{t,i}\sigma_{t,i}/\left(\frac{q_{t,i}}{V_t}\right)^{1/2}$, where $\tilde{b}_{t,i} = 1$, $\sigma_{t,i}$ is the volatility of security i in period t, and $\frac{q_{t,i}}{V_t}$ is the normalized dollar volume of security i in period t.
- There will be no holding cost term: $\boldsymbol{\phi}_t^{\text{hold}}(\mathbf{x}_{t+1}) = 0$.
- The set of trading constraints will be empty: $\mathcal{Z}_t = \{\}$.
- The set of holding constraints will contain the long only constraint: $\mathcal{X}_t = \{\mathbf{x}_{t+1} \geq 0\}$.

Based on the above, we will solve the following multiperiod optimization problem:

$$\begin{array}{lrcl} \text{maximize} & \sum_{t=0}^{H-1} \boldsymbol{\mu}_t^\mathsf{T}\mathbf{x}_{t+1} - \delta_t\mathbf{x}_{t+1}^\mathsf{T}\boldsymbol{\Sigma}_t\mathbf{x}_{t+1} - & & \\ & -\left(\sum_{i=1}^N a_{t,i}|x_{t+1,i}-x_{t,i}| + b_{t,i}|x_{t+1,i}-x_{t,i}|^{3/2}\right) & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}(\mathbf{x}_{t+1}-\mathbf{x}_t) + \left(\sum_{i=1}^N a_{t,i}|x_{t+1,i}-x_{t,i}| + \tilde{b}_{t,i}|x_{t+1,i}-x_{t,i}|^{3/2}\right) & = & 0, \\ & \mathbf{x}_{t+1} & \geq & 0, \end{array} \tag{11.7}$$

where we repeat all constraints for all $t = 0, \ldots, H-1$.

Then we rewrite (11.7) into conic form:

$$\begin{array}{lrcll} \text{maximize} & \sum_{t=0}^{H-1} \boldsymbol{\mu}_t^\mathsf{T}\mathbf{x}_{t+1} - \delta_t s_t - & & & \\ & -\left(\sum_{i=1}^N a_{t,i}v_{t,i} + b_{t,i}w_{t,i}\right) & & & \\ \text{subject to} & (s_t, 0.5, \mathbf{G}_t^\mathsf{T}\mathbf{x}_{t+1}) & \in & \mathcal{Q}_{\text{r}}^{N+2}, & \\ & |x_{t+1}-x_t| & \leq & v_t, & \\ & (w_{t,i}, 1, x_{t+1,i}-x_{t,i}) & \in & \mathcal{P}_3^{2/3,1/3}, \quad i = 1, \ldots, N & \\ & \mathbf{1}^\mathsf{T}\mathbf{x}_{t+1} + \left(\sum_{i=1}^N a_{t,i}v_{t,i} + b_{t,i}w_{t,i}\right) & = & \mathbf{1}^\mathsf{T}\mathbf{x}_t, & \\ & \mathbf{x}_{t+1} & \geq & 0, & \end{array} \tag{11.8}$$

where we repeat all constraints for all $t = 0, \ldots, H-1$.

Finally, we implement (11.7) in Fusion API. The following auxiliary functions encapsulate specific parts of modeling, and help us to write a clearer code:

```
def absval(M, x, z):
    M.constraint(z >= x)
    M.constraint(z >= -x)

def norm1(M, x, t):
    z = M.variable(x.getSize(), Domain.greaterThan(0.0))
    absval(M, x, z)
    M.constraint(Expr.sum(z) == t)
```

Next we present the Fusion model, built inside a function, that we can easily call later.

```
def multiperiod_mvo(N, T, m, G, x_0, delta, a, b):
    with Model("Multiperiod") as M:
        # Variables
        # - Portfolio
        x = M.variable("x", [N, T], Domain.greaterThan(0.0))
        # - Risk
        s = M.variable("s", T)
        # - Linear transaction cost
        v = M.variable("v", [N, T])
        # - Market impact cost
        w = M.variable("w", [N, T])

        # Constraint
        M.constraint("budget", Expr.sum(x, 0) == np.ones(T))

        # Objective
        M.objective("obj", ObjectiveSense.Maximize,
            Expr.add([ x[:, t].T @ m[t] - delta[t] * s[t] - \
                       v[:, t].T @ a[:, t] - w[:, t].T @ b[:, t]
                       for t in range(T)]))

        # Conic constraints related to the objective
        for t in range(T):
            xt = x[:, t]
            xtprev = x_0 if t == 0 else x[:, t - 1]
            xtdiff = xt - xtprev
            M.constraint(f'risk_{t}',
                Expr.flatten(Expr.vstack(s[t], 0.5, G[t].T @ xt)),
                Domain.inRotatedQCone()
            )
            absval(M, xtdiff, v[:, t])
            M.constraint(f'market_impact_{t}',
                Expr.hstack(w[:, t], Expr.constTerm(N, 1.0), xtdiff),
                Domain.inPPowerCone(2 / 3)
            )
```

(continues on next page)

(continued from previous page)

```
        # Solve the problem
        M.solve()

        # Get the solution values
        x_value = x.level().reshape(N, T)

        return x_value
```

Next, we define the parameters. Here we assume that we have an estimate of the average daily volume and daily volatility (std. dev.) for all periods.

```
# Number of securities
N = 8

# Number of periods
T = 10

# Initial weights
x_0 = np.array([1] * N) / N
portfolio_value = 10**8

# Transaction cost
a = 0.05 * np.ones((N, T))

# Market impact
beta = 3 / 2
b = 1
# The variable vol contains the volume estimates
rel_volume = [v / portfolio_value for v in vol] # Relative volume.
# The variable vty contains the volatility estimates
impact_coef = \
    np.vstack([(b * v / r**(beta - 1)).to_numpy()
    for v, r in zip(vty, rel_volume)]).T
```

Assuming also that we have a list of estimates of the yearly mean return and covariance matrix for each trading period, we compute the matrix $\mathbf{G}$ such that $\mathbf{\Sigma} = \mathbf{G}\mathbf{G}^\mathsf{T}$ for all periods, using Cholesky factorization.

```
# S is the list of covariance estimates
G = [np.linalg.cholesky(s) for s in S]
```

Finally, we run the optimization with the risk aversion parameter $\delta = 10$ for each period.

```
delta = np.array([10] * T)
# m is the list of mean return estimates
x = multiperiod_mvo(N, T, m, G, x_0, delta, a, impact_coef)
```

Chapter 12

Regression and regularization

Regression is statistical tool used in many disciplines, in particular in finance and investing. It is used to determine the type and strength of the function relationship between statistical variables. In some cases when data is scarce or has errors, regularization is also useful. In this chapter we introduce regression and regularizaton in the context of conic optimization along the lines of [SHAD13], and show how to apply these methods for portfolio optimization.

12.1 Linear regression

The most basic regression problem assumes a linear relationship $\mathbf{y} = \mathbf{X}\mathbf{w} + \varepsilon$ between N explanatory variables $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{T\times N}$ and dependent variable $\mathbf{y} \in \mathbb{R}^T$, and attempts to estimate $\mathbf{w}$. This is commonly done using the method of *ordinary least squares (OLS)*, which minimizes the norm of the residual term $\mathbf{y}-\mathbf{X}\mathbf{w}$. In other words, we are looking for the coefficient vector $\mathbf{w}$ that is the optimal solution of the unconstrained least squares optimization problem

$$\text{minimize} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2. \tag{12.1}$$

The same problem can also be written with squared norm:

$$\text{minimize} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2. \tag{12.2}$$

The geometric interpretation is that we are looking for the vector in the column space of $\mathbf{X}$ that is closest to $\mathbf{y}$. If the problem is unconstrained, the solution is simply an orthogonal projecton of $\mathbf{y}$ onto the column space of $\mathbf{X}$.[1]

[1] OLS is an approximate solution to the overdetermined system of linear equations $\mathbf{X}\mathbf{w} \approx \mathbf{y}$, where $\mathbf{w}$ is the unknown.

12.1.1 Assumptions

Let $\mathbf{X}_i$ and $\mathbf{X}_j$ be two observations. For the OLS method to give meaningful results, we have to impose some assumptions:

- Exogeneity: $\mathbb{E}(\varepsilon \mid \mathbf{X}_i) = 0$, meaning that the error term is orthogonal to the explanatory variables, so there are no endogeneous drivers for $\mathbf{X}$ in the model. This also implies $\mathrm{Cov}(\varepsilon, \mathbf{X}) = 0$.
- No autocorrelation: The error terms are uncorrelated between observations. This implies that the off-diagonal of $\mathrm{Cov}(\varepsilon_i, \varepsilon_j \mid \mathbf{X}_i, \mathbf{X}_j)$ is zero.
- Homoscedasticity: The error term has the same variance for all observations, i. e., for any values of the explanatory variables. This implies that $\mathrm{Var}(\varepsilon \mid \mathbf{X}_i) = \sigma^2\mathbf{1}$.
- No linear dependence: The observation matrix $\mathbf{X}$ must have full column rank.

