

# Course 2023-2024 in Portfolio Allocation and Asset Management

## Lecture 7. Machine Learning in Asset Management

Thierry Roncalli\*

\*Amundi Asset Management<sup>1</sup>

\*University of Paris-Saclay

January 2024

---

<sup>1</sup>The opinions expressed in this presentation are those of the authors and are not meant to represent the opinions or official positions of Amundi Asset Management.

# General information

## 1 Overview

The objective of this course is to understand the theoretical and practical aspects of asset management

## 2 Prerequisites

M1 Finance or equivalent

## 3 ECTS

3

## 4 Keywords

Finance, Asset Management, Optimization, Statistics

## 5 Hours

Lectures: 24h, HomeWork: 30h

## 6 Evaluation

Project + oral examination

## 7 Course website

[www.thierry-roncalli.com/AssetManagementCourse.html](http://www.thierry-roncalli.com/AssetManagementCourse.html)

# Objective of the course

The objective of the course is twofold:

- ① having a financial culture on asset management
- ② being proficient in quantitative portfolio management

# Class schedule

## Course sessions

- January 12 (6 hours, AM+PM)
- January 19 (6 hours, AM+PM)
- January 26 (6 hours, AM+PM)
- February 2 (6 hours, AM+PM)

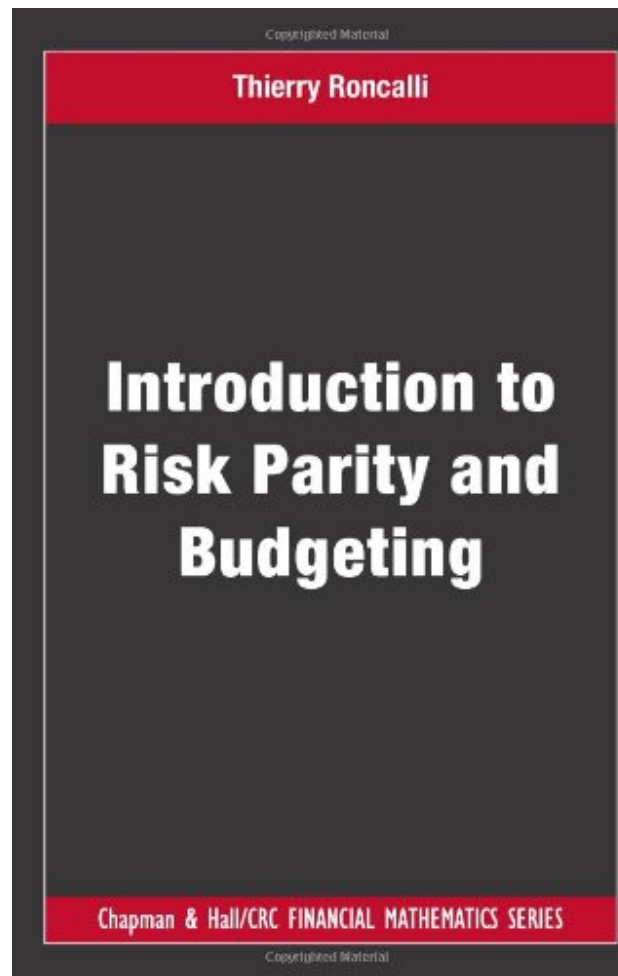
Class times: Fridays 9:00am-12:00pm, 1:00pm-4:00pm, University of Evry

# Agenda

- Lecture 1: Portfolio Optimization
- Lecture 2: Risk Budgeting
- Lecture 3: Smart Beta, Factor Investing and Alternative Risk Premia
- Lecture 4: Equity Portfolio Optimization with ESG Scores
- Lecture 5: Climate Portfolio Construction
- Lecture 6: Equity and Bond Portfolio Optimization with Green Preferences
- Lecture 7: Machine Learning in Asset Management

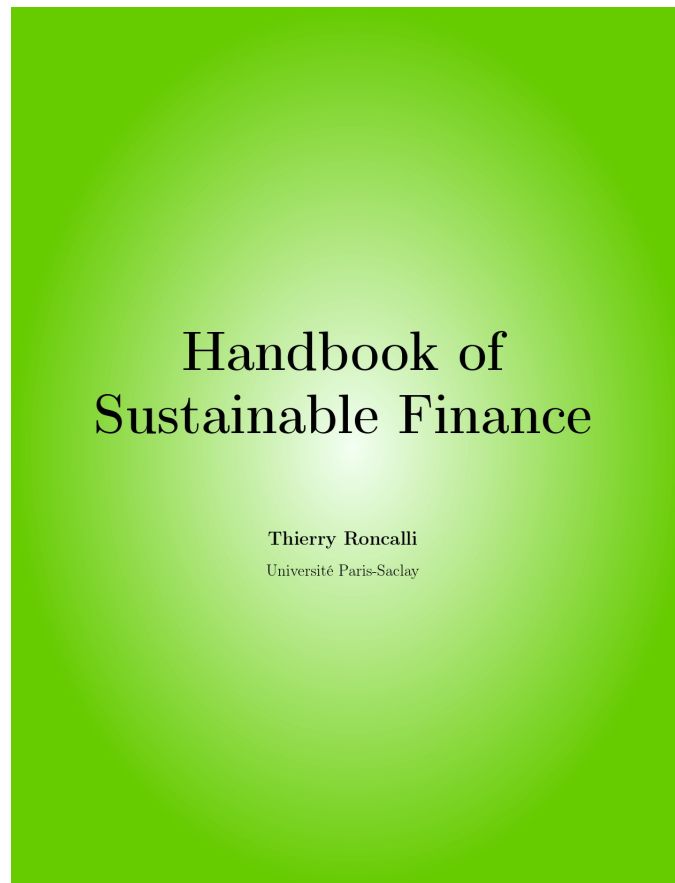
# Textbook (Asset Management)

- Roncalli, T. (2013), *Introduction to Risk Parity and Budgeting*, Chapman & Hall/CRC Financial Mathematics Series.



# Textbook (Sustainable Finance)

- Roncalli, T. (2024), *Handbook of Sustainable Finance*.



# Additional materials

- Slides, tutorial exercises and past exams can be downloaded at the following address:

`www.thierry-roncalli.com/AssetManagementCourse.html`

- Solutions of exercises can be found in the companion book, which can be downloaded at the following address:

`http://www.thierry-roncalli.com/RiskParityBook.html`



# Agenda

- Lecture 1: Portfolio Optimization
- Lecture 2: Risk Budgeting
- Lecture 3: Smart Beta, Factor Investing and Alternative Risk Premia
- Lecture 4: Equity Portfolio Optimization with ESG Scores
- Lecture 5: Climate Portfolio Construction
- Lecture 6: Equity and Bond Portfolio Optimization with Green Preferences
- **Lecture 7: Machine Learning in Asset Management**

# Prologue

- Machine learning is a hot topic in asset management (and more generally in finance)
- Machine learning and data mining are two sides of the same coin

**backtesting performance**  $\neq$  **live performance**

- Reaching for the stars: a complex/complicated process does not mean a good solution

Don't forget the 3 rules in asset management

- 1 It is difficult to make money
- 2 It is difficult to make money
- 3 It is difficult to make money

# Prologue

- In this lecture, we focus on ML optimization algorithms, because they have proved their worth
- We have no time to study classical ML methods that can be used by quants to build investment strategies<sup>2</sup>

---

<sup>2</sup>Don't believe that they are always significantly better than standard statistical approaches!!!

# Standard optimization algorithms

- Gradient descent methods
- Conjugate gradient (CG) methods (Fletcher–Reeves, Polak–Ribiere, etc.)
- Quasi-Newton (QN) methods (NR, BFGS, DFP, etc.)
- Quadratic programming (QP) methods
- Sequential QP methods
- Interior-point methods

# Standard optimization algorithms

- We consider the following unconstrained minimization problem:

$$x^* = \arg \min_x f(x) \quad (1)$$

where  $x \in \mathbb{R}^n$  and  $f(x)$  is a continuous, smooth and convex function

- In order to find the solution  $x^*$ , optimization algorithms use iterative algorithms:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \Delta x^{(k)} \\ &= x^{(k)} - \eta^{(k)} D^{(k)} \end{aligned}$$

where:

- $x^{(0)}$  is the vector of starting values
- $x^{(k)}$  is the approximated solution of Problem (1) at the  $k^{\text{th}}$  iteration
- $\eta^{(k)} > 0$  is a scalar that determines the step size
- $D^{(k)}$  is the direction

# Standard optimization algorithms

- Gradient descent:

$$D^{(k)} = \nabla f \left( x^{(k)} \right) = \frac{\partial f \left( x^{(k)} \right)}{\partial x}$$

- Newton-Raphson method:

$$D^{(k)} = \left( \nabla^2 f \left( x^{(k)} \right) \right)^{-1} \nabla f \left( x^{(k)} \right) = \left( \frac{\partial^2 f \left( x^{(k)} \right)}{\partial x \partial x^\top} \right)^{-1} \frac{\partial f \left( x^{(k)} \right)}{\partial x}$$

- Quasi-Newton method:

$$D^{(k)} = H^{(k)} \nabla f \left( x^{(k)} \right)$$

where  $H^{(k)}$  is an approximation of the inverse of the Hessian matrix

# Standard optimization algorithms

What are the issues?

- 1 How to solve large-scale optimization problems?
- 2 How to solve optimization problems where there are multiple solutions?
- 3 How to just find an “*acceptable*” solution?

## The case of neural networks and deep learning

⇒ Standard approaches are not well adapted

# Machine learning optimization algorithms

## Machine learning problems

- Non-smooth objective function
- Non-unique solution
- Large-scale dimension

**Optimization in machine learning requires  
to reinvent numerical optimization**



# Machine learning optimization algorithms

We consider 4 methods:

- Cyclical coordinate descent (CCD)
- Alternative direction method of multipliers (ADMM)
- Proximal operators (PO)
- Dykstra's algorithm (DA)

# Coordinate descent methods

## The fall and the rise of the steepest descent method

In the 1980s:

- Conjugate gradient methods (Fletcher–Reeves, Polak–Ribiere, etc.)
- Quasi-Newton methods (NR, BFGS, DFP, etc.)

In the 1990s:

- Neural networks
- Learning rules: Descent, Momentum/Nesterov and Adaptive learning methods

In the 2000s:

- Gradient descent (by **observations**): Batch gradient descent (BGD), Stochastic gradient descent (SGD), Mini-batch gradient descent (MGD)
- Gradient descent (by **parameters**): Coordinate descent (CD), cyclical coordinate descent (CCD), Random coordinate descent (RCD)

# Coordinate descent methods

## Descent method

The descent algorithm is defined by the following rule:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)} = x^{(k)} - \eta^{(k)} D^{(k)}$$

At the  $k^{\text{th}}$  iteration, the current solution  $x^{(k)}$  is updated by going in the opposite direction to  $D^{(k)}$  (generally, we set  $D^{(k)} = \partial_x f(x^{(k)})$ )

## Coordinate descent method

Coordinate descent is a modification of the descent algorithm by minimizing the function along one coordinate at each step:

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)} = x_i^{(k)} - \eta^{(k)} D_i^{(k)}$$

⇒ The coordinate descent algorithm becomes a scalar problem

# Coordinate descent methods

Choice of the variable  $i$

1 Random coordinate descent (RCD)

We assign a random number between 1 and  $n$  to the index  $i$   
(Nesterov, 2012)

2 Cyclical coordinate descent (CCD)

We cyclically iterate through the coordinates (Tseng, 2001):

$$x_i^{(k+1)} = \arg \min_x f \left( x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x, x_{i+1}^{(k)}, \dots, x_n^{(k)} \right)$$

# Cyclical coordinate descent (CCD)

## Example 1

We consider the following function:

$$f(x_1, x_2, x_3) = (x_1 - 1)^2 + x_2^2 - x_2 + (x_3 - 2)^4 e^{x_1 - x_2 + 3}$$

We have:

$$D_1 = \frac{\partial f(x_1, x_2, x_3)}{\partial x_1} = 2(x_1 - 1) + (x_3 - 2)^4 e^{x_1 - x_2 + 3}$$

$$D_2 = \frac{\partial f(x_1, x_2, x_3)}{\partial x_2} = 2x_2 - 1 - (x_3 - 2)^4 e^{x_1 - x_2 + 3}$$

$$D_3 = \frac{\partial f(x_1, x_2, x_3)}{\partial x_3} = 4(x_3 - 2)^3 e^{x_1 - x_2 + 3}$$

# Cyclical coordinate descent (CCD)

The CCD algorithm is defined by the following iterations:

$$\left\{ \begin{array}{l} x_1^{(k+1)} = x_1^{(k)} - \eta^{(k)} \left( 2 \left( x_1^{(k)} - 1 \right) + \left( x_3^{(k)} - 2 \right)^4 e^{x_1^{(k)} - x_2^{(k)} + 3} \right) \\ x_2^{(k+1)} = x_2^{(k)} - \eta^{(k)} \left( 2x_2^{(k)} - 1 - \left( x_3^{(k)} - 2 \right)^4 e^{x_1^{(k+1)} - x_2^{(k)} + 3} \right) \\ x_3^{(k+1)} = x_3^{(k)} - \eta^{(k)} \left( 4 \left( x_3^{(k)} - 2 \right)^3 e^{x_1^{(k+1)} - x_2^{(k+1)} + 3} \right) \end{array} \right.$$

We have the following scheme:

$$\begin{aligned} (x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) &\rightarrow x_1^{(1)} \rightarrow (x_1^{(1)}, x_2^{(0)}, x_3^{(0)}) \rightarrow x_2^{(1)} \rightarrow (x_1^{(1)}, x_2^{(1)}, x_3^{(0)}) \rightarrow x_3^{(1)} \rightarrow \\ (x_1^{(1)}, x_2^{(1)}, x_3^{(1)}) &\rightarrow x_1^{(2)} \rightarrow (x_1^{(2)}, x_2^{(1)}, x_3^{(1)}) \rightarrow x_2^{(2)} \rightarrow (x_1^{(2)}, x_2^{(2)}, x_3^{(1)}) \rightarrow x_3^{(2)} \rightarrow \\ (x_1^{(2)}, x_2^{(2)}, x_3^{(2)}) &\rightarrow x_1^{(3)} \rightarrow \dots \end{aligned}$$

