Quantitative Portfolio Management with Python

Dr. Cyril Bachelard

Spring Semester, February 26th 2025

Lecture 2

Portfolio

Optimization

Lecture 2

Universität Zürich[UZH] | ETH zürich

Portfolio (Economic Perspective)

- Economics
- Modern Portfolio Theory (MPT)
- Markowitz (1952)

Optimization (Technical Perspective)

- Mathematics
- Operations Research (OR)
- Programmatic Implementation

Part I

Optimization

Optimization

Note: The following explanations do not offer a comprehensive treatment of the topic but are intended to present concepts that are practically relevant.

For an in-depth exploration of (convex) optimization, refer to:

- S. Boyd and L. Vandenberghe (2004), *Convex optimization*.
  In particular, Chapter 5.

- L. Younes (2024), Introduction to Machine Learning
  Chapter 3.

Optimization

Informally:
An optimization problem involves a decision-making scenario where we aim to identify the option that minimizes a cost (or maximizes a reward) function, given a set of feasible choices.

The three main components of an optimization problem are:

- **Decision Variable Space**: set where our decision variable, x, lives. For us, $x \in \mathbb{R}^n$.
- **Objective Function** (cost function, reward function) $f : \mathbb{R}^n \to \mathbb{R}$. Ideally, f is smooth and convex.
- **Constraints**.
  - Equality Constraints: $a_i(x) - b_i = 0, \quad i = 1, \ldots, m$
  - Inequality Constraints: $g_j(x) - h_j \leq 0, \quad j = 1, \ldots, p$

Primal Problem Formulation

Consider the general form of a **primal optimization problem**:

$$\min_x f(x)$$

subject to:

$$a_i(x) - b_i = 0, \quad i = 1, \ldots, m$$

$$g_j(x) - h_j \leq 0, \quad j = 1, \ldots, p$$

where

- $f(x)$ is the objective function to minimize,
- $a_i(x)$ are equality constraints,
- $g_j(x)$ are inequality constraints,
- x is the vector of decision variables.

Optimality Conditions

- If we were minimizing a convex function $f(x)$ without constraints, the optimum, x^*, is a critical point which we find by setting the gradient of the objective function to zero and solving for x. I.e., $\nabla f(x) \stackrel{!}{=} 0$.

- With additional equality constraints, we move the constraint functions to the objective function, weighted by scalars λ_i, called **Lagrange multipliers** or **dual values**. This forms the Lagrangian function

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i a_i(x).$$

To find the optimal solution, set the gradient of the Lagrangian to zero: $\nabla \mathcal{L}(x, \lambda) \stackrel{!}{=} 0$.

Optimality Conditions

- With further inequality constraints we extend the Lagrangian to

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i a_i(x) + \sum_{j=1}^{p} \nu_j g_j(x),$$

and set $\nabla \mathcal{L}(x, \lambda, \nu) \stackrel{!}{=} 0$, and further ensure that the KKT conditions are satisfied.

Karush-Kuhn-Tucker (KKT) Conditions

$$
\begin{aligned}
&I. \quad a_i(x^*) = 0, && i = 1, ..., m \\
&II. \quad g_j(x^*) \leq 0, && j = 1, ..., p \\
&III. \quad \nabla f(x^*) + \sum_i \lambda_i \nabla a_i(x^*) + \sum_j \nu_i \nabla g_i(x^*) = 0 \\
&IV. \quad \nu_j \geq 0, && j = 1, ..., p \\
&V. \quad \nu_j g_j(x^*) = 0 && j = 1, ..., p
\end{aligned}
$$

Karush-Kuhn-Tucker (KKT) Conditions

- I.,II. **Primal feasibility:** The solution must satisfy the constraints of the optimization problem.
- III. **Stationarity:** The gradient of the Lagrangian function with respect to the decision variables must vanish at the optimal point.
- IV. **Dual feasibility:** The Lagrange multipliers associated with the inequality constraints have to be non-negative.
- V. **Complementary slackness:** Each Lagrange multiplier must correspond to an **active** constraint, meaning that if a constraint is inactive (not binding), the associated multiplier must be zero.