We also assume that $\mathbf{X}$ has at least as many rows as columns, i.e., $T \geq N$.

Some of the above assumptions can be relaxed by using specific extensions of the OLS method. However, these extensions might be more complex and might have a greater data requirement in order to produce an equally precise model.

12.1.2 Solution

Normal equations

If problem (12.1) is unconstrained, we can also derive its explicit solution called the *normal equations*:

$$\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} = \mathbf{X}^\mathsf{T}\mathbf{y}. \tag{12.3}$$

Note however, that this should be used only as a theoretical result. In practice, it can be numerically unstable because of the matrix inversion step when solving for $\mathbf{w}$.

Conic optimization

If we convert the problem (12.2) into a conic optimization problem, we can not only solve it in a more efficient and numerically stable way, but will also be able to extend the problem with constraints. Here we state the conic equivalent of (12.1):

$$\begin{array}{lrcl} \text{minimize} & t & & \\ \text{subject to} & (t, \mathbf{y} - \mathbf{X}\mathbf{w}) & \in & \mathcal{Q}^{T+1}. \end{array} \tag{12.4}$$

For (12.2) we use the rotated quadratic cone,

$$\begin{array}{lrcl} \text{minimize} & t & & \\ \text{subject to} & (t, 0.5, \mathbf{y} - \mathbf{X}\mathbf{w}) & \in & \mathcal{Q}_\mathrm{r}^{T+2}. \end{array} \tag{12.5}$$

Let us now extend (12.5) with constraints:

$$\begin{array}{lrcl} \text{minimize} & t & & \\ \text{subject to} & (t, 0.5, \mathbf{y} - \mathbf{X}\mathbf{w}) & \in & \mathcal{Q}_\mathrm{r}^{T+2}, \\ & \mathbf{1}^\mathsf{T}\mathbf{w} & = & 1, \\ & \mathbf{w} & \geq & 0. \end{array} \tag{12.6}$$

12.1.3 Portfolio optimization as regression problem

Portfolio optimization problems can also take the form of (12.2), if instead of working with the covariance matrix of securities $\mathbf{\Sigma}$, we work directly with the security return observation matrix (scenario matrix) $\mathbf{R} \in \mathbb{R}^{N \times T}$ defined in Sec. 3.1, where T is the number of observations. We assume here that the matrix is centered. Then for the sample covariance matrix, we have

$$\mathbf{\Sigma} = \frac{1}{T-1}\mathbf{R}\mathbf{R}^\mathsf{T}. \tag{12.7}$$

Equation (12.7) gives us a way to define the matrix $\mathbf{X}$ in problem (12.2): $\mathbf{X} = \mathbf{R}^\mathsf{T}/\sqrt{T-1}$.

Then we can write a simple benchmark relative optimization problem that minimizes tracking error as

$$\begin{array}{lrcl} \text{minimize} & \|\mathbf{R}^\mathsf{T}\mathbf{x} - \mathbf{r}_{\text{bm}}\|_2^2, & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{12.8}$$

where we omitted the factor $1/\sqrt{T-1}$ from the objective, and $\mathbf{r}_{\text{bm}} = \mathbf{R}^\mathsf{T}\mathbf{x}_{\text{bm}}$ is the benchmark return series, assuming $\mathbf{x}$ and $\mathbf{x}_{\text{bm}}$ represent the same security universe. In general, we could also assume that $\mathbf{x}$ and $\mathbf{x}_{\text{bm}}$ represent different security universes with different return data $\mathbf{R}$ and $\mathbf{R}_{\text{bm}}$. That is equivalent to assuming the same (larger) universe for both, with different security weights fixed to zero.

The conic form of (12.8) becomes

$$\begin{array}{lrcl} \text{minimize} & t & & \\ \text{subject to} & (t, 0.5, \mathbf{R}^\mathsf{T}\mathbf{x} - \mathbf{r}_{\text{bm}}) & \in & \mathcal{Q}_{\text{r}}^{T+2}, \\ & \mathbf{1}^\mathsf{T}\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{12.9}$$

12.2 Regularization

In some cases the data matrix $\mathbf{X}$ in (12.2) has nearly dependent columns, i.e., we have high correlations between the corresponding explanatory variables. We can also have less observations than variables ($T < N$). In these situations, the results can be unreliable and highly sensitive to small perturbations and rounding errors. We discuss this topic in detail in the context of portfolio optimization, see in Sec. 4. To handle these issues, we can also use regularization techniques to stabilize the computational results.

Regularization typically means that we add a penalty term to our objective function, that helps to direct the solution procedure towards optimal solutions $\mathbf{w}$ that have favorable properties. This way of adjusting the objective can also be viewed as a *soft constraint*, meaning that we express a preference regarding the quality of the optimal solution, but without using exact bounds.

12.2.1 Example penalty terms

Here we discuss three examples of regularization terms, two of which are commonly applied. Let $\mathbf{w}_0$ be a specific solution vector with desired properties.

Ridge regularization

Ridge regularization adds a quadratic penalty term to the objective function:

$$\text{minimize} \quad \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\Gamma(\mathbf{w} - \mathbf{w}_0)\|_2^2. \tag{12.10}$$

If $\Gamma = \mathbf{I}$, then it is called L2 regularization, because it simply gives preference to solution vectors with smaller norms.

We can derive the normal equations also in this case, and we can observe that there is a relation between ridge regularization and covariance shrinkage:

$$(\mathbf{X}^T\mathbf{X} + \lambda\Gamma^T\Gamma)\mathbf{w} = \mathbf{X}^T\mathbf{y} + \lambda\Gamma^T\Gamma\mathbf{w}_0. \tag{12.11}$$

The conic equivalent of (12.10) will introduce an extra quadratic cone constraint for the regularization term:

$$\begin{array}{lrcl} \text{minimize} & t + \lambda u & & \\ \text{subject to} & (t, 0.5, \mathbf{X}\mathbf{w} - \mathbf{y}) & \in & \mathcal{Q}_{\text{r}}^{T+2}, \\ & (u, 0.5, (\Gamma\mathbf{w} - \mathbf{w}_0)) & \in & \mathcal{Q}_{\text{r}}^{N+2}. \end{array} \tag{12.12}$$

LASSO regularization

LASSO regularization adds a linear penalty term to the objective function:

$$\text{minimize} \quad \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w} - \mathbf{w}_0\|_1. \tag{12.13}$$

It is also called L1 regularization or sparse regularization, because it gives preference to sparse solutions, i.e., ones with few nonzeros.

The conic equivalent of (12.13) will model the 1-norm constraint for the regularization term, as described in Sec. 13.1.1:

$$\begin{array}{lrcl} \text{minimize} & t + \lambda u & & \\ \text{subject to} & (t, 0.5, \mathbf{X}\mathbf{w} - \mathbf{y}) & \in & \mathcal{Q}_{\text{r}}^{T+2}, \\ & \mathbf{z} - (\mathbf{w} - \mathbf{w}_0) & \geq & 0, \\ & \mathbf{z} + (\mathbf{w} - \mathbf{w}_0) & \geq & 0, \\ & \mathbf{1}^T\mathbf{z} - u & = & 0. \end{array} \tag{12.14}$$

The 3/2 regularization

The penalty term discussed here is not common, but has applications in modeling market impact cost (see Sec. 6.3). We add a subquadratic penalty term to the objective function:

$$\text{minimize} \quad \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w} - \mathbf{w}_0\|_{3/2}^{3/2}, \tag{12.15}$$

where $\|\mathbf{w} - \mathbf{w}_0\|_{3/2}^{3/2} = \sum_i |w_i - w_{0,i}|^{3/2}$.

The conic equivalent of (12.15) uses the power cone for the regularization term,

$$\begin{array}{lrcl} \text{minimize} & t + \lambda\sum_i u_i & & \\ \text{subject to} & (t, 0.5, \mathbf{X}\mathbf{w} - \mathbf{y}) & \in & \mathcal{Q}_{\text{r}}^{T+2}, \\ & (u_i, 1, w_i - w_{i,0}) & \in & \mathcal{P}_3^{2/3,1/3}, \ i = 1, \ldots, N. \end{array} \tag{12.16}$$

12.2.2 Regularization in portfolio optimization

In the context of portfolio optimization, examples of regularization terms appear when we penalize trading costs or do robust optimization. Here we show an example of a trading cost penalty. For robust optimization, see Sec. 10.

Let $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ denote the traded amount with respect to the initial holdings $\mathbf{x}_0$. In constrast to Sec. 6, where costs are part of the budget constraint, making the portfolio self-financing, here we implement trading costs as penalties in the objectve function.