# Cyclical coordinate descent (CCD)

**Table 1:** Solution obtained with the CCD algorithm ( $\eta^{(k)} = 0.25$ )

$k$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$D_1^{(k)}$	$D_2^{(k)}$	$D_3^{(k)}$
0	1.0000	1.0000	1.0000			
1	-4.0214	0.7831	1.1646	20.0855	0.8675	-0.6582
2	-1.5307	0.8834	2.2121	-9.9626	-0.4013	-4.1902
3	-0.2663	0.6949	2.1388	-5.0578	0.7540	0.2932
4	0.3661	0.5988	2.0962	-2.5297	0.3845	0.1703
5	0.6827	0.5499	2.0758	-1.2663	0.1957	0.0818
6	0.8412	0.5252	2.0638	-0.6338	0.0989	0.0480
7	0.9205	0.5127	2.0560	-0.3172	0.0498	0.0314
8	0.9602	0.5064	2.0504	-0.1588	0.0251	0.0222
9	0.9800	0.5033	2.0463	-0.0795	0.0126	0.0166
$\infty$	1.0000	0.5000	2.0000	0.0000	0.0000	0.0000

# The lasso revolution

## Least absolute shrinkage and selection operator (lasso)

The lasso method consists in adding a  $\ell_1$  penalty function to the least square problem:

$$\begin{aligned}\hat{\beta}^{\text{lasso}}(\tau) &= \arg \min \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) \\ \text{s.t. } \|\beta\|_1 &= \sum_{j=1}^m |\beta_j| \leq \tau\end{aligned}$$

This problem is equivalent to:

$$\hat{\beta}^{\text{lasso}}(\lambda) = \arg \min \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) + \lambda \|\beta\|_1$$

We have:

$$\tau = \left\| \hat{\beta}^{\text{lasso}}(\lambda) \right\|_1$$



# Solving the lasso regression problem

We introduce the parametrization:

$$\beta = \begin{pmatrix} I_m & -I_m \end{pmatrix} \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix} = \beta^+ - \beta^-$$

under the constraints  $\beta^+ \geq \mathbf{0}_m$  and  $\beta^- \geq \mathbf{0}_m$ . We deduce that:

$$\|\beta\|_1 = \sum_{j=1}^m |\beta_j^+ - \beta_j^-| = \sum_{j=1}^m |\beta_j^+| + \sum_{j=1}^m |\beta_j^-| = \mathbf{1}_m^\top \beta^+ + \mathbf{1}_m^\top \beta^-$$

# Solving the lasso regression problem

## Augmented QP program of the lasso regression ( $\lambda$ -problem)

The augmented QP program is specified as follows:

$$\begin{aligned}\hat{\theta} &= \arg \min \frac{1}{2} \theta^\top Q \theta - \theta^\top R \\ \text{s.t. } &\theta \geq \mathbf{0}_{2m}\end{aligned}$$

where  $\theta = (\beta^+, \beta^-)$ ,  $\tilde{X} = \begin{pmatrix} X & -X \end{pmatrix}$ ,  $Q = \tilde{X}^\top \tilde{X}$  and  $R = \tilde{X}^\top Y + \lambda \mathbf{1}_{2m}$ . If we denote  $T = \begin{pmatrix} I_m & -I_m \end{pmatrix}$ , we obtain:

$$\hat{\beta}^{\text{lasso}}(\lambda) = T \hat{\theta}$$

# Solving the lasso regression problem

## Augmented QP program of the lasso regression ( $\tau$ -problem)

If we consider the  $\tau$ -problem, we obtain another augmented QP program:

$$\begin{aligned} \hat{\theta} &= \arg \min \frac{1}{2} \theta^\top Q \theta - \theta^\top R \\ \text{s.t.} & \begin{cases} C \theta \leq D \\ \theta \geq \mathbf{0}_{2m} \end{cases} \end{aligned}$$

where  $Q = \tilde{X}^\top \tilde{X}$ ,  $R = \tilde{X}^\top Y$ ,  $C = \mathbf{1}_{2m}^\top$  and  $D = \tau$ . Again, we have:

$$\hat{\beta}(\tau) = T \hat{\theta}$$

# Solving the lasso regression problem

We consider the linear regression:

$$Y = X\beta + \varepsilon$$

where  $Y$  is a  $n \times 1$  vector,  $X$  is a  $n \times m$  matrix and  $\beta$  is a  $m \times 1$  vector.  
 The optimization problem is:

$$\hat{\beta} = \arg \min f(\beta) = \frac{1}{2} (Y - X\beta)^\top (Y - X\beta)$$

Since we have  $\partial_\beta f(\beta) = -X^\top (Y - X\beta)$ , we deduce that:

$$\begin{aligned} \frac{\partial f(\beta)}{\partial \beta_j} &= x_j^\top (X\beta - Y) \\ &= x_j^\top (x_j\beta_j + X_{(-j)}\beta_{(-j)} - Y) \\ &= x_j^\top x_j\beta_j + x_j^\top X_{(-j)}\beta_{(-j)} - x_j^\top Y \end{aligned}$$

where  $x_j$  is the  $n \times 1$  vector corresponding to the  $j^{\text{th}}$  variable and  $X_{(-j)}$  is the  $n \times (m - 1)$  matrix (without the  $j^{\text{th}}$  variable)

# Solving the lasso regression problem

At the optimum, we have  $\partial_{\beta_j} f(\beta) = 0$  or:

$$\beta_j = \frac{x_j^\top Y - x_j^\top X_{(-j)} \beta_{(-j)}}{x_j^\top x_j} = \frac{x_j^\top (Y - X_{(-j)} \beta_{(-j)})}{x_j^\top x_j}$$

## CCD algorithm for the linear regression

We have:

$$\beta_j^{(k+1)} = \frac{x_j^\top \left( Y - \sum_{j'=1}^{j-1} x_{j'} \beta_{j'}^{(k+1)} - \sum_{j'=j+1}^m x_{j'} \beta_{j'}^{(k)} \right)}{x_j^\top x_j}$$

⇒ Introducing pointwise constraints is straightforward

# Solving the lasso regression problem

The objective function becomes:

$$\begin{aligned} f(\beta) &= \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) + \lambda \|\beta\|_1 \\ &= f_{\text{OLS}}(\beta) + \lambda \|\beta\|_1 \end{aligned}$$

Since the norm is separable —  $\|\beta\|_1 = \sum_{j=1}^m |\beta_j|$ , the first-order condition is:

$$\frac{\partial f_{\text{OLS}}(\beta)}{\partial \beta_j} + \lambda \partial |\beta_j| = 0$$

or:

$$\underbrace{(x_j^\top x_j)}_c \beta_j - \underbrace{x_j^\top (Y - X_{(-j)} \beta_{(-j)})}_v + \lambda \partial |\beta_j| = 0$$

# Derivation of the soft-thresholding operator

We consider the following equation:

$$c\beta_j - v + \lambda \partial |\beta_j| \in \{0\}$$

where  $c > 0$  and  $\lambda > 0$ . Since we have  $\partial |\beta_j| = \text{sign}(\beta_j)$ , we deduce that:

$$\beta_j^* = \begin{cases} c^{-1}(v + \lambda) & \text{if } \beta_j^* < 0 \\ 0 & \text{if } \beta_j^* = 0 \\ c^{-1}(v - \lambda) & \text{if } \beta_j^* > 0 \end{cases}$$

If  $\beta_j^* < 0$  or  $\beta_j^* > 0$ , then we have  $v + \lambda < 0$  or  $v - \lambda > 0$ . This is equivalent to set  $|v| > \lambda > 0$ . The case  $\beta_j^* = 0$  implies that  $|v| \leq \lambda$ . We deduce that:

$$\beta_j^* = c^{-1} \cdot \mathcal{S}(v; \lambda)$$

where  $\mathcal{S}(v; \lambda)$  is the soft-thresholding operator:

$$\begin{aligned} \mathcal{S}(v; \lambda) &= \begin{cases} 0 & \text{if } |v| \leq \lambda \\ v - \lambda \text{sign}(v) & \text{otherwise} \end{cases} \\ &= \text{sign}(v) \cdot (|v| - \lambda)_+ \end{aligned}$$

# Solving the lasso regression problem

## CCD algorithm for the lasso regression

We have:

$$\beta_j^{(k+1)} = \frac{1}{x_j^\top x_j} \mathcal{S} \left( x_j^\top \left( Y - \sum_{j'=1}^{j-1} x_{j'} \beta_{j'}^{(k+1)} - \sum_{j'=j+1}^m x_{j'} \beta_{j'}^{(k)} \right); \lambda \right)$$

where  $\mathcal{S}(v; \lambda)$  is the **soft-thresholding operator**:

$$\mathcal{S}(v; \lambda) = \text{sign}(v) \cdot (|v| - \lambda)_+$$



# Solving the lasso regression problem

Table 2: Matlab code

```
for k = 1:nIters
    for j = 1:m
        x_j = X(:,j);
        X_j = X;
        X_j(:,j) = zeros(n,1);
        if lambda > 0
            v = x_j'*(Y - X_j*beta);
            beta(j) = max(abs(v) - lambda,0) * sign(v) / (x_j'*x_j);
        else
            beta(j) = x_j'*(Y - X_j*beta) / (x_j'*x_j);
        end
    end
end
```

# Solving the lasso regression problem

## Example 2

We consider the following data:

$i$	$y$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
1	3.1	2.8	4.3	0.3	2.2	3.5
2	24.9	5.9	3.6	3.2	0.7	6.4
3	27.3	6.0	9.6	7.6	9.5	0.9
4	25.4	8.4	5.4	1.8	1.0	7.1
5	46.1	5.2	7.6	8.3	0.6	4.5
6	45.7	6.0	7.0	9.6	0.6	0.6
7	47.4	6.1	1.0	8.5	9.6	8.6
8	-1.8	1.2	9.6	2.7	4.8	5.8
9	20.8	3.2	5.0	4.2	2.7	3.6
10	6.8	0.5	9.2	6.9	9.3	0.7
11	12.9	7.9	9.1	1.0	5.9	5.4
12	37.0	1.8	1.3	9.2	6.1	8.3
13	14.7	7.4	5.6	0.9	5.6	3.9
14	-3.2	2.3	6.6	0.0	3.6	6.4
15	44.3	7.7	2.2	6.5	1.3	0.7

# Solving the lasso regression problem

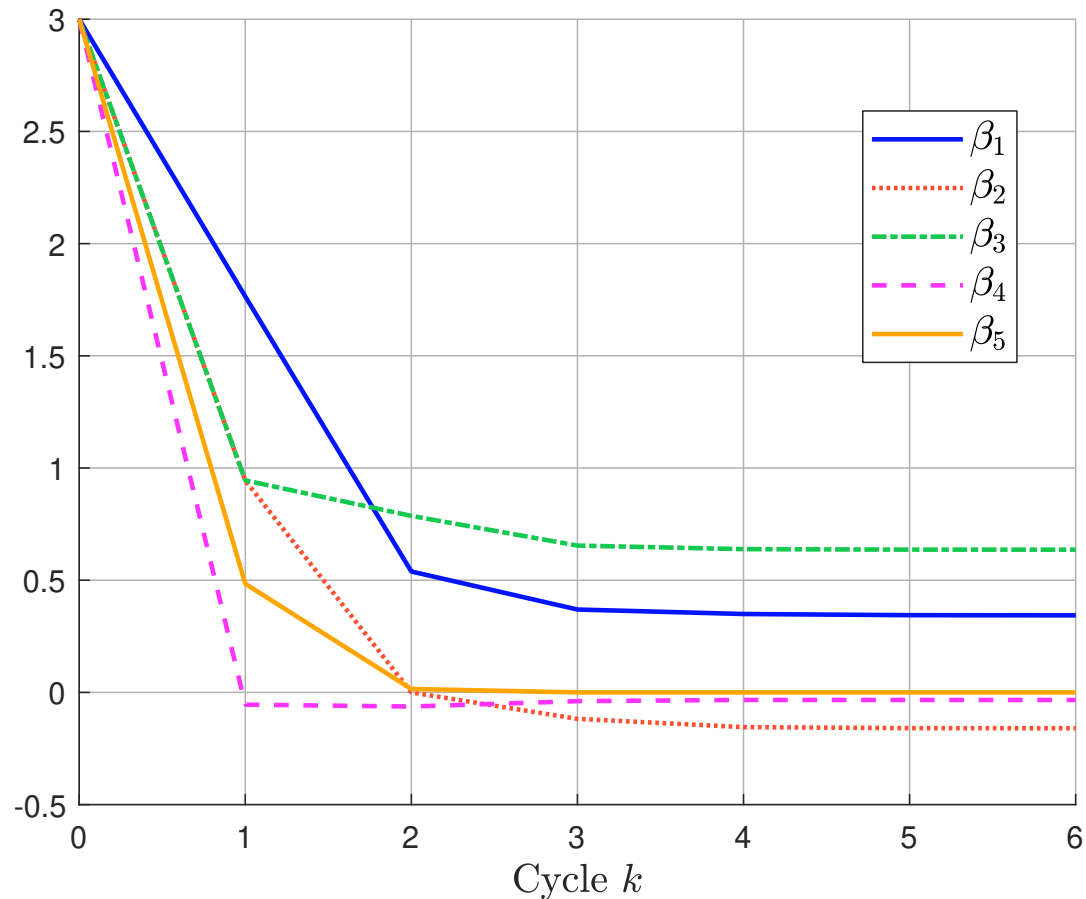


Figure 1: Convergence of the CCD algorithm (lasso regression,  $\lambda = 2$ )

Note: we start the CCD algorithm with  $\beta_j^{(0)} = 0$  (don't forget to standardize the data!)