Formulating the Dual Problem

The **dual function** is obtained by minimizing the Lagrangian over x for given multipliers:

$$q(\lambda, \nu) = \inf_x \mathcal{L}(x, \lambda, \nu).$$

This dual function provides a lower bound on the primal objective value. It can be interpreted as a function of the dual variables λ and ν, representing the best achievable objective for the given set of multipliers.

The **Lagrange dual problem** is:

$$\max_{\lambda, \nu > 0} q(\lambda, \nu).$$

Primal-Dual Structure

- We can solve the problem using a max-min approach:

$$\max_{\lambda,\nu \geq 0} \min_{x} \mathcal{L}(x, \lambda, \nu).$$

- Interpretation:
  - Inner minimization: Find the best x given penalty terms λ, ν.
  - Outer maximization: Adjust λ, ν to enforce feasibility.
    - ★ Feasibility: If $g_j(x) > 0$, increasing ν_j raises the penalty.
    - ★ Optimality: Correct ν_j^* enforces the constraint.
  - This leads to a saddle-point solution where the optimization reaches an equilibrium satisfying the constraints optimally.

In Summary

- Constraints can be reformulated to **penalty terms** in the objective.
- The Lagrange multipliers can be interpreted as **shadow prices** — they indicate how much the objective function would improve if the constraint were relaxed.
- Duality: Ability to view a mathematical concept from two perspectives: the primal and the dual.
- The dual problem provides a lower bound on the primal problem's objective.
- In convex optimization, if certain conditions (like Slater's condition[1] for inequality constraints) are satisfied, strong duality means that the dual formulation can provide an exact solution to the primal problem.

[1] See e.g., https://en.wikipedia.org/wiki/Slaters_condition

Algorithms, e.g., Interior-Point Methods

Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

Faster Solutions Seen

These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances, or whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use them in creating portfolios with the best mix of stocks and bonds.

The Bell Labs mathematician, Dr. Narendra Karmarkar, has devised a radically new procedure that may speed the routine handling of such problems by businesses and Government agencies and also make it possible to tackle problems that are now far out of reach.

"This is a path-breaking result," said Dr. Ronald L. Graham, director of mathematical sciences for Bell Labs in Murray Hill, N.J.

"Science has its moments of great progress, and this may well be one of them."

Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

A procedure devised in 1947, the simplex method, is now used for such problems,

Continued on Page A19, Column 1

THE NEW YORK TIMES, November 19, 1984

Numerical Tolerances

- Numerical tolerances determine when to stop iterating in the search of a solution.
- Common tolerances include:
  - **Primal residuals**: The violation of the primal feasibility condition.
  - **Dual residuals**: The violation of the dual feasibility condition.
  - **Duality gap**: The difference between the primal and dual objective values.
- Solvers stop when all residuals are below a predefined tolerance.
- The choice of tolerance is important: too loose a tolerance may lead to suboptimal solutions, too tight may lead to unnecessary iterations.

Quadratic Optimization

- Quadratic Programming (QP) solvers are used to solve problems of the form:

$$x^* = \underset{x \in \mathcal{P}}{\text{argmin}} \quad \frac{1}{2} x^\top \Sigma x + \mu^\top x$$
$$\text{with} \quad \mathcal{P} := \{x \in \mathbb{R}^n | Ax = b, Gx \leq h, l \leq x \leq u\}.$$

- A has dimension $m \times n$ and G has dimension $p \times n$.
- We assume that $\Sigma \succ 0$ (i.e., positive definite) such that our problem is convex.

Example:
Analytical Minimum Variance Portfolio

Consider the following minimum-variance portfolio optimization problem:

$$\min_{x \in \mathbb{R}^n} x^\top \Sigma x$$
$$s.t. \quad Ax = b$$

where Σ has dimension $n \times n$, and A has dimension $m \times n$.

Define the Lagrangian function:

$$\mathcal{L}(x, \lambda) = x^\top \Sigma x + \lambda(Ax - b)$$

where λ is the $m \times 1$ vector of Lagrange multipliers.