Suppose we have linear trading costs, expressed by the penalty $\|\tilde{\mathbf{x}}\|_1$ and market impact costs, expressed by the penalty $\|\tilde{\mathbf{x}}\|_{3/2}^{3/2}$. Then problem (12.8) with trading costs will be

$$\begin{array}{lrcl} \text{minimize} & \|\mathbf{R}^T\mathbf{x} - \mathbf{r}_{\text{bm}}\|_2^2 + \lambda_1\|\tilde{\mathbf{x}}\|_1 + \lambda_2\|\tilde{\mathbf{x}}\|_{3/2}^{3/2}, & & \\ \text{subject to} & \mathbf{1}^T\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{12.17}$$

The conic form of (12.17) becomes

$$\begin{array}{lrcl} \text{minimize} & t + \lambda_1 u + \lambda_2 \sum_i v_i & & \\ \text{subject to} & (t, 0.5, \mathbf{R}^T\mathbf{x} - \mathbf{r}_{\text{bm}}) & \in & \mathcal{Q}_{\text{r}}^{T+2}, \\ & \mathbf{z} - \tilde{\mathbf{x}} & \geq & 0, \\ & \mathbf{z} + \tilde{\mathbf{x}} & \geq & 0, \\ & \mathbf{1}^T\mathbf{z} - u & = & 0, \\ & (v_i, 1, \tilde{x}_i) & \in & \mathcal{P}_3^{2/3,1/3}, \ i = 1, \ldots, N, \\ & \mathbf{1}^T\mathbf{x} & = & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{12.18}$$

12.3 Example

In this chapter we present the example (12.18) as **MOSEK** Fusion code. We will use yearly linear return scenario data from the same eight stocks used in other examples. The benchmark will be the return series of the SPY ETF.

12.3.1 Data preparation

We generate the data the same way as in Sec. 3.4 up to the point where we have the expected yearly logarithmic return vector $\boldsymbol{\mu}_h^{\log}$ and covariance matrix $\boldsymbol{\Sigma}_h^{\log}$. Then we generate Monte Carlo scenarios using these parameters, and convert these to linear returns. The number T is the number of observations we generate.

```
# Generate logarithmic return observations assuming normal distribution
scenarios_log = \
    np.random.default_rng().multivariate_normal(m_log, S_log, T)

# Convert logarithmic return observations to linear return observations
scenarios_lin = np.exp(scenarios_log) - 1
```

Next, we center the data and separate security data from the benchmark data. Note that we also scale the returns by the factor of $1/\sqrt{T-1}$ coming from (12.7). We do this because the relative magnitude of the tracking error term matters compated to the

magnitude of the regularization terms. We would like to control these relative magnitudes only by setting λ_1 and λ_2 parameters.

```
# Center the return data
centered_return = scenarios_lin - scenarios_lin.mean(axis=0)

# Security return scenarios (scaled)
security_return = scenarios_lin[:, :N] / np.sqrt(T - 1)

# Benchmark return scenarios (scaled)
benchmark_return = scenarios_lin[:, -1] / np.sqrt(T - 1)
```

12.3.2 Optimization model

We start by defining the variables. The variable $\mathbf{x}$ represents the portfolio, t is the tracking error, u and $\mathbf{v}$ are the penalty terms. The latter three helps us to model the objective terms as constraints. We also add the usual budget constraint here.

```
# Variables
# The variable x is the fraction of holdings in each security.
# It is restricted to be positive - no short-selling.
x = M.variable("x", N, Domain.greaterThan(0.0))
xt = x - x0

# The variable t models the OLS objective function term (tracking error).
t = M.variable("t", 1, Domain.unbounded())
# The variables u and v model the regularization terms
# (transaction cost penalties).
u = M.variable("u", 1, Domain.unbounded())
v = M.variable("v", N, Domain.unbounded())

# Budget constraint
M.constraint('budget', Expr.sum(x) == 1.0)
```

The objective function will thus be the sum of the above variables:

```
# Objective
penalty_lin = lambda_1 * u
penalty_32 = lambda_2 * Expr.sum(v)
M.objective('obj', ObjectiveSense.Minimize,
                   t + penalty_lin + penalty_32)
```

The constraints corresponding to the penalty terms are modeled by the following rows:

```
# Constraints for the penalties
norm1(M, xt, u)
M.constraint('market_impact',
             Expr.hstack(v, Expr.constTerm(N, 1.0), xt),
             Domain.inPPowerCone(1.0 / beta))
```

The `norm1` custom function definition will be given below where the full code of the Fusion model is presented.

Finally, we implement the tracking error constraint:

```
# Constraint for the tracking error
residual = R.T @ x - r_bm
M.constraint('tracking_error',
             Expr.vstack(t, 0.5, residual),
             Domain.inRotatedQCone())
```

The complete Fusion model will then be the following code:

```
def absval(M, x, z):
    M.constraint(z >= x)
    M.constraint(z >= -x)

def norm1(M, x, t):
    z = M.variable(x.getSize(), Domain.greaterThan(0.0))
    absval(M, x, z)
    M.constraint(Expr.sum(z) == t)

def MinTrackingError(N, R, r_bm, x0, lambda_1, lambda_2, beta=1.5):
    with Model("Regression") as M:
        # Variables
        # The variable x is the fraction of holdings in each security.
        # It is restricted to be positive - no short-selling.
        x = M.variable("x", N, Domain.greaterThan(0.0))
        xt = x - x0

        # The variable t models the OLS objective function term
        # (tracking error).
        t = M.variable("t", 1, Domain.unbounded())
        # The variables u and v model the regularization terms
        # (transaction cost penalties).
        u = M.variable("u", 1, Domain.unbounded())
        v = M.variable("v", N, Domain.unbounded())

        # Budget constraint
        M.constraint('budget', Expr.sum(x) == 1.0)

        # Objective
        penalty_lin = lambda_1 * u
        penalty_32 = lambda_2 * Expr.sum(v)
        M.objective('obj', ObjectiveSense.Minimize,
                           t + penalty_lin + penalty_32)

        # Constraints for the penalties
        norm1(M, xt, u)
        M.constraint('market_impact',
                     Expr.hstack(v, Expr.constTerm(N, 1.0), xt),
                     Domain.inPPowerCone(1.0 / beta))

        # Constraint for the tracking error
```

(continues on next page)

(continued from previous page)

```
        residual = R.T @ x - r_bm
        M.constraint('tracking_error',
                     Expr.vstack(t, 0.5, residual),
                     Domain.inRotatedQCone())

        # Create DataFrame to store the results.
        # Last security name (the SPY) is removed.
        columns = ["track_err", "lin_tcost", "mkt_tcost"] + \
                  df_prices.columns[:N].tolist()
        df_result = pd.DataFrame(columns=columns)

        # Solve optimization
        M.solve()

        # Save results
        tracking_error = t.level()[0]
        linear_tcost = u.level()[0]
        market_impact_tcost = np.sum(v.level())
        row = pd.Series(
                 [tracking_error, linear_tcost, market_impact_tcost] + \
                 list(x.level()), index=columns)
        df_result = pd.concat([df_result, pd.DataFrame([row])],
                              ignore_index=True)

        return df_result
```

12.3.3 Results

Here we show how to run the optimization model, and present the results. First, we set the number of stocks, then we set the penalty coefficients λ_1 and λ_2 to desired values, finally we also define the initial holdings vector $\mathbf{x}_0$ to be the equal weighted portfolio in this case.

```
N = 8
lambda_1 = 0.0001
lambda_2 = 0.0001
x0 = np.ones(N) / N
```

Then we run the optimization model:

```
df_result = MinTrackingError(N, security_return.T,
                             benchmark_return, x0, lambda_1, lambda_2)
```

The optimal portfolio composition is

$$\mathbf{x} = [0.0900, 0.1667, 0.1681, 0.1712, 0.0481, 0.1014, 0.1173, 0.1373].$$

Chapter 13

Appendix

13.1 Conic optimization refresher

In this section we give a short summary about conic optimization. Read more about the topic from a mathematical and modeling point of view in our other publication, the **MOSEK** Modeling Cookbook [MOSEKApS24].

Conic optimization is a class of convex optimization problems, which contains and generalizes in an unified way many specific and well known convex models. These models include *linear optimization* (LO), *quadratic optimization* (QO), *quadratically constrained quadratic optimization* (QCQO), *second-order cone optimization* (SOCO), and *semidefinite optimization* (SDO). The general form of a conic optimization problem is

$$\begin{array}{lll} \text{maximize} & & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x}+\mathbf{b} & \in \quad \mathcal{K}. \end{array} \tag{13.1}$$

where $\mathcal{K}$ is a product of the following basic types of cones:

- **Linear cone**:

  $$\mathbb{R}^n,\ \mathbb{R}^n_+,\ \{0\}.$$

  Linear cones model all traditionally LO problems.