# Solving the lasso regression problem

- 1 The dimension problem is  $(2m, 2m)$  for QP and  $(1, 0)$  for CCD!
- 2 CCD is faster for lasso regression than for linear regression (because of the soft-thresholding operator)!

**Suppose  $n = 50\,000$  and  $m = 1\,000\,000$  (DNA sequence problem!)**

# Solving the lasso regression problem

## Example 3

- We consider an experiment with  $n = 100\,000$  observations and  $m = 50$  variables.
- The design matrix  $X$  is built using the uniform distribution while the residuals are simulated using a Gaussian distribution and a standard deviation of 20%.
- The beta coefficients are distributed uniformly between  $-3$  and  $+3$  except four coefficients that take a larger value.
- We then standardize the data of  $X$  and  $Y$ .
- For initializing the coordinates, we use uniform random numbers between  $-1$  and  $+1$ .

# Solving the lasso regression problem

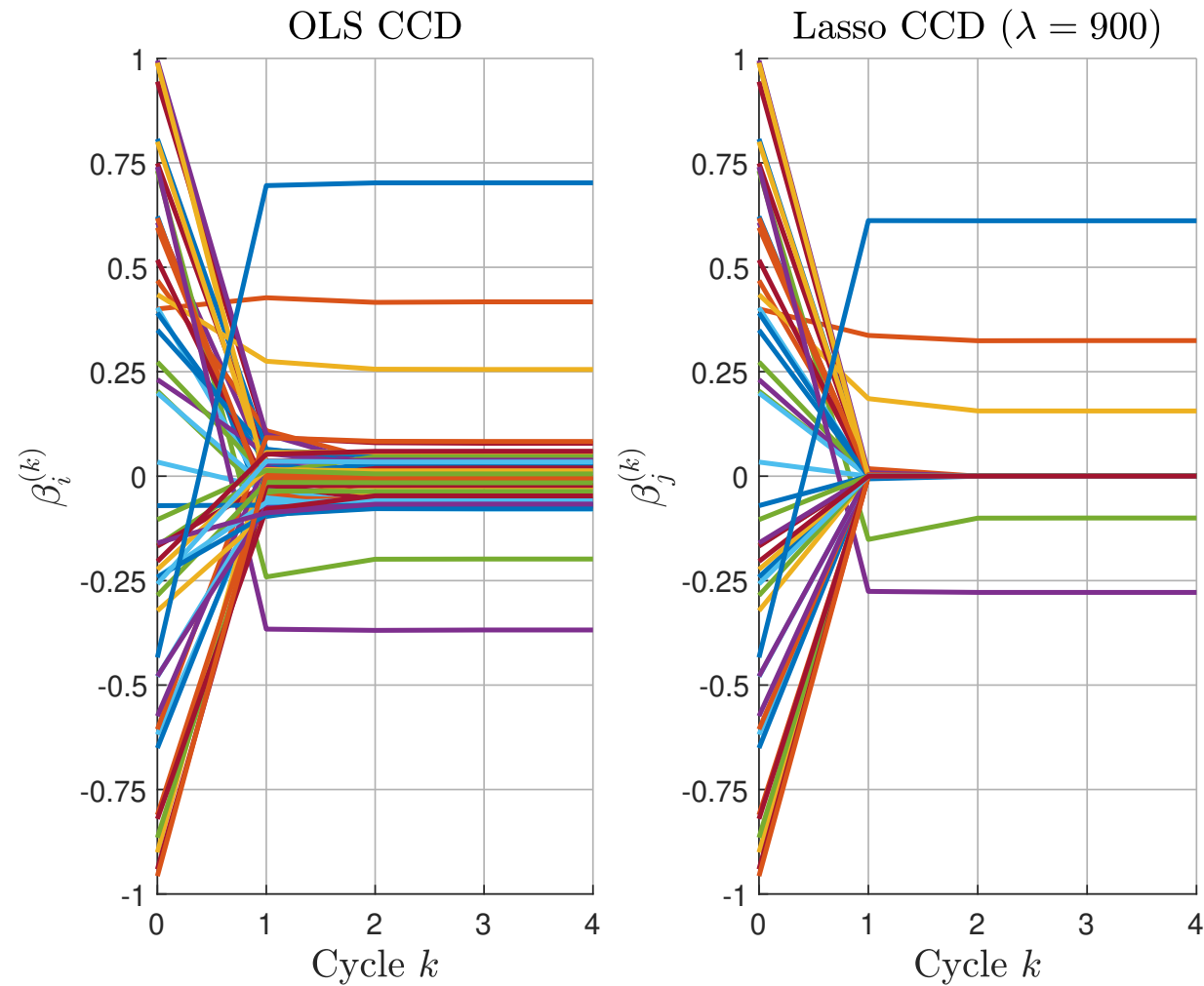


Figure 2: Convergence of the CCD algorithm (lasso vs linear regression)

# Alternative direction method of multipliers

## Definition

The alternating direction method of multipliers (ADMM) is an algorithm introduced by Gabay and Mercier (1976) to solve optimization problems which can be expressed as:

$$\begin{aligned} \{x^*, y^*\} &= \arg \min_{(x,y)} f_x(x) + f_y(y) \\ \text{s.t. } & Ax + By = c \end{aligned}$$

The algorithm is:

$$\begin{aligned} x^{(k+1)} &= \arg \min_x \left\{ f_x(x) + \frac{\varphi}{2} \left\| Ax + By^{(k)} - c + u^{(k)} \right\|_2^2 \right\} \\ y^{(k+1)} &= \arg \min_y \left\{ f_y(y) + \frac{\varphi}{2} \left\| Ax^{(k+1)} + By - c + u^{(k)} \right\|_2^2 \right\} \\ u^{(k+1)} &= u^{(k)} + \left( Ax^{(k+1)} + By^{(k+1)} - c \right) \end{aligned}$$

# Alternative direction method of multipliers

What is the underlying idea?

- Minimizing  $f_x(x) + f_y(y)$  with respect to  $(x, y)$  is a difficult task
- Minimizing

$$g_x(x) = f_x(x) + \frac{\varphi}{2} \|Ax + By - c\|_2^2$$

with respect to  $x$  and minimizing

$$g_y(y) = f_y(y) + \frac{\varphi}{2} \|Ax + By - c\|_2^2$$

with respect to  $y$  is easier



# Alternative direction method of multipliers

We use the following notations:

- $f_x^{(k+1)}(x)$  is the objective function of the  $x$ -update step:

$$f_x^{(k+1)}(x) = f_x(x) + \frac{\varphi}{2} \left\| Ax + By^{(k)} - c + u^{(k)} \right\|_2^2$$

- $f_y^{(k+1)}(y)$  is the objective function of the  $y$ -update step:

$$f_y^{(k+1)}(y) = f_y(y) + \frac{\varphi}{2} \left\| Ax^{(k+1)} + By - c + u^{(k)} \right\|_2^2$$

# Alternative direction method of multipliers

When  $A = I_n$  and  $B = -I_n$ , we have:

1

$$Ax + By^{(k)} - c + u^{(k)} = x - y^{(k)} - c + u^{(k)} = x - v_x^{(k+1)}$$

where:

$$v_x^{(k+1)} = y^{(k)} + c - u^{(k)}$$

2

$$Ax^{(k+1)} + By - c + u^{(k)} = x^{(k+1)} - y - c + u^{(k)} = v_y^{(k+1)} - y$$

where:

$$v_y^{(k+1)} = x^{(k+1)} - c + u^{(k)}$$

3

$$f_x^{(k+1)}(x) = f_x(x) + \frac{\varphi}{2} \left\| x - v_x^{(k+1)} \right\|_2^2$$

$$f_y^{(k+1)}(y) = f_y(y) + \frac{\varphi}{2} \left\| y - v_y^{(k+1)} \right\|_2^2$$

# Alternative direction method of multipliers

- We consider a problem of the form:

$$x^* = \arg \min_x g(x)$$

The idea is then to write  $g(x)$  as a separable function:

$$g(x) = g_1(x) + g_2(x)$$

and to consider the following equivalent ADMM problem:

$$\begin{aligned} \{x^*, y^*\} &= \arg \min_{(x,y)} f_x(x) + f_y(y) \\ \text{s.t. } &x = y \end{aligned}$$

where  $f_x(x) = g_1(x)$  and  $f_y(y) = g_2(y)$

# Alternative direction method of multipliers

- We consider a problem of the form:

$$\begin{aligned} x^* &= \arg \min_x g(x) \\ \text{s.t. } &x \in \Omega \end{aligned}$$

We have:

$$\begin{aligned} \{x^*, y^*\} &= \arg \min_{(x,y)} f_x(x) + f_y(y) \\ \text{s.t. } &x = y \end{aligned}$$

where  $f_x(x) = g(x)$ ,  $f_y(y) = \mathbb{1}_\Omega(y)$  and:

$$\mathbb{1}_\Omega(y) = \begin{cases} 0 & \text{if } y \in \Omega \\ +\infty & \text{if } y \notin \Omega \end{cases}$$

# Alternative direction method of multipliers

## Special case

$$\Omega = \{x : x^- \leq x \leq x^+\}$$

By setting  $\varphi = 1$ , the  $y$ -step becomes:

$$\begin{aligned} y^{(k+1)} &= \arg \min \left\{ \mathbb{1}_\Omega(y) + \frac{1}{2} \left\| x^{(k+1)} - y + u^{(k)} \right\|_2^2 \right\} \\ &= \mathbf{prox}_{f_y} \left( x^{(k+1)} + u^{(k)} \right) \end{aligned}$$

where the proximal operator is the box projection or the truncation operator:

$$\begin{aligned} \mathbf{prox}_{f_y}(v) &= x^- \odot \mathbb{1}\{v < x^-\} + \\ &\quad v \odot \mathbb{1}\{x^- \leq v \leq x^+\} + \\ &\quad x^+ \odot \mathbb{1}\{v > x^+\} \\ &= \mathcal{T}(v; x^-, x^+) \end{aligned}$$

# Alternative direction method of multipliers

## Special case

$$\Omega = \{x : x^- \leq x \leq x^+\}$$

The ADMM algorithm is then:

$$\begin{aligned}x^{(k+1)} &= \arg \min \left\{ g(x) + \frac{1}{2} \left\| x - y^{(k)} + u^{(k)} \right\|_2^2 \right\} \\y^{(k+1)} &= \mathbf{prox}_{f_y} \left( x^{(k+1)} + u^{(k)} \right) \\u^{(k+1)} &= u^{(k)} + \left( x^{(k+1)} - y^{(k+1)} \right)\end{aligned}$$

⇒ Solving the constrained optimization problem consists in solving the unconstrained optimization problem, applying the box projection and iterating these steps until convergence

# Alternative direction method of multipliers

## Lasso regression

The  $\lambda$ -problem of the lasso regression has the following ADMM formulation:

$$\begin{aligned} \{\beta^*, \bar{\beta}^*\} &= \arg \min \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) + \lambda \|\bar{\beta}\|_1 \\ \text{s.t. } &\beta - \bar{\beta} = \mathbf{0}_m \end{aligned}$$

We have:

$$\begin{aligned} f_x(\beta) &= \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) \\ &= \frac{1}{2} \beta^\top (X^\top X) \beta - \beta^\top (X^\top Y) + \frac{1}{2} Y^\top Y \end{aligned}$$

and:

$$f_y(\bar{\beta}) = \lambda \|\bar{\beta}\|_1$$

# Alternative direction method of multipliers

The x-step is:

$$\beta^{(k+1)} = \arg \min_{\beta} \left\{ \frac{1}{2} \beta^{\top} (X^{\top} X) \beta - \beta^{\top} (X^{\top} Y) + \frac{\varphi}{2} \left\| \beta - \bar{\beta}^{(k)} + u^{(k)} \right\|_2^2 \right\}$$

Since we have:

$$\begin{aligned} \frac{\varphi}{2} \left\| \beta - \bar{\beta}^{(k)} + u^{(k)} \right\|_2^2 &= \frac{\varphi}{2} \beta^{\top} \beta - \varphi \beta^{\top} (\bar{\beta}^{(k)} - u^{(k)}) + \\ &\quad \frac{\varphi}{2} (\bar{\beta}^{(k)} - u^{(k)})^{\top} (\bar{\beta}^{(k)} - u^{(k)}) \end{aligned}$$

we deduce that the x-update is a standard QP problem where:

$$f_x^{(k+1)}(\beta) = \frac{1}{2} \beta^{\top} (X^{\top} X + \varphi I_m) \beta - \beta^{\top} (X^{\top} Y + \varphi (\bar{\beta}^{(k)} - u^{(k)}))$$

It follows that the solution is:

$$\begin{aligned} \beta^{(k+1)} &= \arg \min_{\beta} f_x^{(k+1)}(\beta) \\ &= (X^{\top} X + \varphi I_m)^{-1} (X^{\top} Y + \varphi (\bar{\beta}^{(k)} - u^{(k)})) \end{aligned}$$



## Alternative direction method of multipliers

The  $y$ -step is:

$$\begin{aligned}\bar{\beta}^{(k+1)} &= \arg \min_{\bar{\beta}} \left\{ \lambda \|\bar{\beta}\|_1 + \frac{\varphi}{2} \left\| \beta^{(k+1)} - \bar{\beta} + u^{(k)} \right\|_2^2 \right\} \\ &= \arg \min_{\bar{\beta}} \left\{ \frac{1}{2} \left\| \bar{\beta} - \left( \beta^{(k+1)} + u^{(k)} \right) \right\|_2^2 + \frac{\lambda}{\varphi} \|\bar{\beta}\|_1 \right\}\end{aligned}$$