Applying first order conditions with respect x,

$$\frac{\delta \mathcal{L}(x, \lambda)}{x} = 2\Sigma x + A^\top \lambda = 0.$$

Example:
Analytical Minimum Variance Portfolio

Solving for x:

$$x = -\frac{1}{2}\Sigma^{-1}A^{\top}\lambda.$$

Substituting into the constraint,

$$A(-\frac{1}{2}\Sigma^{-1}A^{\top}\lambda) = b$$

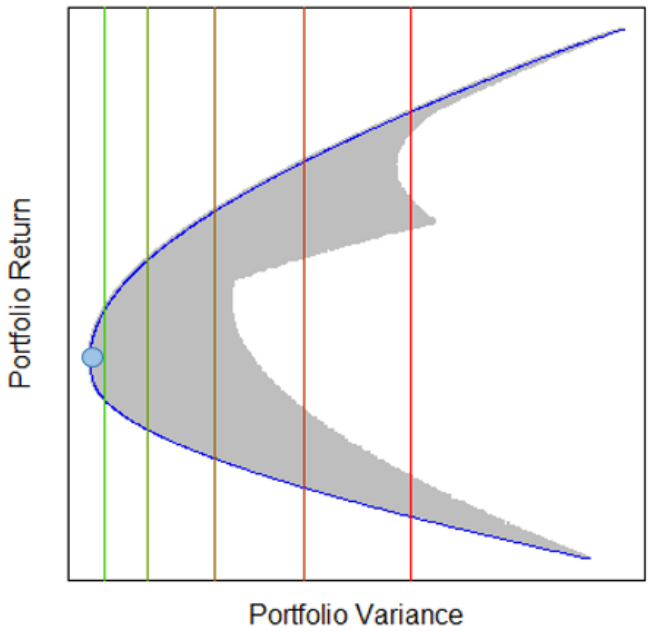
Solving for λ,

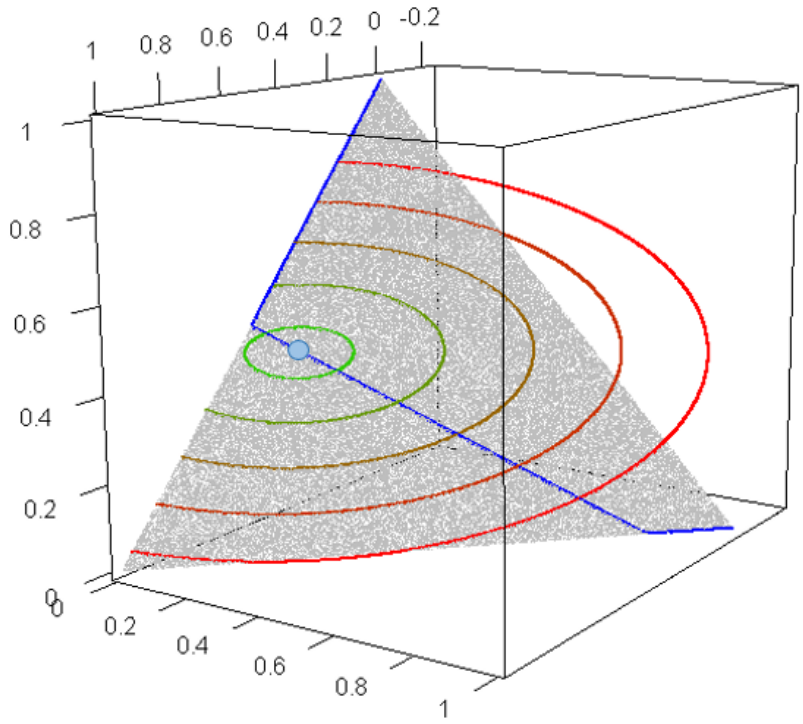
$$\lambda = -2(A\Sigma^{-1}A^{\top})^{-1}b.$$

Substituting λ back into the expression for x,

$$x^* = \Sigma^{-1}A^{\top}(A\Sigma^{-1}A^{\top})^{-1}b.$$

Concepts from Geometry





Blue line:

- Left: Efficient Frontier
- Right: Critical Line[2]

[2] Markowitz, H. (1952). *Portfolio Selection*. Journal of Finance, 7(1), 77-91.