- **Quadratic cone and rotated quadratic cone**:

  The quadratic cone is the set

  $$\mathcal{Q}^n = \left\{ x \in \mathbb{R}^n \middle| x_1 \geq \sqrt{x_2^2 + \cdots + x_n^2} \right\}.$$

  The rotated quadratic cone is the set

  $$\mathcal{Q}^n_{\mathrm{r}} = \left\{ x \in \mathbb{R}^n \middle| 2x_1x_2 \geq x_3^2 + \cdots + x_n^2,\ x_1, x_2 \geq 0 \right\}.$$

  Modeling with these two cones cover the class of SOCO problems, which include all traditionally QO and QCQO problems as well. See in Sec. 13.1.2 for more details.

- **Primal power cone**:

  $$\mathcal{P}^{\alpha,1-\alpha}_n = \left\{ x \in \mathbb{R}^n \middle| x_1^\alpha x_2^{1-\alpha} \geq \sqrt{x_3^2 + \cdots + x_n^2},\ x_1, x_2 \geq 0 \right\},$$

  or its dual cone.

- **Primal exponential cone**:

$$K_{\exp} = \left\{ x \in \mathbb{R}^3 \middle| x_1 \geq x_2 \exp\left(\frac{x_3}{x_2}\right),\ x_1, x_2 \geq 0 \right\},$$

  or its dual cone.

- **Semidefinite cone**:

$$\mathcal{S}_+^n = \left\{ X \in \mathbb{R}^{n \times n} \middle| X \text{ is symmetric positive semidefinite} \right\}.$$

  Semidefinite cones model all traditionally SDO problems.

Each of these cones allow formulating different types of convex constraints.

13.1.1 Selection of conic constraints

In the following we will list the constraints appearing in financial context in this book and show how can we convert them to conic form.

Maximum function

We can model the maximum constraint $\max(x_1, x_2, \ldots, x_n) \leq c$ using linear constraints by introducing an auxiliary variable t:

$$\begin{array}{rcl} t & \leq & c, \\ t & \geq & x_1, \\ & \vdots & \\ t & \geq & x_n. \end{array} \tag{13.2}$$

For instance, we could write a series of constraints $\max(x_i, 0) \leq c_i,\ i = 1, \ldots, n$ as

$$\mathbf{t} \leq \mathbf{c}, \quad \mathbf{t} \geq \mathbf{x}, \quad \mathbf{t} \geq \mathbf{0}, \tag{13.3}$$

where $\mathbf{t}$ is an n-dimensional vector.

Positive and negative part

A special case of modeling the maximum function is to model the positive part x^+ and negative part x^- of a variable x. We define these as $x^+ = \max(x, 0)$ and $x^- = \max(-x, 0)$.

We can model them explicitly based on Sec. 13.1.1 by relaxing the definitions to inequalities $x^+ \geq \max(x, 0)$ and $x^- \geq \max(-x, 0)$, or we can also model them implicitly by the following set of constraints:

$$\begin{array}{rcl} x & = & x^+ - x^-, \\ x^+,\ x^- & \geq & 0. \end{array} \tag{13.4}$$

Note however, that in either case a freedom remains in the magnitude of x^+ and x^-. This is because in the explicit case we relaxed the equalities, and in the impicit case only the difference of the variables is constrained. In other words it will be possible for x^+ and x^- to be both positive, allowing optimal solutions where $x^+ = \max(x, 0)$ and $x^- = \max(-x, 0)$ does not hold. We could ensure that only either x^+ or x^- is positive

and never both by stating the complementarity constraint $x^+x^- = 0$ (or in the vector case $\langle \mathbf{x}^+, \mathbf{x}^- \rangle = 0$), but unfortunately such an equality constraint is non-convex and cannot be modeled.

We can do workarounds to ensure that equalities $x^+ = \max(x, 0)$ and $x^- = \max(-x, 0)$ hold in the optimal solution. One approach is to penalize the magnitude of these two variables, so that if both are positive in any solution, then the solver could always improve the objective by reducing them until either one becomes zero. Another possible workaround is to formulate a mixed integer problem; see Sec. 13.2.1.

Absolute value

We can model the absolute value constraint $|x| \leq c$ using the maximum function by observing that $|x| = \max(x, -x)$ (see Sec. 13.1.1):

$$-c \leq x \leq c. \tag{13.5}$$

Another possibility is to model it using the quadratic cone:

$$(c, x) \in \mathcal{Q}^2. \tag{13.6}$$

Sum of largest elements

The *sum of the m largest elements* of a vector $\mathbf{x}$ is the optimal solution of the LO problem

$$\begin{array}{ll} \text{maximize} & \mathbf{x}^\mathsf{T}\mathbf{z} \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{z} = m, \\ & \mathbf{0} \leq \mathbf{z} \leq \mathbf{1}. \end{array} \tag{13.7}$$

Here $\mathbf{x}$ cannot be a variable, because that would result in a nonlinear objective. Let us take the dual of problem (13.7):

$$\begin{array}{lrcl} \text{minimize} & mt + \mathbf{1}^\mathsf{T}\mathbf{u} & & \\ \text{subject to} & \mathbf{u} + t & \geq & \mathbf{x}, \\ & \mathbf{u} & \geq & \mathbf{0}. \end{array} \tag{13.8}$$

Problem (13.8) is actually the same as $\min_t \; mt + \sum_i \max(0, x_i - t)$. In this case $\mathbf{x}$ can be a variable, and thus it can also be optimized.

Linear combination of largest elements

We can slightly extend the problem in Sec. 13.1.1 such that $\mathbf{z}$ can have an upper bound $\mathbf{c} \geq \mathbf{0}$, and there is a real number $0 \leq b \leq c_{\text{sum}}$ instead of the integer m, where $c_{\text{sum}} = \sum_i c_i$:

$$\begin{array}{ll} \text{maximize} & \mathbf{x}^\mathsf{T}\mathbf{z} \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{z} = c_{\text{sum}} - b, \\ & \mathbf{0} \leq \mathbf{z} \leq \mathbf{c}. \end{array} \tag{13.9}$$

This has the optimal objective $c_{i_b}^{\text{frac}} x_{i_b} + \sum_{i > i_b} c_i x_i$, where i_b is such that $\sum_{i=1}^{i_b - 1} c_i < b \leq \sum_{i=1}^{i_b} c_i$, and $c_{i_b}^{\text{frac}} = \sum_{i=1}^{i_b} c_i - b < c_{i_b}$.

If we take the dual of problem (13.9), we get:

$$\begin{array}{lrcl} \text{minimize} & (c_{\text{sum}} - b)t + \mathbf{c}^{\mathsf{T}}\mathbf{u} & & \\ \text{subject to} & \mathbf{u} + t & \geq & \mathbf{x}, \\ & \mathbf{u} & \geq & \mathbf{0}, \end{array} \tag{13.10}$$

which is the same as $\min_t\ (c_{\text{sum}} - b)t + \sum_i c_i \max(0, x_i - t)$.

Manhattan norm (1-norm)

Let $\mathbf{x} \in \mathbb{R}^n$ and observe that $\|\mathbf{x}\|_1 = |x_1| + \cdots + |x_n|$. Then we can model the *Manhattan norm* or *1-norm* constraint $\|\mathbf{x}\|_1 \leq c$ by modeling the absolute value for each coordinate:

$$-\mathbf{z} \leq \mathbf{x} \leq \mathbf{z}, \quad \sum_{i=1}^{n} z_i = c, \tag{13.11}$$

where $\mathbf{z}$ is an auxiliary variable.

Euclidean norm (2-norm)

Let $\mathbf{x} \in \mathbb{R}^n$ and observe that $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$. Then we can model the *Euclidean norm* or *2-norm* constraint $\|\mathbf{x}\|_2 \leq c$ using the quadratic cone:

$$(c, \mathbf{x}) \in \mathcal{Q}^{n+1}. \tag{13.12}$$

Hyperbolic constraint

Let $\mathbf{x} \in \mathbb{R}^n$ and observe that $\|\mathbf{x}\|_2^2 = \mathbf{x}^{\mathsf{T}}\mathbf{x} = x_1^2 + \cdots + x_n^2$. Then we can model the *hyperbolic* constraint $\mathbf{x}^{\mathsf{T}}\mathbf{x} \leq yz$, where $y, z \geq 0$ using the rotated quadratic cone:

$$(y, z, \mathbf{x}) \in \mathcal{Q}_{\text{r}}^{n+2}. \tag{13.13}$$

Squared Euclidean norm

We can model the *squared Euclidean norm* or *sum-of-squares* constraint $\|\mathbf{x}\|_2^2 \leq c$ using the hyperbolic constraint in Sec. 13.1.1 and the rotated quadratic cone:

$$\left(c, \tfrac{1}{2}, \mathbf{x}\right) \in \mathcal{Q}_{\text{r}}^{n+2}. \tag{13.14}$$

Quadratic form

Let $\mathbf{x} \in \mathbb{R}^n$ and let $\mathbf{Q} \in \mathcal{S}_+^n$, i. e., a symmetric positive semidefinite matrix. Then we can model the *quadratic form* constraint $\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{Q}\mathbf{x} \leq c$ either using the quadratic cone or the rotated quadratic cone. To see this, observe that there exists a matrix $\mathbf{G} \in \mathbb{R}^{n \times k}$ such that $\mathbf{Q} = \mathbf{G}\mathbf{G}^{\mathsf{T}}$. Of course this decomposition can be done in many ways, so the matrix $\mathbf{G}$ is not unique. The most interesting cases are when $k \ll n$ or $\mathbf{G}$ is very sparse, because these make the optimization problem much easier to solve numerically (see in Sec. 13.1.2).