We recognize the soft-thresholding problem with  $v = \beta^{(k+1)} + u^{(k)}$ . We have:

$$\bar{\beta}^{(k+1)} = \mathcal{S} \left( \beta^{(k+1)} + u^{(k)}; \varphi^{-1} \lambda \right)$$

where:

$$\mathcal{S}(v; \lambda) = \text{sign}(v) \cdot (|v| - \lambda)_+$$

# Alternative direction method of multipliers

## ADMM-Lasso algorithm (Boyd *et al.*, 2011)

Finally, the ADMM algorithm is made up of the following steps:

$$\begin{cases} \beta^{(k+1)} = (X^T X + \varphi I_m)^{-1} (X^T Y + \varphi (\bar{\beta}^{(k)} - u^{(k)})) \\ \bar{\beta}^{(k+1)} = \mathcal{S}(\beta^{(k+1)} + u^{(k)}; \varphi^{-1} \lambda) \\ u^{(k+1)} = u^{(k)} + (\beta^{(k+1)} - \bar{\beta}^{(k+1)}) \end{cases}$$

# Alternative direction method of multipliers

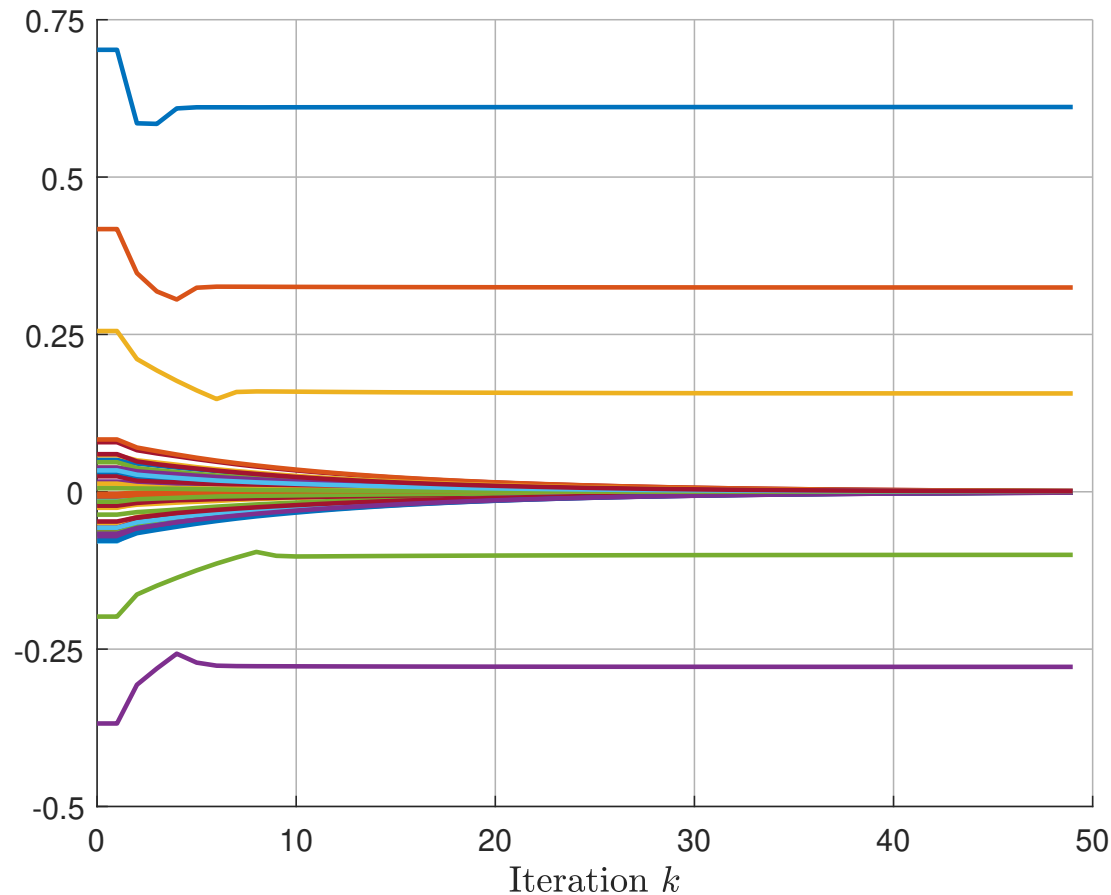


Figure 3: Convergence of the ADMM algorithm (Example 3,  $\lambda = 900$ )

Note: the initial values are the OLS estimates and we set  $\varphi = \lambda$

# Alternative direction method of multipliers

In practice, we use a time-varying parameter  $\varphi^{(k)}$  (see Perrin and Roncalli, 2020).

# Proximal operator

## Definition

The proximal operator  $\mathbf{prox}_f(v)$  of the function  $f(x)$  is defined by:

$$\mathbf{prox}_f(v) = x^* = \arg \min_x \left\{ f_v(x) = f(x) + \frac{1}{2} \|x - v\|_2^2 \right\}$$

# Proximal operator

## Example 4

We consider the scalar-valued logarithmic barrier function  $f(x) = -\lambda \ln x$

# Proximal operator

We have:

$$\begin{aligned} f_v(x) &= -\lambda \ln x + \frac{1}{2} (x - v)^2 \\ &= -\lambda \ln x + \frac{1}{2} x^2 - xv + \frac{1}{2} v^2 \end{aligned}$$

The first-order condition is  $-\lambda x^{-1} + x - v = 0$ . We obtain two roots with opposite signs:

$$x' = \frac{v - \sqrt{v^2 + 4\lambda}}{2} \quad \text{and} \quad x'' = \frac{v + \sqrt{v^2 + 4\lambda}}{2}$$

Since the logarithmic function is defined for  $x > 0$ , we deduce that:

$$\mathbf{prox}_f(v) = \frac{v + \sqrt{v^2 + 4\lambda}}{2}$$

# Proximal operator

In the case where  $f(x) = \mathbb{1}_\Omega(x)$ , we have:

$$\begin{aligned}\mathbf{prox}_f(v) &= \arg \min_x \left\{ \mathbb{1}_\Omega(x) + \frac{1}{2} \|x - v\|_2^2 \right\} \\ &= \arg \min_{x \in \Omega} \left\{ \|x - v\|_2^2 \right\} \\ &= \mathcal{P}_\Omega(v)\end{aligned}$$

where  $\mathcal{P}_\Omega(v)$  is the standard projection of  $v$  onto  $\Omega$



# Proximal operator

Table 3: Projection for some simple polyhedra

Notation	$\Omega$	$\mathcal{P}_\Omega(v)$
$\mathcal{A}ffineset [A, B]$	$Ax = B$	$v - A^\dagger (Av - B)$
$\mathcal{H}yperplane [a, b]$	$a^\top x = b$	$v - \frac{(a^\top v - b)}{\ a\ _2^2} a$
$\mathcal{H}alfspace [c, d]$	$c^\top x \leq d$	$v - \frac{(c^\top v - d)_+}{\ c\ _2^2} c$
$\mathcal{B}ox [x^-, x^+]$	$x^- \leq x \leq x^+$	$\mathcal{T}(v; x^-, x^+)$

Source: Parikh and Boyd (2014)

Note:  $A^\dagger$  is the Moore-Penrose pseudo-inverse of  $A$ , and  $\mathcal{T}(v; x^-, x^+)$  is the truncation operator

Remark: No analytical formula for the (multi-dimensional) inequality constraint  $Cx \leq D \Rightarrow$  it may be solved using the Dykstra's algorithm

# Proximal operator

## Separable sum

If  $f(x) = \sum_{i=1}^n f_i(x_i)$  is fully separable, then the proximal of  $f(v)$  is the vector of the proximal operators applied to each scalar-valued function  $f_i(x_i)$ :

$$\mathbf{prox}_f(v) = \begin{pmatrix} \mathbf{prox}_{f_1}(v_1) \\ \vdots \\ \mathbf{prox}_{f_n}(v_n) \end{pmatrix}$$

# Proximal operator

If  $f(x) = -\lambda \ln x$ , we have:

$$\mathbf{prox}_f(v) = \frac{v + \sqrt{v^2 + 4\lambda}}{2}$$

In the case of the vector-valued logarithmic barrier  $f(x) = -\lambda \sum_{i=1}^n \ln x_i$ , we deduce that:

$$\mathbf{prox}_f(v) = \frac{v + \sqrt{v \odot v + 4\lambda}}{2}$$

# Proximal operator

## Moreau decomposition theorem

We have:

$$\mathbf{prox}_f(v) + \mathbf{prox}_{f^*}(v) = v$$

where  $f^*$  is the convex conjugate of  $f$ .

## Application

If  $f(x)$  is a  $\ell_q$ -norm function, then  $f^*(x) = \mathbb{1}_{\mathcal{B}_p}(x)$  where  $\mathcal{B}_p$  is the  $\ell_p$  unit ball and  $p^{-1} + q^{-1} = 1$ . Since we have  $\mathbf{prox}_{f^*}(v) = \mathcal{P}_{\mathcal{B}_p}(v)$ , we deduce that:

$$\mathbf{prox}_f(v) + \mathcal{P}_{\mathcal{B}_p}(v) = v$$

The proximal of the  $\ell_p$ -ball can be deduced from the proximal operator of the  $\ell_q$ -norm function.

# Proximal operator

**Table 4:** Proximal of the  $\ell_p$ -norm function  $f(x) = \|x\|_p$

$p$	$\mathbf{prox}_{\lambda f}(v)$
$p = 1$	$\mathcal{S}(v; \lambda) = \text{sign}(v) \odot ( v  - \lambda \mathbf{1}_n)_+$
$p = 2$	$\left(1 - \frac{\lambda}{\max(\lambda, \ v\ _2)}\right) v$
$p = \infty$	$\text{sign}(v) \odot \mathbf{prox}_{\lambda \max x}( v )$

We have:

$$\mathbf{prox}_{\lambda \max x}(v) = \min(v, s^*)$$

where  $s^*$  is the solution of the following equation:

$$s^* = \left\{ s \in \mathbb{R} : \sum_{i=1}^n (v_i - s)_+ = \lambda \right\}$$

# Proximal operator

**Table 5:** Proximal of the  $\ell_p$ -ball  $\mathcal{B}_p(c, \lambda) = \{x \in \mathbb{R}^n : \|x - c\|_p \leq \lambda\}$  when  $c$  is equal to  $\mathbf{0}_n$

$p$	$\mathcal{P}_{\mathcal{B}_p(\mathbf{0}_n, \lambda)}(v)$	$q$
$p = 1$	$v - \text{sign}(v) \odot \mathbf{prox}_{\lambda \max_x}( v )$	$q = \infty$
$p = 2$	$v - \mathbf{prox}_{\lambda \ x\ _2}(v)$	$q = 2$
$p = \infty$	$\mathcal{T}(v; -\lambda, \lambda)$	$q = 1$

# Proximal operator

## Scaling and translation

Let us define  $g(x) = f(ax + b)$  where  $a \neq 0$ . We have:

$$\mathbf{prox}_g(v) = \frac{\mathbf{prox}_{a^2 f}(av + b) - b}{a}$$

## Application

We can use this property when the center  $c$  of the  $\ell_p$  ball is not equal to  $\mathbf{0}_n$ . Since we have  $\mathbf{prox}_g(v) = \mathbf{prox}_f(v - c) + c$  where  $g(x) = f(x - c)$  and the equivalence  $\mathcal{B}_p(\mathbf{0}_n, \lambda) = \{x \in \mathbb{R}^n : f(x) \leq \lambda\}$  where  $f(x) = \|x\|_p$ , we deduce that:

$$\mathcal{P}_{\mathcal{B}_p(c, \lambda)}(v) = \mathcal{P}_{\mathcal{B}_p(\mathbf{0}_n, \lambda)}(v - c) + c$$

# Application to the $\tau$ -problem of the lasso regression

We have:

$$\begin{aligned}\hat{\beta}(\tau) &= \arg \min_{\beta} \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) \\ \text{s.t. } &\|\beta\|_1 \leq \tau\end{aligned}$$

The ADMM formulation is:

$$\begin{aligned}\{\beta^*, \bar{\beta}^*\} &= \arg \min_{(\beta, \bar{\beta})} \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) + \mathbf{1}_\Omega(\bar{\beta}) \\ \text{s.t. } &\beta = \bar{\beta}\end{aligned}$$

where  $\Omega = \mathcal{B}_1(\mathbf{0}_m, \tau)$  is the centered  $\ell_1$  ball with radius  $\tau$



# Application to the $\tau$ -problem of the lasso regression

- 1 The  $x$ -update is:

$$\begin{aligned}\beta^{(k+1)} &= \arg \min_{\beta} \left\{ \frac{1}{2} (Y - X\beta)^\top (Y - X\beta) + \frac{\varphi}{2} \left\| \beta - \bar{\beta}^{(k)} + u^{(k)} \right\|_2^2 \right\} \\ &= (X^\top X + \varphi I_m)^{-1} \left( X^\top Y + \varphi \left( \bar{\beta}^{(k)} - u^{(k)} \right) \right)\end{aligned}$$

where  $v_x^{(k+1)} = \bar{\beta}^{(k)} - u^{(k)}$

# Application to the $\tau$ -problem of the lasso regression

2 The  $y$ -update is:

$$\begin{aligned}
 \bar{\beta}^{(k+1)} &= \arg \min_{\bar{\beta}} \left\{ \mathbf{1}_{\Omega}(\bar{\beta}) + \frac{\varphi}{2} \left\| \beta^{(k+1)} - \bar{\beta} + u^{(k)} \right\|_2^2 \right\} \\
 &= \mathbf{prox}_{f_y} \left( \beta^{(k+1)} + u^{(k)} \right) \\
 &= \mathcal{P}_{\Omega} \left( v_y^{(k+1)} \right) \\
 &= v_y^{(k+1)} - \text{sign} \left( v_y^{(k+1)} \right) \odot \mathbf{prox}_{\tau \max x} \left( \left| v_y^{(k+1)} \right| \right)
 \end{aligned}$$

where  $v_y^{(k+1)} = \beta^{(k+1)} + u^{(k)}$

# Application to the $\tau$ -problem of the lasso regression

- 3 The  $u$ -update is:

$$u^{(k+1)} = u^{(k)} + \beta^{(k+1)} - \bar{\beta}^{(k+1)}$$

# Application to the $\tau$ -problem of the lasso regression

## ADMM-Lasso algorithm

The ADMM algorithm is :

$$\begin{cases} \beta^{(k+1)} = (X^T X + \varphi I_m)^{-1} (X^T Y + \varphi (\bar{\beta}^{(k)} - u^{(k)})) \\ \bar{\beta}^{(k+1)} = \begin{cases} \mathcal{S}(\beta^{(k+1)} + u^{(k)}; \varphi^{-1} \lambda) & (\lambda\text{-problem}) \\ \mathcal{P}_{\mathcal{B}_1(\mathbf{0}_m, \tau)}(\beta^{(k+1)} + u^{(k)}) & (\tau\text{-problem}) \end{cases} \\ u^{(k+1)} = u^{(k)} + (\beta^{(k+1)} - \bar{\beta}^{(k+1)}) \end{cases}$$

## Remark

The ADMM algorithm is similar for  $\lambda$ - and  $\tau$ -problems since the only difference concerns the  $y$ -step. However, the  $\tau$ -problem is easier to solve with the ADMM algorithm from a practical point of view, because the  $y$ -update of the  $\tau$ -problem is independent of the penalization parameter  $\varphi$ .