Concepts from Geometry

Definition (Convex Set)

A set $C \subseteq \mathbb{R}^n$ *is convex if for all* $x, y \in C$ *and for all* $\alpha \in [0,1]$, $\alpha x + (1-\alpha)\, y \in C$.

Definition (Convex Function)

A function $f : \mathbb{R}^n \to \mathbb{R}^n$ *is convex if for all* $x, y \in C$ *and for all* $\alpha \in [0,1]$,

$$f\left(\alpha x + (1-\alpha)\, y\right) \leq \alpha f\left(x\right) + (1-\alpha)\, f\left(y\right).$$

Definition (Hyperplane)

A hyperplane $\mathcal{H}(v, \gamma) \in \mathbb{R}^n$ *is the set*

$$\mathcal{H}(v, \gamma) = \{x \in \mathbb{R}^n \mid v^\top x = \gamma\}$$

with $v \in \mathbb{R}^n$ *and* $\gamma \in \mathbb{R}^n$ *fixed.* v *is called the normal.*

Concepts from Geometry

Definition (Polyhedron)

A n-dimensional (convex) polyhedron is the intersection of p n-dimensional half-spaces $H_j = \left\{ x \in \mathbb{R}^n \,\middle|\, g_j^\top x \leq h_j \right\}$ *for* $j = 1, \ldots, p$

$$P = \bigcap_{j=1}^{p} H_j = \{ x \in \mathbb{R}^n \,|\, Gx \leq h \}$$

where the j-th row of G is g_j*. This is the H-representation of a polyhedron.*

Definition (Polytope)

A (convex) polytope $\mathcal{P}(G, h)$ *is a bounded (convex) polyhedron.*

Concepts from Geometry

Definition (Ellipsoid)

A n-dimensional ellipsoid $\mathcal{E}(c, \Sigma) \in \mathbb{R}^n$ *with centre vector* c *and shape matrix* Σ *is the set*

$$\mathcal{E}(c, \Sigma) = \{x \in \mathbb{R}^n \mid (x - c)^\top \Sigma (x - c) \leq 1\}$$

wherein matrix $\Sigma \succeq 0$, *i.e., is positive semi-definite.*

Concepts from Geometry

Definition (Standard Simplex)

A standard (n-1)-simplex is

$$\mathcal{S} = \left\{ x \in \mathbb{R}^n | \mathbf{1}^T x = 1, x \geq 0 \right\}$$

We say (n-1)-simplex to highlight that we are talking about a (n-1) dimensional object embedded in n-dimensional space.