Most common ways to compute the matrix $\mathbf{G}$ are the following:

- Cholesky decomposition: $\mathbf{Q} = \mathbf{C}\mathbf{C}^\mathsf{T}$, where $\mathbf{C}$ is a lower triangular matrix with nonnegative entries in the diagonal. From this decomposition we have $\mathbf{G} = \mathbf{C} \in \mathbb{R}^{n\times n}$.

- Eigenvalue decomposition: $\mathbf{Q} = \mathbf{V}\mathbf{D}\mathbf{V}^\mathsf{T}$, where the diagonal matrix $\mathbf{D}$ contains the (nonnegative) eigenvalues of $\mathbf{Q}$ and the unitary matrix $\mathbf{V}$ contains the corresponding eigenvectors in its columns. From this decomposition we have $\mathbf{G} = \mathbf{V}\mathbf{D}^{1/2} \in \mathbb{R}^{n\times n}$.

- Matrix square root: $\mathbf{Q} = \mathbf{Q}^{1/2}\mathbf{Q}^{1/2}$, where $\mathbf{Q}^{1/2}$ is the symmetric positive semidefinite "square root" matrix of $\mathbf{Q}$. From this decomposition we have $\mathbf{G} = \mathbf{Q}^{1/2} \in \mathbb{R}^{n\times n}$.

- Factor model: If $\mathbf{Q}$ is a covariance matrix of some data, then we can approximate the data series with the combination of $k \ll n$ common factors. Then we have the decomposition $\mathbf{Q} = \boldsymbol{\beta}\mathbf{Q}_F\boldsymbol{\beta}^\mathsf{T} + \mathbf{D}$, where $\mathbf{Q}_F \in \mathbb{R}^{k\times k}$ is the covariance of the factors, $\boldsymbol{\beta} \in \mathbb{R}^{n\times k}$ is the exposure of the data series to each factor, and $\mathbf{D}$ is diagonal. From this, by computing the Cholesky decomposition $\mathbf{Q}_F = \mathbf{F}\mathbf{F}^\mathsf{T}$ we have $\mathbf{G} = [\boldsymbol{\beta}\mathbf{F}, \mathbf{D}^{1/2}] \in \mathbb{R}^{n\times(n+k)}$. The advantage of factor models is that $\mathbf{G}$ is very sparse and the factors have a direct financial interpretation (see Sec. 5 for details).

After obtaining the matrix $\mathbf{G}$, we can write the quadratic form constraint as a sum-of-squares $\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{G}\mathbf{G}^\mathsf{T}\mathbf{x} \leq c$, which is a squared Euclidean norm constraint $\frac{1}{2}\|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2^2 \leq c$.

We can choose to model this using the rotated quadratic cone as

$$\left(c, 1, \mathbf{G}^\mathsf{T}\mathbf{x}\right) \in \mathcal{Q}_\mathrm{r}^{k+2}, \tag{13.15}$$

or we can choose to model its square root using the quadratic cone as

$$\left(\sqrt{c}, \mathbf{G}^\mathsf{T}\mathbf{x}\right) \in \mathcal{Q}^{k+1}. \tag{13.16}$$

Whether to use the quadratic cone or the rotated quadratic cone for modeling can be decided based on which is more natural. Typically the quadratic cone is used to model 2-norm constraints, while the rotated quadratic cone is more natural for modeling of quadratic functions. There can be exceptions, however; see for example in Sec. 2.3.

Power

Let $x \in \mathbb{R}$ and $\alpha > 1$. Then we can model the *power* constraint $c \geq |x|^\alpha$ or equivalently $c^{1/\alpha} \geq |x|$ using the power cone:

$$(c, 1, x) \in \mathcal{P}_3^{1/\alpha,(\alpha-1)/\alpha}. \tag{13.17}$$

Exponential

Let $x \in \mathbb{R}$. Then we can model the *exponential* constraint $t \geq e^x$ using the exponential cone:

$$(t, 1, x) \in K_{\exp}. \tag{13.18}$$

Log-sum-exp

Let $x_1, \ldots, x_n \in \mathbb{R}$. Then we can model the *log-sum-exp* constraint $t \geq \log\left(\sum_{i=1}^n e^{x_i}\right)$ by applying rule Sec. 13.1.1 n times:

$$\begin{array}{rcl} \sum_{i=1}^n u_i & \leq & 1, \\ (u_i, 1, x_i - t) & \in & K_{\exp},\ i = 1, \ldots, n \end{array} \tag{13.19}$$

Perspective of function

The *perspective* of a function $f(x)$ is defined as $sf(x/s)$ on $s > 0$. From any conic representation of $t \geq f(x)$ we can reach a representation of $t \geq sf(x/s)$ by substituting all constants c with their homogenized counterpart sc.

Perspective of log-sum-exp

Let $x_1, \ldots, x_n \in \mathbb{R}$. Then we can model the perspective of the log-sum-exp constraint $t \geq s\log\left(\sum_{i=1}^n e^{x_i/s}\right)$ by applying rule Sec. 13.1.1 on constraint (13.19):

$$\begin{array}{rcl} \sum_{i=1}^n u_i & \leq & s, \\ (u_i, s, x_i - t) & \in & K_{\exp},\ i = 1, \ldots, n \end{array} \tag{13.20}$$

13.1.2 Traditional quadratic models

Optimization problems involving quadratic functions often appear in practice. In this section we show how to convert the traditionally QO or QCQO problems into conic optimization form, because most of the time the solution of these conic models is computationally more efficient.

Quadratic optimization

The standard form of a quadratic optimization (QO) problem is the following:

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b}, \\ & \mathbf{x} & \geq & \mathbf{0}. \end{array} \tag{13.21}$$

The matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ must be symmetric positive semidefinite, otherwise the objective function would not be convex.

Assuming the factorization $\mathbf{Q} = \mathbf{G}\mathbf{G}^T$ with $\mathbf{G} \in \mathbb{R}^{n \times k}$, we can reformulate the problem (13.21) as a conic problem by applying the method described in Sec. 13.1.1:

$$\begin{array}{lrcl} \text{minimize} & t + \mathbf{c}^T\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b}, \\ & \mathbf{x} & \geq & \mathbf{0}, \\ & (t, 1, \mathbf{G}^T\mathbf{x}) & \in & \mathcal{Q}_{\mathrm{r}}^{k+2}. \end{array} \tag{13.22}$$

Quadratically constrained quadratic optimization

Consider the quadratically constrained quadratic optimization (QCQO) problem of the form

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{Q}_0\mathbf{x} + \mathbf{c}_0^\mathsf{T}\mathbf{x} + a_0 & \\ \text{subject to} & \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{Q}_i\mathbf{x} + \mathbf{c}_i^\mathsf{T}\mathbf{x} + a_i \;\leq\; 0, & i = 1, \ldots, m. \end{array} \tag{13.23}$$

The matrices $\mathbf{Q}_i \in \mathbb{R}^{n\times n}$, $i = 0, \ldots, m$ must all be symmetric positive semidefinite, otherwise the optimization problem would not be convex.

Assuming the factorization $\mathbf{Q}_i = \mathbf{G}_i\mathbf{G}_i^\mathsf{T}$ with $\mathbf{G}_i \in \mathbb{R}^{n\times k_i}$, we can reformulate the problem (13.23) as a conic problem by applying the method described in Sec. 13.1.1 for both the objective and the constraints:

$$\begin{array}{lrcll} \text{minimize} & t_0 + \mathbf{c}_0^\mathsf{T}\mathbf{x} + a_0 & & & \\ \text{subject to} & t_i + \mathbf{c}_i^\mathsf{T}\mathbf{x} + a_i & \leq & 0, & i = 1, \ldots, m, \\ & (t_i, 1, \mathbf{G}_i^\mathsf{T}\mathbf{x}) & \in & \mathcal{Q}_\mathrm{r}^{k_i+2}, & i = 0, \ldots, m. \end{array} \tag{13.24}$$

Practical benefits of conic models

The key step in the model conversion is to transform quadratic terms $\mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{x}$ using the factorization $\mathbf{Q} = \mathbf{G}\mathbf{G}^\mathsf{T}$, where $\mathbf{G} \in \mathbb{R}^{n\times k}$. Assuming $k \ll n$, it results in the following benefits:

- The storage requirement nk of $\mathbf{G}$ can be much lower than the storage requirement $n^2/2$ of $\mathbf{Q}$.
- The amount of work to evaluate $\mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{x}$ is proportional to n^2 whereas the work to evaluate $\|\mathbf{G}^\mathsf{T}\mathbf{x}\|_2^2$ is proportional to nk only.
- No need to numerically validate positive semidefiniteness of the matrix $\mathbf{Q}$. This is otherwise difficult owing to the presence of rounding errors that can make $\mathbf{Q}$ indefinite.
- Duality theory is much simpler for conic quadratic optimization.