# Derivation of the soft-thresholding operator

We consider the following equation:

$$cx - v + \lambda \partial |x| \in 0$$

where  $c > 0$  and  $\lambda > 0$ . Since we have  $\partial |x| = \text{sign}(x)$ , we deduce that:

$$x^* = \begin{cases} c^{-1}(v + \lambda) & \text{if } x^* < 0 \\ 0 & \text{if } x^* = 0 \\ c^{-1}(v - \lambda) & \text{if } x^* > 0 \end{cases}$$

If  $x^* < 0$  or  $x^* > 0$ , then we have  $v + \lambda < 0$  or  $v - \lambda > 0$ . This is equivalent to set  $|v| > \lambda > 0$ . The case  $x^* = 0$  implies that  $|v| \leq \lambda$ . We deduce that:

$$x^* = c^{-1} \cdot \mathcal{S}(v; \lambda)$$

where  $\mathcal{S}(v; \lambda)$  is the soft-thresholding operator:

$$\begin{aligned} \mathcal{S}(v; \lambda) &= \begin{cases} 0 & \text{if } |v| \leq \lambda \\ v - \lambda \text{sign}(v) & \text{otherwise} \end{cases} \\ &= \text{sign}(v) \cdot (|v| - \lambda)_+ \end{aligned}$$

# Derivation of the soft-thresholding operator

We use the result on the separable sum

## Remark

If  $f(x) = \lambda \|x\|_1$ , we have  $f(x) = \lambda \sum_{i=1}^n |x_i|$  and  $f_i(x_i) = \lambda |x_i|$ . We deduce that the proximal operator of  $f(x)$  is the vector formulation of the soft-thresholding operator:

$$\mathbf{prox}_{\lambda \|x\|_1}(v) = \begin{pmatrix} \text{sign}(v_1) \cdot (|v_1| - \lambda)_+ \\ \vdots \\ \text{sign}(v_n) \cdot (|v_n| - \lambda)_+ \end{pmatrix} = \text{sign}(v) \odot (|v| - \lambda \mathbf{1}_n)_+$$

The soft-thresholding operator is the proximal operator of the  $\ell_1$ -norm  $f(x) = \|x\|_1$ . Indeed, we have  $\mathbf{prox}_f(v) = \mathcal{S}(v; 1)$  and  $\mathbf{prox}_{\lambda f}(v) = \mathcal{S}(v; \lambda)$ .

# Dykstra's algorithm

We consider the following optimization problem:

$$\begin{aligned} x^* &= \arg \min f_x(x) \\ \text{s.t. } &x \in \Omega \end{aligned}$$

where  $\Omega$  is a complex set of constraints:

$$\Omega = \Omega_1 \cap \Omega_2 \cap \dots \cap \Omega_m$$

We set  $y = x$  and  $f_y(y) = \mathbb{1}_\Omega(y)$ . The ADMM algorithm becomes

$$\begin{aligned} x^{(k+1)} &= \arg \min \left\{ f_x(x) + \frac{\varphi}{2} \left\| x - y^{(k)} + u^{(k)} \right\|_2^2 \right\} \\ v^{(k)} &= x^{(k+1)} + u^{(k)} \\ y^{(k+1)} &= \mathcal{P}_\Omega(v^{(k)}) \\ u^{(k+1)} &= u^{(k)} + (x^{(k+1)} - y^{(k+1)}) \end{aligned}$$

**How to compute  $\mathcal{P}_\Omega(v)$ ?**

# Dijkstra's algorithm

More generally, we consider the proximal optimization problem where the function  $f(x)$  is the convex sum of basic functions  $f_j(x)$ :

$$x^* = \arg \min_x \left\{ \sum_{j=1}^m f_j(x) + \frac{1}{2} \|x - v\|_2^2 \right\}$$

and the proximal of each basic function is known.

**How to find the solution  $x^*$ ?**



# Dykstra's algorithm

The case  $m = 2$

- We know the proximal solution of the  $\ell_1$ -norm function  
 $f_1(x) = \lambda_1 \|x\|_1$
- We know the proximal solution of the logarithmic barrier function  
 $f_2(x) = \lambda_2 \sum_{i=1}^n \ln x_i$
- We don't know how to compute the proximal operator of  
 $f(x) = f_1(x) + f_2(x)$ :

$$\begin{aligned} x^* &= \arg \min_x f_1(x) + f_2(x) + \frac{1}{2} \|x - v\|_2^2 \\ &= \mathbf{prox}_f(v) \end{aligned}$$

# Dykstra's algorithm

The case  $m = 2$

The Dykstra's algorithm consists in the following iterations:

$$\begin{cases} x^{(k+1)} = \mathbf{prox}_{f_1} (y^{(k)} + p^{(k)}) \\ p^{(k+1)} = y^{(k)} + p^{(k)} - x^{(k+1)} \\ y^{(k+1)} = \mathbf{prox}_{f_2} (x^{(k+1)} + q^{(k)}) \\ q^{(k+1)} = x^{(k+1)} + q^{(k)} - y^{(k+1)} \end{cases}$$

where  $x^{(0)} = y^{(0)} = v$  and  $p^{(0)} = q^{(0)} = \mathbf{0}_n$

# Dykstra's algorithm

The case  $m = 2$

This algorithm is related to the Douglas-Rachford splitting framework:

$$\begin{cases} x^{(k+\frac{1}{2})} = \mathbf{prox}_{f_1} (x^{(k)} + p^{(k)}) \\ p^{(k+1)} = p^{(k)} - \Delta_{1/2} x^{(k+\frac{1}{2})} \\ x^{(k+1)} = \mathbf{prox}_{f_2} (x^{(k+\frac{1}{2})} + q^{(k)}) \\ q^{(k+1)} = q^{(k)} - \Delta_{1/2} x^{(k+1)} \end{cases}$$

where  $\Delta_h x^{(k)} = x^{(k)} - x^{(k-h)}$

# Dykstra's algorithm

The case  $m = 2$

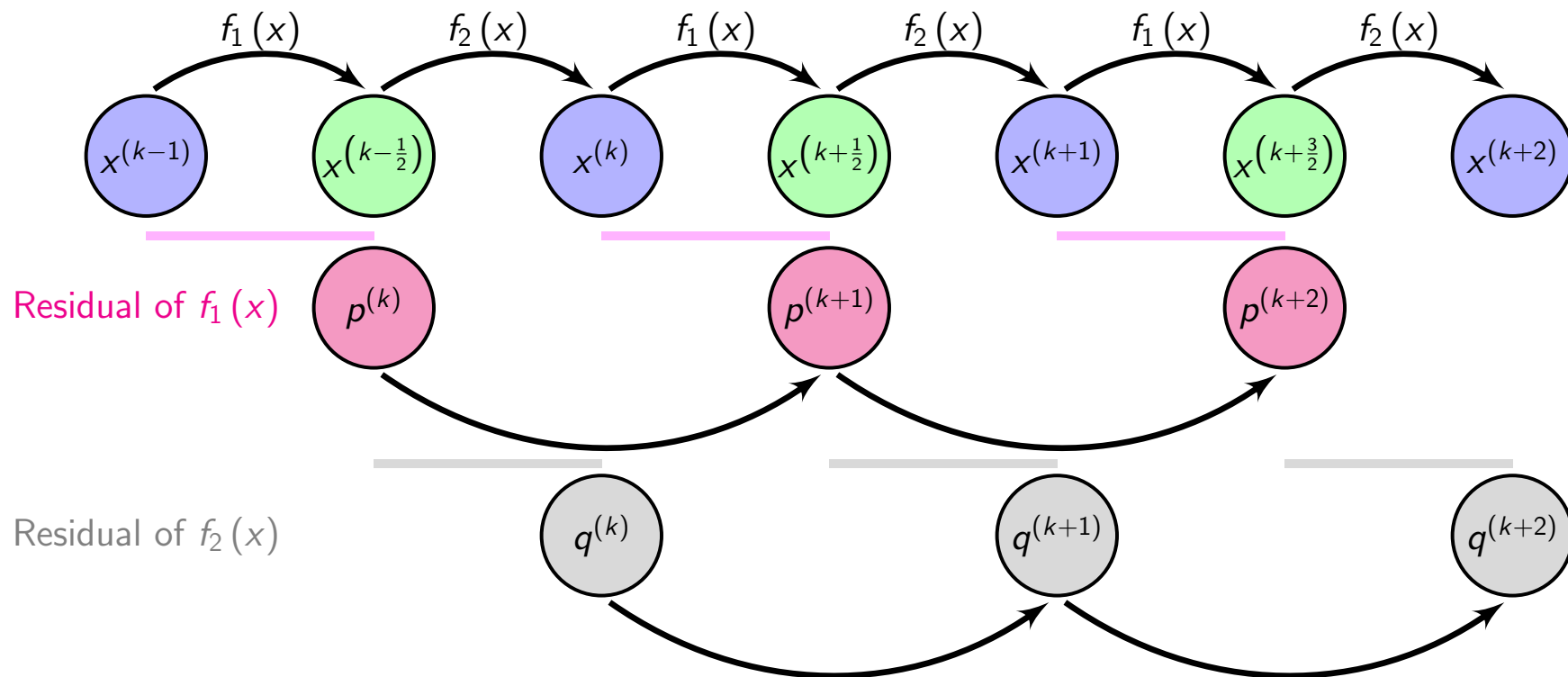


Figure 4: Splitting method of the Dykstra's algorithm

# Dykstra's algorithm

The case  $m > 2$

The case  $m > 2$  is a generalization of the previous algorithm by considering  $m$  residuals:

- 1 The  $x$ -update is:

$$x^{(k+1)} = \mathbf{prox}_{f_{j(k)}} \left( x^{(k)} + z^{(k+1-m)} \right)$$

- 2 The  $z$ -update is:

$$z^{(k+1)} = x^{(k)} + z^{(k+1-m)} - x^{(k+1)}$$

where  $x^{(0)} = v$ ,  $z^{(k)} = \mathbf{0}_n$  for  $k < 0$  and  $j(k) = \text{mod}(k+1, m)$  denotes the modulo operator taking values in  $\{1, \dots, m\}$

## Remark

The variable  $x^{(k)}$  is updated at each iteration while the residual  $z^{(k)}$  is updated every  $m$  iterations. This implies that the basic function  $f_j(x)$  is related to the residuals  $z^{(j)}$ ,  $z^{(j+m)}$ ,  $z^{(j+2m)}$ , etc.

# Dykstra's algorithm

The case  $m > 2$

Tibshirani (2017) proposes to write the Dykstra's algorithm by using two iteration indices  $k$  and  $j$ . The main index  $k$  refers to the cycle, whereas the sub-index  $j$  refers to the constraint number

The Dykstra's algorithm becomes:

- 1 The  $x$ -update is:

$$x^{(k+1,j)} = \mathbf{prox}_{f_j} \left( x^{(k+1,j-1)} + z^{(k,j)} \right)$$

- 2 The  $z$ -update is:

$$z^{(k+1,j)} = x^{(k+1,j-1)} + z^{(k,j)} - x^{(k+1,j)}$$

where  $x^{(1,0)} = v$ ,  $z^{(k,j)} = \mathbf{0}_n$  for  $k = 0$  and  $x^{(k+1,0)} = x^{(k,m)}$

# Dykstra's algorithm

The case  $m > 2$

The Dykstra's algorithm is particularly efficient when we consider the projection problem:

$$x^* = \mathcal{P}_\Omega (v)$$

where:

$$\Omega = \Omega_1 \cap \Omega_2 \cap \dots \cap \Omega_m$$

Indeed, the Dykstra's algorithm becomes:

- 1 The  $x$ -update is:

$$x^{(k+1,j)} = \mathbf{prox}_{f_j} \left( x^{(k+1,j-1)} + z^{(k,j)} \right) = \mathcal{P}_{\Omega_j} \left( x^{(k+1,j-1)} + z^{(k,j)} \right)$$

- 2 The  $z$ -update is:

$$z^{(k+1,j)} = x^{(k+1,j-1)} + z^{(k,j)} - x^{(k+1,j)}$$

where  $x^{(1,0)} = v$ ,  $z^{(k,j)} = \mathbf{0}_n$  for  $k = 0$  and  $x^{(k+1,0)} = x^{(k,m)}$

# Dijkstra's algorithm

Successive projections of  $\mathcal{P}_{\Omega_j} (x^{(k+1,j-1)})$  do not work!