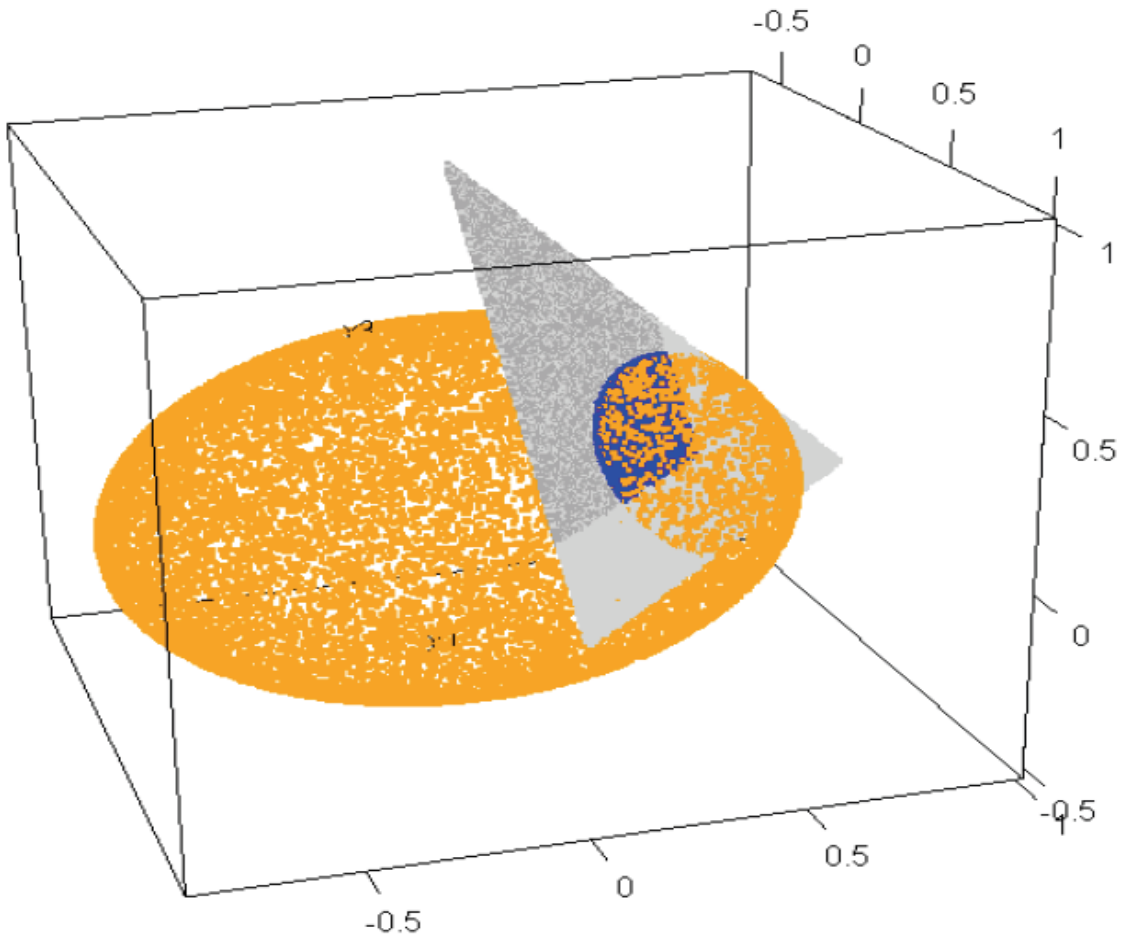
Definition (Constrained Simplex)

A constrained (n-1)-simplex is defined as the intersection of a standard (n-1)-simplex $\mathcal{S}$ and a given set $C \subseteq \mathbb{R}^n$, i.e.,

$$\mathcal{S}^c = \mathcal{S} \cap C = \left\{ x \in \mathbb{R}^n | \mathbf{1}^T x = 1, x \geq 0, x \in C \right\}$$

Concepts from Geometry

Example of convex body defined by the intersection: $\mathcal{S} \cap P \cap \mathcal{E}$ (i.e., a linearly and quadratically constrained simplex).



Part II

Coding

Modules and Classes in qpmwp-course

- Estimation
  - ExpectedReturn
  - Covariance
- Optimization
  - Constraints
  - OptimizationData
  - QuadraticProgram
  - MeanVariance

qpsolvers

- The goal of the qpsolvers library is to simplify the process of solving quadratic programming problems by providing a **single, unified interface** that can be used to access different solvers without needing to directly interact with the underlying solver details and API.
- Installation (from PyPI)
  - `pip install qpsolvers` will only install the library itself.
  - `pip install qpsolvers[wheels_only]` will only install solvers with pre-compiled binaries.
  - `pip install qpsolvers[cvxopt,daqp,highs,osqp,qpalm,quadprog]` (for instance) will install the listed set of QP solvers.
- To inspect which solvers you have installed:
  - `import qpsolvers`
  - `qpsolvers.available_solvers`
- Available solvers

Pull the latest changes from github

- `Sync fork` button in your github repo, then
  `git pull`

- Or:
  - Add the upstream remote: This points to the original repository from which you forked.
    `git remote add upstream`
    `https://github.com/cbachela/qpmwp-course.git`
  - Fetch the upstream changes: This gets the latest changes from the upstream repository.
    `git fetch upstream`
  - Merge the changes into your local branch: Usually, you would merge the changes into your main branch. `git checkout main`
    `git merge upstream/main`
  - Push the merged changes to your fork: Ensure your fork's remote is up-to-date with the upstream changes.
    `git push origin main`

Assignment 1

- See: https://github.com/cbachela/qpmwp-course/blob/main/assignments/assignment_1.ipynb
- Deadline: 19.03.2025, 12:00 CET