In summary, the conic equivalents are not only as easy to solve as the original QP or QCQP problems, but in most cases also need less space and solution time.

13.2 Mixed-integer models

Mixed integer optimization (MIO) is an extension of convex optimization, which introduces variables taking values only in the set of integers or in some subset of integers. This allows for solving a much wider range of optimization problems in addition to the ones convex optimization already covers. However, these problems are no longer convex. Handling of integer variables make the optimization NP-hard and needs specific algorithms, thus the solution of MIO problems is much slower. In many practical cases we cannot expect to find the optimal solution in reasonable time, and only near-optimal solutions will be available.

A general mixed integer optimization problem has the following form:

$$\begin{array}{lrcll} \text{maximize} & \mathbf{c}^\mathsf{T}\mathbf{x} & & & \\ \text{subject to} & \mathbf{A}\mathbf{x}+\mathbf{b} & \in & \mathcal{K}, & \\ & x_i & \in & \mathbb{Z}, & i \in \mathcal{I}. \end{array} \tag{13.25}$$

where $\mathcal{K}$ is a cone and $\mathcal{I} \subseteq \{1, \ldots, n\}$ contains the indices of integer variables. We can model any finite range for the integer variable x_i by simply adding the extra constraint $a_i \leq x_i \leq b_i$.

13.2.1 Selection of integer constraints

In the following we will list the integer constraints appearing in financial context in this book and show how to model them.

Switch

In some practical cases we might wish to impose conditions on parts of our optimization model. For example, allowing nonzero value for a variable only in presence of a condition. We can model such situations using *binary variables* (or *indicator variables*), which can only take 0 or 1 values. The following set of constraints only allow x to be positive when the indicator variable y is equal to 1:

$$\begin{array}{rcl} x & \leq & My, \\ y & \in & \{0,1\}. \end{array} \tag{13.26}$$

The number M here is not related to the optimization problem, but it is necessary to form such a switchable constraint. If $y = 0$, then the upper limit of x is 0. If $y = 1$, then the upper limit of x is M. This modeling technique is called *big-M*. The choice of M can affect the solution performance, but a nice feature of it is that the problem cannot accidentally become infeasible.

If we have a vector variable $\mathbf{x}$ then the switch will look like:

$$\begin{array}{rcl} \mathbf{x} & \leq & \mathbf{M} \circ \mathbf{y}, \\ \mathbf{y} & \in & \{0,1\}^n, \end{array} \tag{13.27}$$

where $\circ$ denotes the elementwise product. We accounted for a possibly different big-M value for each coordinate.

Semi-continuous variable

A slight extension of Sec. 13.2.1 is to model semi-continuous variables, i. e. when $x \in \{0\} \cup [a, b]$, where $0 < a \leq b$. We can model this by

$$ay \leq x \leq by, \quad y \in \{0,1\}. \tag{13.28}$$

Cardinality

We might need to limit the number of nonzeros in a vector $\mathbf{x}$ to some number $K < n$. We can do this with the help of an indicator vector $\mathbf{y}$ of length n, which indicates $|\mathbf{x}| \neq \mathbf{0}$ (see Sec. 13.2.1). First we add a big-M bound $\mathbf{M} \circ \mathbf{y}$ to the absolute value, and model it based on Sec. 13.1.1. Then we limit the cardinality of $\mathbf{x}$ by limiting the cardinality of $\mathbf{y}$:

$$\begin{array}{rcl} \mathbf{x} & \leq & \mathbf{M} \circ \mathbf{y}, \\ \mathbf{x} & \geq & -\mathbf{M} \circ \mathbf{y}, \\ \mathbf{1}^\mathsf{T}\mathbf{y} & \leq & K, \\ \mathbf{y} & \in & \{0,1\}^n. \end{array} \tag{13.29}$$

Positive and negative part

We introduced the positive part x^+ and negative part x^- of a variable x in Sec. 13.1.1. We noted that we need a way to ensure only x^+ or x^- will be positive in the optimal solution and not both. One such way is to use binary variables:

$$\begin{array}{rcl} x & = & x^+ - x^-, \\ x^+ & \leq & My, \\ x^- & \leq & M(1-y), \\ x^+,\ x^- & \geq & 0, \\ y & \in & \{0,1\}. \end{array} \tag{13.30}$$

Here the binary variable y allows the positive (negative) part to become nonzero exactly when the negative (positive) part is zero.

Sometimes we need to handle separately the case when both the positive and the negative part is fixed to be zero. Then we need to introduce two binary variables y^+ and y^-, and include the constraint $y^+ + y^- \leq 1$ to prevent both variables from being 1:

$$\begin{array}{rcl} x & = & x^+ - x^-, \\ x^+ & \leq & My^+, \\ x^- & \leq & My^-, \\ x^+,\ x^- & \geq & 0, \\ y^+ + y^- & \leq & 1, \\ y^+,\ y^- & \in & \{0,1\}. \end{array} \tag{13.31}$$

13.3 Quadratic cones and riskless solution

In Sec. 6 we consider portfolio optimization problems with risk-free security. In this case there is a difference in the computation of the efficient frontier, if we model using the quadratic cone instead of the rotated quadratic cone. Namely, the optimization problem will not have solutions that are a mix of the risk-free security and risky securities. Instead, it will only have either the 100% risk-free solution, or a portfolio of only risky securities. Along the derivation below we will see that the same behavior applies to problems not having a risk-free security but having $\mathbf{x} = \mathbf{0}$ as a feasible solution.

Consider the following simple model:

$$\begin{array}{lrcl} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\mathbf{x} + r^\mathrm{f}x^\mathrm{f} - \tilde{\delta}\sqrt{\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{x}} & & \\ \text{subject to} & \mathbf{1}^\mathsf{T}\mathbf{x} + x^\mathrm{f} & = & 1, \\ & \mathbf{x}, x^\mathrm{f} & \geq & 0, \end{array} \tag{13.32}$$

where r^{f} is the risk-free rate and x^{f} is the allocation to the risk-free asset.

We can transform this problem using $x^{\mathrm{f}} = 1 - \mathbf{1}^{\mathsf{T}}\mathbf{x}$, such that we are only left with the variable $\mathbf{x}$:

$$\begin{array}{lrcl} \text{maximize} & (\boldsymbol{\mu} - r^{\mathrm{f}}\mathbf{1})^{\mathsf{T}}\mathbf{x} + r^{\mathrm{f}} - \tilde{\delta}\sqrt{\mathbf{x}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{x}} & & \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\mathbf{x} & \leq & 1, \\ & \mathbf{x} & \geq & 0. \end{array} \tag{13.33}$$

The feasible region in this form is a probability simplex. The solution $\mathbf{x} = \mathbf{0}$ means that everything is allocated to the risk-free security, and the hyperplane $\mathbf{1}^{\mathsf{T}}\mathbf{x} = 1$ has all the feasible solutions purely involving risky assets.

Let us denote the objective function value by $\mathrm{obj}(\mathbf{x})$. Then the directional derivative of the objective along the direction $\mathbf{u} > \mathbf{0}$, $||\mathbf{u}|| = 1$ will be

$$\partial_{\mathbf{u}}\mathrm{obj}(\mathbf{x}) = (\boldsymbol{\mu} - r^{\mathrm{f}}\mathbf{1})^{\mathsf{T}}\mathbf{u} - \tilde{\delta}\frac{\mathbf{x}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{u}}{\sqrt{\mathbf{x}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{x}}}.$$

This does not depend on the norm of $\mathbf{x}$, meaning that $\partial_{\mathbf{u}}\mathrm{obj}(c\mathbf{u})$ will be constant in c:

$$\partial_{\mathbf{u}}\mathrm{obj}(c\mathbf{u}) = (\boldsymbol{\mu} - r^{\mathrm{f}}\mathbf{1})^{\mathsf{T}}\mathbf{u} - \tilde{\delta}\sqrt{\mathbf{u}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{u}}.$$

Thus the objective is linear along the 1-dimensional slice between $\mathbf{x} = \mathbf{0}$ and any $\mathbf{x} > \mathbf{0}$, meaning that the optimal solution is either $\mathbf{x} = \mathbf{0}$ or some $\mathbf{x} > \mathbf{0}$ satisfying $\mathbf{1}^{\mathsf{T}}\mathbf{x} = 1$.