Successive projections of  $\mathcal{P}_{\Omega_j} (x^{(k+1,j-1)} + z^{(k,j)})$  do work!



# Dykstra's algorithm

**Table 6:** Solving the proximal problem with linear inequality constraints

The goal is to compute the solution  $x^* = \mathbf{prox}_f(v)$  where  $f(x) = \mathbb{1}_\Omega(x)$  and  $\Omega = \{x \in \mathbb{R}^n : Cx \leq D\}$

We initialize  $x^{(0,m)} \leftarrow v$

We set  $z^{(0,1)} \leftarrow \mathbf{0}_n, \dots, z^{(0,m)} \leftarrow \mathbf{0}_n$

$k \leftarrow 0$

**repeat**

$x^{(k+1,0)} \leftarrow x^{(k,m)}$

**for**  $j = 1 : m$  **do**

The x-update is:

$$x^{(k+1,j)} = x^{(k+1,j-1)} + z^{(k,j)} - \frac{\left( c_{(j)}^\top x^{(k+1,j-1)} + c_{(j)}^\top z^{(k,j)} - d_{(j)} \right)_+ c_{(j)}}{\|c_{(j)}\|_2^2}$$

The z-update is:

$$z^{(k+1,j)} = x^{(k+1,j-1)} + z^{(k,j)} - x^{(k+1,j)}$$

**end for**

$k \leftarrow k + 1$

**until** Convergence

**return**  $x^* \leftarrow x^{(k,m)}$

# Dykstra's algorithm

**Table 7:** Solving the proximal problem with general linear constraints

The goal is to compute the solution  $x^* = \mathbf{prox}_f(v)$  where  $f(x) = \mathbf{1}_\Omega(x)$ ,  $\Omega = \Omega_1 \cap \Omega_2 \cap \Omega_3$ ,  $\Omega_1 = \{x \in \mathbb{R}^n : Ax = B\}$ ,  $\Omega_2 = \{x \in \mathbb{R}^n : Cx \leq D\}$  and  $\Omega_3 = \{x \in \mathbb{R}^n : x^- \leq x \leq x^+\}$

We initialize  $x_m^{(0)} \leftarrow v$

We set  $z_1^{(0)} \leftarrow \mathbf{0}_n$ ,  $z_2^{(0)} \leftarrow \mathbf{0}_n$  and  $z_3^{(0)} \leftarrow \mathbf{0}_n$

$k \leftarrow 0$

**repeat**

$$x_0^{(k+1)} \leftarrow x_m^{(k)}$$

$$x_1^{(k+1)} \leftarrow x_0^{(k+1)} + z_1^{(k)} - A^\dagger \left( Ax_0^{(k+1)} + Az_1^{(k)} - B \right)$$

$$z_1^{(k+1)} \leftarrow x_0^{(k+1)} + z_1^{(k)} - x_1^{(k+1)}$$

$$x_2^{(k+1)} \leftarrow \mathcal{P}_{\Omega_2} \left( x_1^{(k+1)} + z_2^{(k)} \right)$$

$$z_2^{(k+1)} \leftarrow x_1^{(k+1)} + z_2^{(k)} - x_2^{(k+1)}$$

$$x_3^{(k+1)} \leftarrow \mathcal{T} \left( x_2^{(k+1)} + z_3^{(k)}; x^-, x^+ \right)$$

$$z_3^{(k+1)} \leftarrow x_2^{(k+1)} + z_3^{(k)} - x_3^{(k+1)}$$

$$k \leftarrow k + 1$$

**until** Convergence

**return**  $x^* \leftarrow x_3^{(k)}$

► Previous algorithm

# Dijkstra's algorithm

## Remark

Since we have:

$$\frac{1}{2} \|x - v\|_2^2 = \frac{1}{2} x^\top x - x^\top v + \frac{1}{2} v^\top v$$

the two previous problems can be cast into a QP problem:

$$\begin{aligned} x^* &= \arg \min_x \frac{1}{2} x^\top I_n x - x^\top v \\ \text{s.t. } & x \in \Omega \end{aligned}$$

# Dijkstra's algorithm

## Dijkstra's algorithm versus QP algorithm

- The vector  $v$  is defined by the elements  $v_i = \ln(1 + i^2)$
- The set of constraints is:

$$\Omega = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n x_i \leq \frac{1}{2}, \sum_{i=1}^n e^{-i} x_i \geq 0 \right\}$$

- Using a Matlab implementation, we find that the computational time of the Dijkstra's algorithm when  $n$  is equal to 10 million is equal to the QP algorithm when  $n$  is equal to 12 500!
- The QP algorithm requires to store the matrix  $I_n$  — impossible when  $n > 10^5$ . For instance, the size of  $I_n$  is equal to 7450.6 GB when  $n = 10^6$

# Application to portfolio allocation

**Table 8:** Some objective functions used in portfolio optimization

Item	Portfolio	$f(x)$	Reference
(1)	MVO	$\frac{1}{2}x^\top \Sigma x - \gamma x^\top \mu$	Markowitz (1952)
(2)	GMV	$\frac{1}{2}x^\top \Sigma x$	Jagganathan and Ma (2003)
(3)	MDP	$\ln \left( \sqrt{x^\top \Sigma x} \right) - \ln (x^\top \sigma)$	Choueifaty and Coignard (2008)
(4)	KL	$\sum_{i=1}^n x_i \ln (x_i / \tilde{x}_i)$	Bera and Park (2008)
(5)	ERC	$\frac{1}{2}x^\top \Sigma x - \lambda \sum_{i=1}^n \ln x_i$	Maillard <i>et al.</i> (2010)
(6)	RB	$\mathcal{R}(x) - \lambda \sum_{i=1}^n \mathcal{R}\mathcal{B}_i \cdot \ln x_i$	Roncalli (2015)
(7)	RQE	$\frac{1}{2}x^\top D x$	Carmichael <i>et al.</i> (2018)

# Application to portfolio allocation

**Table 9:** Some regularization penalties used in portfolio optimization

Item	Regularization	$\mathfrak{R}(x)$	Reference
(8)	Ridge	$\lambda \ x - \tilde{x}\ _2^2$	DeMiguel <i>et al.</i> (2009)
(9)	Lasso	$\lambda \ x - \tilde{x}\ _1$	Brodie <i>et al.</i> (2009)
(10)	Log-barrier	$-\sum_{i=1}^n \lambda_i \ln x_i$	Roncalli (2013)
(11)	Shannon's entropy	$\lambda \sum_{i=1}^n x_i \ln x_i$	Yu <i>et al.</i> (2014)

# Application to portfolio allocation

Table 10: Some constraints used in portfolio optimization

Item	Constraint	$\Omega$
(12)	No cash and leverage	$\sum_{i=1}^n x_i = 1$
(13)	No short selling	$x_i \geq 0$
(14)	Weight bounds	$x_i^- \leq x_i \leq x_i^+$
(15)	Asset class limits	$c_j^- \leq \sum_{i \in \mathcal{C}_j} x_i \leq c_j^+$
(16)	Turnover	$\sum_{i=1}^n  x_i - \tilde{x}_i  \leq \tau^+$
(17)	Transaction costs	$\sum_{i=1}^n (c_i^- (\tilde{x}_i - x_i)_+ + c_i^+ (x_i - \tilde{x}_i)_+) \leq \mathbf{c}^+$
(18)	Leverage limit	$\sum_{i=1}^n  x_i  \leq \mathcal{L}^+$
(19)	Long/short exposure	$-\mathcal{L}\mathcal{S}^- \leq \sum_{i=1}^n x_i \leq \mathcal{L}\mathcal{S}^+$
(20)	Benchmarking	$\sqrt{(x - \tilde{x})^\top \Sigma (x - \tilde{x})} \leq \sigma^+$
(21)	Tracking error floor	$\sqrt{(x - \tilde{x})^\top \Sigma (x - \tilde{x})} \geq \sigma^-$
(22)	Active share floor	$\frac{1}{2} \sum_{i=1}^n  x_i - \tilde{x}_i  \geq \mathcal{A}\mathcal{S}^-$
(23)	Number of active bets	$(x^\top x)^{-1} \geq \mathcal{N}^-$

# Application to portfolio allocation

Most of portfolio optimization problems are a combination of:

- 1 an objective function (Table 8)
- 2 one or two regularization penalty functions (Table 9)
- 3 some constraints (Table 10)

Perrin and Roncalli (2020) solve **all these problems** using CCD, ADMM, Dykstra and the appropriate proximal functions. For that, they derive:

- the semi-analytical solution of the  $x$ -step for all objective functions
- the proximal solution of the  $y$ -step for all regularization penalty functions and constraints



# Herfindahl-MV optimization

## Formulation of the mathematical problem

- The second generation of minimum variance strategies uses a global diversification constraint
- The most popular solution is based on the Herfindahl index:

$$\mathcal{H}(x) = \sum_{i=1}^n x_i^2$$

- The effective number of bets is the inverse of the Herfindahl index:

$$\mathcal{N}(x) = \mathcal{H}(x)^{-1}$$

- The optimization program is:

$$x^* = \arg \min_x \frac{1}{2} x^\top \Sigma x$$
$$\text{s.t.} \quad \begin{cases} \mathbf{1}_n^\top x = 1 \\ \mathbf{0}_n \leq x \leq x^+ \\ \mathcal{N}(x) \geq \mathcal{N}^- \end{cases}$$

where  $\mathcal{N}^-$  is the minimum number of effective bets.

# Herfindahl-MV optimization

## The QP solution

- The Herfindhal constraint is equivalent to:

$$\begin{aligned} \mathcal{N}(x) \geq \mathcal{N}^- &\Leftrightarrow (x^\top x)^{-1} \geq \mathcal{N}^- \\ &\Leftrightarrow x^\top x \leq \frac{1}{\mathcal{N}^-} \end{aligned}$$

- The QP problem is:

$$\begin{aligned} x^*(\lambda) &= \arg \min_x \frac{1}{2} x^\top \Sigma x + \lambda x^\top x = \frac{1}{2} x^\top (\Sigma + 2\lambda I_n) x \\ \text{s.t. } &\begin{cases} \mathbf{1}_n^\top x = 1 \\ \mathbf{0}_n \leq x \leq x^+ \end{cases} \end{aligned}$$

where  $\lambda \geq 0$  is a scalar

- We have  $\mathcal{N}(x) \in [\mathcal{N}(x^*(0)), n]$
- The optimal value  $\lambda^*$  is found using the bi-section algorithm such that  $\mathcal{N}(x^*(\lambda)) = \mathcal{N}^-$

# Herfindahl-MV optimization

## The ADMM solution (first version)

- The ADMM form is:

$$\{x^*, y^*\} = \arg \min_{(x,y)} \frac{1}{2} x^\top \Sigma x + \mathbb{1}_{\Omega_1}(x) + \mathbb{1}_{\Omega_2}(y)$$

s.t.  $x = y$

where  $\Omega_1 = \{x \in \mathbb{R}^n : \mathbf{1}_n^\top x = 1, \mathbf{0}_n \leq x \leq x^+\}$  and

$$\Omega_2 = \mathcal{B}_2 \left( \mathbf{0}_n, \sqrt{\frac{1}{\mathcal{N}^-}} \right)$$

- The  $x$ -update is a QP problem:

$$x^{(k+1)} = \arg \min_x \left\{ \frac{1}{2} x^\top (\Sigma + \varphi I_n) x - \varphi x^\top (y^{(k)} - u^{(k)}) + \mathbb{1}_{\Omega_1}(x) \right\}$$

- The  $y$ -update is:

$$y^{(k+1)} = \frac{x^{(k+1)} + u^{(k)}}{\max \left( 1, \sqrt{\mathcal{N}^-} \|x^{(k+1)} + u^{(k)}\|_2 \right)}$$

# Herfindahl-MV optimization

## The ADMM solution (second version)

- A better approach is to write the problem as follows:

$$\{x^*, y^*\} = \arg \min_{(x,y)} \frac{1}{2} x^\top \Sigma x + \mathbb{1}_{\Omega_3}(x) + \mathbb{1}_{\Omega_4}(y)$$

s.t.  $x = y$

where  $\Omega_3 = \mathcal{H}_{\text{hyperplane}}[\mathbf{1}_n, 1]$  and  $\Omega_4 = \mathcal{B}_{\text{ox}}[\mathbf{0}_n, x^+] \cap \mathcal{B}_2\left(\mathbf{0}_n, \sqrt{\frac{1}{\mathcal{N}^-}}\right)$

- The  $x$ -update is:

$$x^{(k+1)} = (\Sigma + \varphi I_n)^{-1} \left( \varphi \left( y^{(k)} - u^{(k)} \right) + \frac{\mathbf{1}_n^\top (\Sigma + \varphi I_n)^{-1} \varphi \left( y^{(k)} - u^{(k)} \right)}{\mathbf{1}_n^\top (\Sigma + \varphi I_n)^{-1} \mathbf{1}_n} \mathbf{1}_n \right)$$

- The  $y$ -update is:

$$y^{(k+1)} = \mathcal{P}_{\mathcal{B}_{\text{ox}}-\mathcal{B}_{\text{all}}} \left( x^{(k+1)} + u^{(k)}; \mathbf{0}_n, x^+, \mathbf{0}_n, \sqrt{\frac{1}{\mathcal{N}^-}} \right)$$

where  $\mathcal{P}_{\mathcal{B}_{\text{ox}}-\mathcal{B}_{\text{all}}}$  corresponds to the Dykstra's algorithm given by Perrin and Roncalli (2020)