Furthermore, along every 1-dimensional slice represented by a direction $\mathbf{u}$, there is a $\tilde{\delta}$ threshold

$$\tilde{\delta}_{\mathbf{u}} = \frac{(\boldsymbol{\mu} - r^{\mathrm{f}}\mathbf{1})^{\mathsf{T}}\mathbf{u}}{\sqrt{\mathbf{u}^{\mathsf{T}}\mathbf{S}\mathbf{u}}}.$$

This is the Sharpe ratio of all portfolios of the form $c\mathbf{u}$. If $\tilde{\delta} > \tilde{\delta}_{\mathbf{u}}$ then $\partial_{\mathbf{u}}\mathrm{obj}(\mathbf{u})$ turns negative.

In general, the larger we set $\tilde{\delta}$, the fewer directions $\mathbf{u}$ will exist for which $\tilde{\delta} < \tilde{\delta}_{\mathbf{u}}$, i. e., $\partial_{\mathbf{u}}\mathrm{obj}(\mathbf{u}) > 0$, and the optimal solution is a combination of risky assets. The largest $\tilde{\delta}$ for which we still have such a direction is $\tilde{\delta}^* = \max_{\mathbf{u}} \tilde{\delta}_{\mathbf{u}}$, the maximal Sharpe ratio. The corresponding optimal portfolio is the one having the smallest risk while consisting purely of risky assets. For $\tilde{\delta} > \tilde{\delta}^*$ we have $\partial_{\mathbf{u}}\mathrm{obj}(\mathbf{u}) < 0$ in all directions, meaning that the optimal solution will always be $\mathbf{x} = \mathbf{0}$, the 100% risk-free portfolio.

13.4 Monte Carlo Optimization Selection (MCOS)

We can use the following procedure to compare the accuracy of different portfolio optimization methods.

1. Inputs of the procedure are the estimated mean return $\boldsymbol{\mu}$ and estimated covariance $\boldsymbol{\Sigma}$. Treat this pair of inputs as true values.
2. Compute the optimal asset allocation $\mathbf{x}$ for the original input pair $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
3. Repeat K_{sim} times:
    1. Do parametric bootstrap resampling, i. e., draw a new $N \times T$ return data sample $\mathbf{R}_k$ and derive a simulated pair $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

2. De-noise the covariance matrix $\mathbf{\Sigma}_k$ using the method in Sec. 4.2.2.
3. Compute the optimal portfolio $\mathbf{x}_k$ using the optimization method(s) that we wish to analyze.

4. To estimate the error for each security in the optimal portfolio, compute the standard deviation of the difference vectors $\mathbf{x}_k - \mathbf{x}$. Then by taking the mean we get a single number estimating the error of the optimization method.

Regarding the final step, we can also measure the average decay in performance by any of the following:

- the mean difference in expected outcomes: $\frac{1}{K_{\text{sim}}} \sum_k (\mathbf{x}_k - \mathbf{x})^\mathsf{T} \boldsymbol{\mu}$
- the mean difference in variance: $\frac{1}{K_{\text{sim}}} \sum_k (\mathbf{x}_k - \mathbf{x})^\mathsf{T} \mathbf{\Sigma} (\mathbf{x}_k - \mathbf{w})$
- the mean difference in Sharpe ratio or other metric computed from the above statistics.

Chapter 14

Notation and definitions

Here we list some of the notations used in this book.

14.1 Financial notations

- N: Number of securities in the security universe considered.
- $p_{0,i}$: The known price of security i at the beginning of the investment period.
- $\mathbf{p}_0$: The known vector of security prices at the beginning of the investment period.
- $P_{h,i}$: The random price of security i at the end of the investment period.
- P_h: The random vector of security prices at the end of the investment period.
- R_i: The random rate of return of security i.
- R: The random vector of security returns.
- $\mathbf{r}$: The known vector of security returns.
- $\mathbf{R}$: The sample return data matrix consisting of security return samples as columns.
- μ_i: The expected rate of return of security i.
- μ: The vector of expected security returns.
- Σ: The covariance matrix of security returns.
- x_i: The fraction of funds invested into security i.
- $\mathbf{x}$: Portfolio vector.
- $\mathbf{x}_0$: Initial portfolio vector at the beginning of the investment period.
- $\tilde{\mathbf{x}}$: The change in the portfolio vector compared to $\mathbf{x}_0$.
- x^{f}: The fraction of funds invested into the risk-free security.
- x_0^{f}: The fraction of initial funds invested into the risk-free security.
- $\tilde{x}^{\mathrm{f}}$: The change in the risk-free investment compared to x_0^{f}.

- $R_{\mathbf{x}}$: Portfolio return computed from portfolio $\mathbf{x}$.
- $\mathbf{R}_{\mathbf{x}}$: Sample portfolio return computed from data matrix $\mathbf{R}$ and portfolio $\mathbf{x}$.
- $\mu_{\mathbf{x}}$: Expected portfolio return computed from portfolio $\mathbf{x}$.
- $\sigma_{\mathbf{x}}$: Expected portfolio variance computed from portfolio $\mathbf{x}$.
- $\boldsymbol{\mu}$: Estimate of expected security return vector μ.
- $\boldsymbol{\Sigma}$: Estimate of security return covariance matrix Σ.
- T: Number of data samples or scenarios.
- h: Time period of investment.
- τ: Time period of estimation.

14.2 Mathematical notations

- $\mathbf{x}^\mathsf{T}$: Transpose of vector $\mathbf{x}$. Vectors are all column vectors, so their transpose is always a row vector.
- $\mathbb{E}(R)$: Expected value of R.
- $\mathrm{Var}(R)$: Variance of R.
- $\mathrm{Cov}(R_i, R_j)$: Covariance of R_i and R_j.
- $\mathbf{1}$: Vector of ones.
- $\mathcal{F}$: Feasible region generated by a set of constraints.
- $\mathbb{R}^n$: Set of n-dimensional real vectors.
- $\mathbb{Z}^n$: Set of n-dimensional integer vectors.
- $\langle \mathbf{a}, \mathbf{b} \rangle$: Inner product of vectors $\mathbf{a}$ and $\mathbf{b}$. Sometimes used instead of notation $\mathbf{a}^\mathsf{T}\mathbf{b}$.
- $\mathrm{diag}(\mathbf{S})$: Vector formed by taking the main diagonal of matrix $\mathbf{S}$.
- $\mathrm{Diag}(\mathbf{x})$: Diagonal matrix with vector $\mathbf{x}$ in the main diagonal.
- $\mathrm{Diag}(\mathbf{S})$: Diagonal matrix formed by taking the main diagonal of matrix $\mathbf{S}$.
- $\mathcal{F}$: Part of the feasible set of an optimization problem. Indicates that the problem can be extended with further constraints.
- e: Euler's number.

14.3 Abbreviations

- MVO: Mean–variance optimization
- LO: Linear optimization
- QO: Quadratic optimization
- QCQO: Quadratically constrained quadratic optimization
- SOCO: Second-order cone optimization
- SDO: Semidefinite optimization

Bibliography

[AJ12] A. Ahmadi-Javid. Entropic value-at-risk: a new coherent risk measure. *Journal of Optimization Theory and Applications*, 12 2012. doi:10.1007/s10957-011-9968-2.

[AZ10] D. Avramov and G. Zhou. Bayesian portfolio analysis. *Annual Review of Financial Economics*, 2(1):25–47, 2010.

[BN01] J. Bai and S. Ng. Determining the number of factors in approximate factor models. *Econometrica*, 01 2001. doi:10.1111/1468-0262.00273.

[BS11] J. Bai and S. Shi. Estimating high dimensional covariance matrices and its applications. *Annals of Economics and Finance*, 12:199–215, 11 2011.

[BGP12] D. Bertsimas, V. Gupta, and I. Paschalidis. Inverse optimization: a new perspective on the black-litterman model. *Operations Research*, 60:1389–1403, 12 2012. doi:10.2307/23323707.

[BBD+17] S. Boyd, E. Busseti, S. Diamond, R. N. Kahn, K. Koh, P. Nystrup, and J. Speth. Multi-period trading via convex optimization. 2017. arXiv:1705.00109.

[Bra10] M. W. Brandt. Chapter 5 - portfolio choice problems. In *Handbook of Financial Econometrics: Tools and Techniques*, volume 1 of Handbooks in Finance, pages 269–336. North-Holland, San Diego, 2010.

[CY16] J. Chen and M. Yuan. Efficient portfolio selection in a large market. *Journal of Financial Econometrics*, 14:496–524, 06 2016. doi:10.1093/jjfinec/nbw003.

[Con95] G. Connor. The three types of factor models: a comparison of their explanatory power. *Financial Analysts Journal*, 51:42–46, 05 1995. doi:10.2469/faj.v51.n3.1904.

[CM13] G. Coqueret and V. Milhau. Estimating covariance matrices for portfolio optimization. 12 2013. EDHEC Risk Institute.

[CJPT18] G. Cornuéjols, J. Peña, and R. Tütüncü. *Optimization Methods in Finance*. Cambridge University Press, 2 edition, 2018. doi:10.1017/9781107297340.