# Herfindahl-MV optimization

## Remark

If we compare the computational time of the three approaches, we observe that the best method is the second version of the ADMM algorithm:

$$CT(QP; n = 1000) = 50 \times CT(ADMM_2; n = 1000)$$

$$CT(ADMM_1; n = 1000) = 400 \times CT(ADMM_2; n = 1000)$$

# Herfindahl-MV optimization

## The QP solution

### Example 5

We consider an investment universe of eight stocks. We assume that their volatilities are 21%, 20%, 40%, 18%, 35%, 23%, **7%** and 29%. The correlation matrix is defined as follows:

$$\rho = \begin{pmatrix} 100\% & & & & & & & & & \\ 80\% & 100\% & & & & & & & & \\ 70\% & 75\% & 100\% & & & & & & & \\ 60\% & 65\% & 90\% & 100\% & & & & & & \\ 70\% & 50\% & 70\% & 85\% & 100\% & & & & & \\ 50\% & 60\% & 70\% & 80\% & 60\% & 100\% & & & & \\ 70\% & 50\% & 70\% & 75\% & 80\% & 50\% & 100\% & & & \\ 60\% & 65\% & 70\% & 75\% & 65\% & 70\% & 80\% & 100\% & & \end{pmatrix}$$

# Herfindahl-MV optimization

Table 11: Minimum variance portfolios (in %)

$\mathcal{N}^-$	1.00	2.00	3.00	4.00	5.00	6.00	6.50	7.00	7.50	8.00
$x_1^*$	0.00	3.22	9.60	13.83	15.18	15.05	14.69	14.27	13.75	12.50
$x_2^*$	0.00	12.75	14.14	15.85	16.19	15.89	15.39	14.82	14.13	12.50
$x_3^*$	0.00	0.00	0.00	0.00	0.00	0.07	2.05	4.21	6.79	12.50
$x_4^*$	0.00	10.13	15.01	17.38	17.21	16.09	15.40	14.72	13.97	12.50
$x_5^*$	0.00	0.00	0.00	0.00	0.71	5.10	6.33	7.64	9.17	12.50
$x_6^*$	0.00	5.36	8.95	12.42	13.68	14.01	13.80	13.56	13.25	12.50
$x_7^*$	100.00	68.53	52.31	40.01	31.52	25.13	22.92	20.63	18.00	12.50
$x_8^*$	0.00	0.00	0.00	0.50	5.51	8.66	9.41	10.14	10.95	12.50
$\lambda^*$ (in %)	0.00	1.59	3.10	5.90	10.38	18.31	23.45	31.73	49.79	$\infty$

Note: the upper bound  $x^+$  is set to  $\mathbf{1}_n$ . The solutions are those found by the ADMM algorithm. We also report the value of  $\lambda^*$  found by the bi-section algorithm when we use the QP algorithm.

# ERC portfolio optimization

We recall that:

$$x^* = \arg \min_x \frac{1}{2} x^\top \Sigma x - \lambda \sum_{i=1}^n \ln x_i$$

and:

$$x_{\text{erc}} = \frac{x^*}{\mathbf{1}_n^\top x^*}$$



# ERC portfolio optimization

## The CCD solution

- The first-order condition  $(\Sigma x)_i - \lambda x_i^{-1} = 0$  implies that:

$$x_i^2 \sigma_i^2 + x_i \sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j - \lambda = 0$$

- The CCD algorithm is:

$$x_i^{(k+1)} = \frac{-v_i^{(k+1)} + \sqrt{\left(v_i^{(k+1)}\right)^2 + 4\lambda\sigma_i^2}}{2\sigma_i^2}$$

where:

$$v_i^{(k+1)} = \sigma_i \sum_{j < i} x_j^{(k+1)} \rho_{i,j} \sigma_j + \sigma_i \sum_{j > i} x_j^{(k)} \rho_{i,j} \sigma_j$$

# ERC portfolio optimization

## The ADMM solution

- In the case of the ADMM algorithm, we set:

$$\begin{aligned}f_x(x) &= \frac{1}{2}x^\top \Sigma x \\f_y(y) &= -\lambda \sum_{i=1}^n \ln y_i \\x &= y\end{aligned}$$

- The  $x$ -update step is:

$$x^{(k+1)} = (\Sigma + \varphi I_n)^{-1} \varphi \left( y^{(k)} - u^{(k)} \right)$$

- The  $y$ -update step is:

$$y_i^{(k+1)} = \frac{1}{2} \left( \left( x_i^{(k+1)} + u_i^{(k)} \right) + \sqrt{\left( x_i^{(k+1)} + u_i^{(k)} \right)^2 + 4\lambda\varphi^{-1}} \right)$$

# RB portfolio optimization

The RB portfolio is equal to:

$$x_{\text{rb}} = \frac{x^*}{\mathbf{1}_n^\top x^*}$$

where  $x^*$  is the solution of the logarithmic barrier problem:

$$x^* = \arg \min_x \mathcal{R}(x) - \lambda \sum_{i=1}^n \mathcal{RB}_i \cdot \ln x_i$$

$\lambda$  is any positive scalar and  $\mathcal{RB}_i$  is the risk budget allocated to Asset  $i$

# RB portfolio optimization

## The CCD solution (SD risk measure)

- In the case of the standard deviation-based risk measure:

$$\mathcal{R}(x) = -x^\top (\mu - r) + \xi \sqrt{x^\top \Sigma x}$$

the first-order condition for defining the CCD algorithm is:

$$-(\mu_i - r) + \xi \frac{(\Sigma x)_i}{\sqrt{x^\top \Sigma x}} - \lambda \frac{\mathcal{RB}_i}{x_i} = 0$$

- It follows that  $\xi x_i (\Sigma x)_i - (\mu_i - r) x_i \sigma(x) - \lambda \sigma(x) \cdot \mathcal{RB}_i = 0$  or equivalently:

$$\alpha_i x_i^2 + \beta_i x_i + \gamma_i = 0$$

where  $\alpha_i = \xi \sigma_i^2$ ,  $\beta_i = \xi \sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j - (\mu_i - r) \sigma(x)$  and  $\gamma_i = -\lambda \sigma(x) \cdot \mathcal{RB}_i$

# RB portfolio optimization

## The CCD solution (SD risk measure)

- The CCD algorithm is:

$$x_i^{(k+1)} = \frac{-\beta_i^{(k+1)} + \sqrt{\left(\beta_i^{(k+1)}\right)^2 - 4\alpha_i^{(k+1)}\gamma_i^{(k+1)}}}{2\alpha_i^{(k+1)}}$$

where:

$$\left\{ \begin{array}{l} \alpha_i^{(k+1)} = \xi\sigma_i^2 \\ \beta_i^{(k+1)} = \xi\sigma_i \left( \sum_{j<i} x_j^{(k+1)} \rho_{i,j}\sigma_j + \sum_{j>i} x_j^{(k)} \rho_{i,j}\sigma_j \right) - (\mu_i - r) \sigma_i^{(k+1)}(x) \\ \gamma_i^{(k+1)} = -\lambda \sigma_i^{(k+1)}(x) \cdot \mathcal{RB}_i \\ \sigma_i^{(k+1)}(x) = \sqrt{\chi^\top \Sigma \chi} \\ \chi = \left( x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k)}, x_{i+1}^{(k)}, \dots, x_n^{(k)} \right) \end{array} \right.$$

# RB portfolio optimization

## The ADMM solution (convex risk measure)

- We have:

$$\begin{aligned} \{x^*, y^*\} &= \arg \min_{x, y} \mathcal{R}(x) - \lambda \sum_{i=1}^n \mathcal{R}\mathcal{B}_i \cdot \ln y_i \\ \text{s.t. } &x = y \end{aligned}$$

- The ADMM algorithm is:

$$\begin{cases} x^{(k+1)} = \mathbf{prox}_{\varphi^{-1}\mathcal{R}(x)}(y^{(k)} - u^{(k)}) \\ v_y^{(k+1)} = x^{(k+1)} + u^{(k)} \\ y^{(k+1)} = \frac{1}{2} \left( v_y^{(k+1)} + \sqrt{v_y^{(k+1)} \odot v_y^{(k+1)} + 4\lambda\varphi^{-1} \cdot \mathcal{R}\mathcal{B}} \right) \\ u^{(k+1)} = u^{(k)} + x^{(k+1)} - y^{(k+1)} \end{cases}$$

# Tips and tricks of portfolio optimization

- Full allocation —  $\sum_{i=1}^n x_i = 1$ :

$$\Omega = \mathcal{H}_{\text{hyperplane}} [\mathbf{1}_n, 1]$$

We have:

$$\mathcal{P}_{\Omega}(v) = v - \left( \frac{\mathbf{1}_n^{\top} v - 1}{n} \right) \mathbf{1}_n$$

- Cash neutral —  $\sum_{i=1}^n x_i = 0$ :

$$\Omega = \mathcal{H}_{\text{hyperplane}} [\mathbf{1}_n, 0]$$

We have:

$$\mathcal{P}_{\Omega}(v) = v - \left( \frac{\mathbf{1}_n^{\top} v}{n} \right) \mathbf{1}_n$$

# Tips and tricks of portfolio optimization

- No short selling —  $x \geq \mathbf{0}_n$ :

$$\Omega = \mathcal{B}_{\text{ox}} [\mathbf{0}_n, \infty]$$

We have:

$$\mathcal{P}_{\Omega}(v) = \mathcal{T}(v; \mathbf{0}_n, \infty)$$

- Weight bounds —  $x^- \leq x \leq x^+$ :

$$\Omega = \mathcal{B}_{\text{ox}} [x^-, x^+]$$

We have:

$$\mathcal{P}_{\Omega}(v) = \mathcal{T}(v; x^-, x^+)$$



# Tips and tricks of portfolio optimization

- $\mu$ -problem —  $\mu(x) \geq \mu^*$ :

$$\Omega = \mathcal{H}_{\text{halfspace}}[-\mu, -\mu^*]$$

We have:

$$\mathcal{P}_{\Omega}(v) = v + \frac{(\mu^* - \mu^{\top} v)_+}{\|\mu\|_2^2} \mu$$

# Tips and tricks of portfolio optimization

- $\sigma$ -problem —  $\sigma(x) \leq \sigma^*$ :

$$\Omega = \left\{ x : \sqrt{x^\top \Sigma x} \leq \sigma^* \right\}$$

We have:

$$\begin{aligned} \sqrt{x^\top \Sigma x} \leq \sigma^* &\Leftrightarrow \sqrt{x^\top (LL^\top) x} \leq \sigma^* \\ &\Leftrightarrow \|y^\top y\|_2 \leq \sigma^* \\ &\Leftrightarrow y \in \mathcal{B}_2(\mathbf{0}_n, \sigma^*) \end{aligned}$$

where  $y = L^\top x$  and  $L$  is the Cholesky decomposition of  $\Sigma$ . It follows that the proximal of the  $y$ -update is the projection onto the  $\ell_2$  ball  $\mathcal{B}_2(\mathbf{0}_n, \sigma^*)$ :

$$\begin{aligned} \mathcal{P}_\Omega(v) &= v - \mathbf{prox}_{\sigma^* \|x\|_2}(v) \\ &= v - \left( 1 - \frac{\sigma^*}{\max(\sigma^*, \|v\|_2)} \right) v \end{aligned}$$

# Tips and tricks of portfolio optimization

- Leverage management —  $\sum_{i=1}^n |x_i| \leq \mathcal{L}^+$ :

$$\begin{aligned}\Omega &= \{x : \|x\|_1 \leq \mathcal{L}^+\} \\ &= \mathcal{B}_1(\mathbf{0}_n, \mathcal{L}^+)\end{aligned}$$

The proximal of the  $y$ -update is the projection onto the  $\ell_1$  ball  $\mathcal{B}_1(\mathbf{0}_n, \mathcal{L}^+)$ :

$$\mathcal{P}_\Omega(v) = v - \text{sign}(v) \odot \mathbf{prox}_{\mathcal{L}^+ \max_x}(|v|)$$

# Tips and tricks of portfolio optimization

- Leverage management —  $\mathcal{L}\mathcal{S}^- \leq \sum_{i=1}^n x_i \leq \mathcal{L}\mathcal{S}^+$ :

$$\Omega = \mathcal{H}_{\text{alfspace}} [\mathbf{1}_n, \mathcal{L}\mathcal{S}^+] \cap \mathcal{H}_{\text{alfspace}} [-\mathbf{1}_n, -\mathcal{L}\mathcal{S}^-]$$

The proximal of the  $y$ -update is obtained with the Dykstra's algorithm by combining the two half-space projections.

- Leverage management —  $|\sum_{i=1}^n x_i| \leq \mathcal{L}^+$ :

$$\Omega = \{x : |\mathbf{1}_n^\top x| \leq \mathcal{L}^+\}$$

This is a special case of the previous result where  $\mathcal{L}\mathcal{S}^+ = \mathcal{L}^+$  and  $\mathcal{L}\mathcal{S}^- = -\mathcal{L}^+$ :

$$\Omega = \mathcal{H}_{\text{alfspace}} [\mathbf{1}_n, \mathcal{L}^+] \cap \mathcal{H}_{\text{alfspace}} [-\mathbf{1}_n, \mathcal{L}^+]$$

# Tips and tricks of portfolio optimization

- Concentration management<sup>3</sup>

Portfolio managers can also use another constraint concerning the sum of the  $k$  largest values:

$$f(x) = \sum_{i=n-k+1}^n x_{(i:n)} = x_{(n:n)} + \dots + x_{(n-k+1:n)}$$

where  $x_{(i:n)}$  is the order statistics of  $x$ :  $x_{(1:n)} \leq x_{(2:n)} \leq \dots \leq x_{(n:n)}$ .  
 Beck (2017) shows that:

$$\mathbf{prox}_{\lambda f(x)}(v) = v - \lambda \mathcal{P}_{\Omega} \left( \frac{v}{\lambda} \right)$$

where:

$$\Omega = \{x \in [0, 1]^n : \mathbf{1}_n^\top x = k\} = \mathcal{B}_{\text{ox}}[\mathbf{0}_n, \mathbf{1}_n] \cap \mathcal{H}_{\text{yperlane}}[\mathbf{1}_n, k]$$

---

<sup>3</sup>An example is the 5/10/40 UCITS rule: A UCITS fund may invest no more than 10% of its net assets in transferable securities or money market instruments issued by the same body, with a further aggregate limitation of 40% of net assets on exposures of greater than 5% to single issuers.