[CK20] Giorgio Costa and Roy H. Kwon. Generalized risk parity portfolio optimization: an admm approach. *J. of Global Optimization*, 78(1):207–238, sep 2020.

[dP19] M. López de Prado. A robust estimator of the efficient frontier. *SSRN Electronic Journal*, 01 2019. doi:10.2139/ssrn.3469961.

[dPL19] M. López de Prado and M. Lewis. Detection of false investment strategies using unsupervised learning methods. *Quantitative Finance*, 19:1–11, 07 2019. doi:10.1080/14697688.2019.1622311.

[DGNU09] V. Demiguel, L. Garlappi, F. Nogales, and R. Uppal. A generalized approach to portfolio optimization: improving performance by constraining portfolio norms. *Management Science*, 55:798–812, 05 2009. doi:10.2139/ssrn.1004707.

[Dod12] J. Dodson. Why is it so hard to estimate expected returns? 2012. University of Minnesota, School of Mathematics. URL: https://www-users.math.umn.edu/~dodso013/docs/dodson2012-lambda.pdf.

[EM75] B. Efron and C. Morris. Data analysis using stein's estimator and its generalizations. *Journal of The American Statistical Association*, 70:311–319, 06 1975. doi:10.1080/01621459.1975.10479864.

[EM76] B. Efron and C. Morris. Families of minimax estimators of the mean of a multivariate normal distribution. *The Annals of Statistics*, 01 1976. doi:10.1214/aos/1176343344.

[FS02] Hans Foellmer and Alexander Schied. Convex measures of risk and trading constraints. *Finance and Stochastics*, 6:429–447, 09 2002. doi:10.1007/s007800200072.

[GI03] Donald Goldfarb and Garud Iyengar. Robust portfolio selection problems. *Math. Oper. Res.*, 28:1–38, 02 2003. doi:10.1287/moor.28.1.1.14260.

[GK00] Richard C. Grinold and Ronald N. Kahn. *Active portfolio management*. McGraw-Hill, New York, 2 edition, 2000.

[Har91] W.V. Harlow. Asset allocation in a downside-risk framework. *Financial Analysts Journal*, 47(5):28–40, 1991. doi:10.2469/faj.v47.n5.28.

[JM03] R. Jagannathan and T. Ma. Risk reduction in large portfolios: why imposing the wrong constraint helps. *Journal of Finance*, 58:1651–1684, 08 2003. doi:10.1111/1540-6261.00580.

[Jor86] P. Jorion. Bayes-stein estimation for portfolio analysis. *Journal of Financial and Quantitative Analysis*, 21:279–292, 09 1986. doi:10.2307/2331042.

[Jor04] P. Jorion. Portfolio optimization with tracking-error constraints. *Financial Analysts Journal*, 01 2004. doi:10.2469/faj.v59.n5.2565.

[Kak16] Z. Kakushadze. Shrinkage = factor model. *Journal of Asset Management*, 17:, 03 2016. doi:10.1057/jam.2015.40.

[KS13] R. Karels and M. Sun. Active portfolio construction when risk and alpha factors are misaligned. In C. S. Wehn, C. Hoppe, and G. N. Gregoriou, editors, *Rethinking Valuation and Pricing Models*, pages 399–410. Academic Press, 12 2013. doi:10.1016/B978-0-12-415875-7.00024-5.

[KY91] Hiroshi Konno and Hiroaki Yamazaki. Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market. *Management Science*, 37(5):519–531, 1991.

[Lau01] G. J. Lauprête. *Portfolio risk minimization under departures from normality*. Ph.D. Thesis, Massachusetts Institute of Technology, Sloan School of Management, Operations Research Center., 2001. URL: http://dspace.mit.edu/handle/1721.1/7582.

[LW03a] O. Ledoit and M. Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 07 2003. doi:10.2139/ssrn.433840.

[LW03b] O. Ledoit and M. Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10(5):603–621, 2003. doi:https://doi.org/10.1016/S0927-5398(03)00007-0.

[LW04] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *J. Multivariate Anal.*, 88(2):365–411, 2004. doi:https://doi.org/10.1016/S0047-259X(03)00096-4.

[LW20] O. Ledoit and M. Wolf. Analytical nonlinear shrinkage of large-dimensional covariance matrices. *The Annals of Statistics*, 48(5):3043–3065, 2020. doi:10.1214/19-AOS1921.

[LC98] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer, New York, NY, 2 edition, 1998. doi:https://doi.org/10.1007/b98854.

[LMFB07] M. S. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 152:341–365, 03 2007. doi:10.1007/s10479-006-0145-1.

[LB22] E. Luxenberg and S. Boyd. Portfolio construction with gaussian mixture returns and exponential utility via convex optimization. 2022. URL: https://arxiv.org/abs/2205.04563, doi:10.48550/ARXIV.2205.04563.

[MWOS15] R. Mansini, W. Ogryczak, and M. G. Speranza. *Linear and Mixed Integer Programming for Portfolio Optimization*. Springer International Publishing, 1 edition, 2015. doi:10.1007/978-3-319-18482-1.

[Meu05] A. Meucci. *Risk and Asset Allocation*. Springer-Verlag Berlin Heidelberg, 1 edition, 2005. doi:10.1007/978-3-540-27904-4.

[Meu10a] A. Meucci. Quant nugget 2: linear vs. compounded returns – common pitfalls in portfolio management. *GARP Risk Professional*, pages 49–51, 04 2010. URL: https://ssrn.com/abstract=1586656.

[Meu10b] A. Meucci. Quant nugget 4: annualization and general projection of skewness, kurtosis and all summary statistics. *GARP Risk Professional - "The Quant Classroom"*, pages 59–63, 08 2010. URL: https://ssrn.com/abstract=1635484.

[Meu10c] A. Meucci. Quant nugget 5: return calculations for leveraged securities and portfolios. *GARP Risk Professional*, pages 40–43, 10 2010. URL: https://ssrn.com/abstract=1675067.

[Meu11] A. Meucci. 'the prayer' ten-step checklist for advanced risk and portfolio management. *SSRN*, 02 2011. URL: https://ssrn.com/abstract=1753788.

[MM08] Richard Michaud and Robert Michaud. *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation 2nd Edition*. Oxford University Press, 2 edition, 01 2008.

[Pal20] D. P. Palomar. Risk parity portfolio. 2020. Lecture notes, The Hong Kong University of Science and Technology.

[Ric99] J. A. Richards. An introduction to james–stein estimation. 11 1999. M.I.T. EECS Area Exam Report.

[RU00] R. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 01 2000.

[RU02] R. T. Rockafellar and S. P. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002. doi:https://doi.org/10.1016/S0378-4266(02)00271-6.

[RUZ06] R. T. Rockafellar, S. P. Uryasev, and M. Zabarankin. Generalized deviations in risk analysis. *Finance and Stochastics*, 10:51–74, 2006.

[RSTF20] D. Rosadi, E. Setiawan, M. Templ, and P. Filzmoser. Robust covariance estimators for mean-variance portfolio optimization with transaction lots. *Operations Research Perspectives*, 7:100154, 06 2020. doi:10.1016/j.orp.2020.100154.

[RWZ99] M. Rudolf, H.-J. Wolter, and H. Zimmermann. A linear model for tracking error minimization. *Journal of Banking & Finance*, 23(1):85–103, 1999. doi:https://doi.org/10.1016/S0378-4266(98)00076-4.

[SHAD13] Thomas Schmelzer, Raphael Hauser, Erling Andersen, and Joachim Dahl. Regression techniques for portfolio optimisation using mosek. 2013. URL: https://arxiv.org/abs/1310.3397.

[The71] H. Theil. *Principles of Econometrics*. John Wiley and Sons, 1 edition, 06 1971.

[TLD+11] B. Tóth, Y. Lemperiere, C. Deremble, J. Lataillade, J. Kockelkoren, and J.-P. Bouchaud. Anomalous price impact and the critical nature of liquidity in financial markets. *Physical Review X*, 05 2011. doi:10.2139/ssrn.1836508.

[VD09] F. J. Nogales V. DeMiguel. Portfolio selection with robust estimation. *Operations Research*, 57(3):560–577, 02 2009. doi:https://doi.org/10.1287/opre.1080.0566.

[WZ07] R. Welsch and X. Zhou. Application of robust statistics to asset allocation models. *REVSTAT*, 5:97–114, 03 2007.

[WO11] M. Woodside-Oriakhi. *Portfolio Optimisation with Transaction Cost*. PhD thesis, School of Information Systems, Computing and Mathematics, Brunel University, 01 2011.

[MOSEKApS24] MOSEK ApS. *MOSEK Modeling Cookbook*. MOSEK ApS, Fruebjergvej 3, Boks 16, 2100 Copenhagen O, 2024. Last revised April 2024. URL: https://docs.mosek.com/modeling-cookbook/index.html.