# Tips and tricks of portfolio optimization

- Entropy portfolio management  
Bera and Park (2008) propose using a cross-entropy measure as the objective function:

$$\begin{aligned} x^* &= \arg \min_x \text{KL}(x \mid \tilde{x}) \\ \text{s.t.} & \begin{cases} \mathbf{1}_n^\top x = 1 \\ \mathbf{0}_n \leq x \leq \mathbf{1}_n \\ \mu(x) \geq \mu^*, \sigma(x) \leq \sigma^* \end{cases} \end{aligned}$$

where  $\text{KL}(x \mid \tilde{x})$  is the Kullback-Leibler measure:

$$\text{KL}(x \mid \tilde{x}) = \sum_{i=1}^n x_i \ln(x_i / \tilde{x}_i)$$

and  $\tilde{x}$  is a reference portfolio

# Tips and tricks of portfolio optimization

- Entropy portfolio management  
We have:

$$\mathbf{prox}_{\lambda \text{KL}(v|\tilde{x})}(v) = \lambda \begin{pmatrix} W\left(\lambda^{-1}\tilde{x}_1 e^{\lambda^{-1}v_1 - \tilde{x}_1^{-1}}\right) \\ \vdots \\ W\left(\lambda^{-1}\tilde{x}_n e^{\lambda^{-1}v_n - \tilde{x}_n^{-1}}\right) \end{pmatrix}$$

where  $W(x)$  is the Lambert  $W$  function

# Tips and tricks of portfolio optimization

## Remark

Since the Shannon's entropy is equal to  $SE(x) = -\text{KL}(x \mid \mathbf{1}_n)$ , we deduce that:

$$\text{prox}_{\lambda SE(x)}(v) = \lambda \begin{pmatrix} W(\lambda^{-1} e^{\lambda^{-1} v_1 - 1}) \\ \vdots \\ W(\lambda^{-1} e^{\lambda^{-1} v_n - 1}) \end{pmatrix}$$



# Tips and tricks of portfolio optimization

- Active share constraint —  $\mathcal{AS}(x | \tilde{x}) \geq \mathcal{AS}^-$ :

$$\mathcal{AS}(x | \tilde{x}) = \frac{1}{2} \sum_{i=1}^n |x_i - \tilde{x}_i| \geq \mathcal{AS}^-$$

We use the projection onto the complement  $\bar{\mathcal{B}}_1(c, r)$  of the  $\ell_1$  ball and we obtain:

$$\mathcal{P}_\Omega(v) = v + \text{sign}(v - \tilde{x}) \odot \frac{\max(2\mathcal{AS}^- - \|v - \tilde{x}\|_1, 0)}{n}$$

# Tips and tricks of portfolio optimization

- Tracking error volatility —  $\sigma(x | \tilde{x}) \leq \sigma^*$ :

$$\begin{aligned} \sigma(x | \tilde{x}) \leq \sigma^* &\Leftrightarrow \sqrt{(x - \tilde{x})^\top \Sigma (x - \tilde{x})} \leq \sigma^* \\ &\Leftrightarrow \|y\|_2 \leq \sigma^* \\ &\Leftrightarrow y \in \mathcal{B}_2(\mathbf{0}_n, \sigma^*) \end{aligned}$$

where  $y = L^\top x - L^\top \tilde{x}$ . It follows that  $Ax + By = c$  where  $A = L^\top$ ,  $B = -I_n$  and  $c = L^\top \tilde{x}$ . It follows that the proximal of the  $y$ -update is the projection onto the  $\ell_2$  ball  $\mathcal{B}_2(\mathbf{0}_n, \sigma^*)$ :

$$\begin{aligned} \mathcal{P}_\Omega(v) &= v - \mathbf{prox}_{\sigma^* \|x\|_2}(v) \\ &= v - \left( 1 - \frac{\sigma^*}{\max(\sigma^*, \|v\|_2)} \right) v \end{aligned}$$

# Tips and tricks of portfolio optimization

- Bid-ask transaction cost management:

$$\mathbf{c}(x | x_0) = \lambda \sum_{i=1}^n (c_i^- (x_{0,i} - x_i)_+ + c_i^+ (x_i - x_{0,i})_+)$$

where  $c_i^-$  and  $c_i^+$  are the bid and ask transaction costs. We have:

$$\mathbf{prox}_{\mathbf{c}(x|x_0)}(v) = x_0 + \mathcal{S}(v - x_0; \lambda c^-, \lambda c^+)$$

where  $\mathcal{S}(v; \lambda_-, \lambda_+) = (v - \lambda_+)_+ - (v + \lambda_-)_-$  is the two-sided soft-thresholding operator.

# Tips and tricks of portfolio optimization

- Turnover management:

$$\Omega = \{x \in \mathbb{R}^n : \|x - x_0\|_1 \leq \tau^+\}$$

The proximal operator is:

$$\mathcal{P}_\Omega(v) = v - \text{sign}(v - x_0) \odot \min(|v - x_0|, s^*)$$

where  $s^* = \{s \in \mathbb{R} : \sum_{i=1}^n (|v_i - x_{0,i}| - s)_+ = \tau^+\}$ .

# Pattern learning and self-automated strategies

Table 12: What works / What doesn't

	Bond Scoring	Stock Picking	Trend Filtering	Mean Reverting	Index Tracking	HF Tracking	Stock Classification	Technical Analysis
Lasso		😊	😊	😊	😞	😊		
NMF							😊	😞
Boosting		😊				😊		
Bagging		😊				😊		
Random forests	😊			😞				😞
Neural nets	😊					😞		
SVM	😊	😞	😞				😞	
Sparse Kalman					😞	😊		
K-NN	😞							
K-means	😊						😊	
Testing protocols <sup>4</sup>	😊	😊	😊	😊		😊		

Source: Roncalli (2014), Big Data in Asset Management, ESMA/CEMA/GEA meeting, Madrid.

<sup>4</sup>Cross-validation, training/test/probe sets, K-fold, etc.

# Pattern learning and self-automated strategies

2021  $\neq$  2014

**The evolution of machine learning in finance is fast, very fast!**

# Pattern learning and self-automated strategies

## Some examples

- Natural Language Processing (NLP)
- Deep learning (DL)
- Reinforcement learning (RL)
- Gaussian process (GP) and Bayesian optimization (BO)
- Learning to rank (MLR)
- Etc.

## Some applications

- Robo-advisory
- Stock classification
- $Q_1 - Q_5$  long/short strategy
- Trend-following strategies
- Mean-reverting strategies
- Scoring models
- Sentiment and news analysis
- Etc.

# Market generators

- The underlying idea is to simulate artificial multi-dimensional financial time series, whose statistical properties are the same as those observed in the financial markets
  - ≈ **Monte Carlo simulation of the financial market**
- 3 main approaches:
  - ① Restricted Boltzmann machines (RBM)
  - ② Generative adversarial networks (GAN)
  - ③ Convolutional Wasserstein models (W-GAN)
- The goal is to:
  - improve the the risk management of quantitative investment strategies
  - avoid the over-fitting bias of backtesting

**The current research shows that results are disappointed until now**



# Portfolio optimization with CCD and ADMM algorithms

## Question 1

We consider the following optimization program:

$$x^* = \arg \min \frac{1}{2} x^\top \Sigma x - \lambda \sum_{i=1}^n b_i \ln x_i$$

where  $\Sigma$  is the covariance matrix,  $b$  is a vector of positive budgets and  $x$  is the vector of portfolio weights.

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.a

Write the first-order condition with respect to the coordinate  $x_i$  and show that the solution  $x^*$  corresponds to a risk-budgeting portfolio.

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\mathcal{L}(x; \lambda) = \arg \min \frac{1}{2} x^\top \Sigma x - \lambda \sum_{i=1}^n b_i \ln x_i$$

The first-order condition is:

$$\frac{\partial \mathcal{L}(x; \lambda)}{\partial x_i} = (\Sigma x)_i - \lambda \frac{b_i}{x_i} = 0$$

or:

$$x_i \cdot (\Sigma x)_i = \lambda b_i$$

# Portfolio optimization with CCD and ADMM algorithms

If we assume that the risk measure is the portfolio volatility:

$$\mathcal{R}(x) = \sqrt{x^\top \Sigma x}$$

the risk contribution of Asset  $i$  is equal to:

$$\mathcal{RC}_i(x) = \frac{x_i \cdot (\Sigma x)_i}{\sqrt{x^\top \Sigma x}}$$

We deduce that the optimization problem defines a risk budgeting portfolio:

$$\frac{x_i \cdot (\Sigma x)_i}{b_i} = \frac{x_j \cdot (\Sigma x)_j}{b_j} = \lambda \Leftrightarrow \frac{\mathcal{RC}_i(x)}{b_i} = \frac{\mathcal{RC}_j(x)}{b_j}$$

where the risk measure is the portfolio volatility and the risk budgets are  $(b_1, \dots, b_n)$ .

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.b

Find the optimal value  $x_i^*$  when we consider the other coordinates  $(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$  as fixed.

# Portfolio optimization with CCD and ADMM algorithms

The first-order condition is equivalent to:

$$x_i \cdot (\Sigma x)_i - \lambda b_i = 0$$

We have:

$$(\Sigma x)_i = x_i \sigma_i^2 + \sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j$$

It follows that:

$$x_i^2 \sigma_i^2 + x_i \sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j - \lambda b_i = 0$$

# Portfolio optimization with CCD and ADMM algorithms

We obtain a second-degree equation:

$$\alpha_i x_i^2 + \beta_i x_i + \gamma_i = 0$$

where:

$$\begin{cases} \alpha_i = \sigma_i^2 \\ \beta_i = \sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j \\ \gamma_i = -\lambda b_i \end{cases}$$

- 1 The polynomial function is convex because we have  $\alpha_i = \sigma_i^2 > 0$
- 2 The product of the roots is negative:

$$x_i' x_i'' = \frac{\gamma_i}{\alpha_i} = -\frac{\lambda b_i}{\sigma_i^2} < 0$$

- 3 The discriminant is positive:

$$\Delta = \beta_i^2 - 4\alpha_i\gamma_i = \left( \sigma_i \sum_{j \neq i} \rho_{i,j} \sigma_j y_j \right)^2 + 4\lambda b_i \sigma_i^2 > 0$$

# Portfolio optimization with CCD and ADMM algorithms

We always have two solutions with opposite signs. We deduce that the solution is the positive root of the second-degree equation:

$$\begin{aligned}
 x_i^* &= x_i'' = \frac{-\beta_i + \sqrt{\beta_i^2 - 4\alpha_i\gamma_i}}{2\alpha_i} \\
 &= \frac{-\sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j + \sqrt{\sigma_i^2 \left( \sum_{j \neq i} x_j \rho_{i,j} \sigma_j \right)^2 + 4\lambda b_i \sigma_i^2}}{2\sigma_i^2}
 \end{aligned}$$



# Portfolio optimization with CCD and ADMM algorithms

## Question 1.c

We note  $x_i^{(k)}$  the value of the  $i^{\text{th}}$  coordinate at the  $k^{\text{th}}$  iteration. Deduce the corresponding CCD algorithm. How to find the RB portfolio  $x_{\text{rb}}$ ?

# Portfolio optimization with CCD and ADMM algorithms

The CCD algorithm consists in iterating the following formula:

$$x_i^{(k)} = \frac{-\beta_i^{(k)} + \sqrt{\left(\beta_i^{(k)}\right)^2 - 4\alpha_i^{(k)}\gamma_i^{(k)}}}{2\alpha_i^{(k)}}$$

where:

$$\begin{cases} \alpha_i^{(k)} = \sigma_i^2 \\ \beta_i^{(k)} = \sigma_i \left( \sum_{j < i} \rho_{i,j} \sigma_j x_j^{(k)} + \sum_{j > i} \rho_{i,j} \sigma_j x_j^{(k-1)} \right) \\ \gamma_i^{(k)} = -\lambda b_i \end{cases}$$

The RB portfolio is the scaled solution:

$$x_{rb} = \frac{x^*}{\sum_{i=1}^n x_i^*}$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d

We consider a universe of three assets, whose volatilities are equal to 20%, 25% and 30%. The correlation matrix is equal to:

$$\rho = \begin{pmatrix} 100\% & & \\ 50\% & 100\% & \\ 60\% & 70\% & 100\% \end{pmatrix}$$

We would like to compute the ERC portfolio<sup>a</sup> using the CCD algorithm. We initialize the CCD algorithm with the following starting values  $x^{(0)} = (33.3\%, 33.3\%, 33.3\%)$ . We assume that  $\lambda = 1$ .

---

<sup>a</sup>This means that:

$$b_i = \frac{1}{3}$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.i

Starting from  $x^{(0)}$ , find the optimal coordinate  $x_1^{(1)}$  for the first asset.

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\begin{cases} \alpha_1^{(1)} = 0.2^2 = 4\% \\ \beta_1^{(1)} = 0.02033 \\ \gamma_i^{(1)} = -0.333\% \end{cases}$$

We obtain:

$$x_1^{(1)} = 2.64375$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.ii

Compute then the optimal coordinate  $x_2^{(1)}$  for the second asset.

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\begin{cases} \alpha_2^{(1)} = 0.25^2 = 6.25\% \\ \beta_2^{(1)} = 0.08359 \\ \gamma_2^{(1)} = -0.333\% \end{cases}$$

We obtain:

$$x_2^{(1)} = 1.73553$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.iii

Compute then the optimal coordinate  $x_3^{(1)}$  for the third asset.



# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\begin{cases} \alpha_3^{(1)} = 0.3^2 = 9\% \\ \beta_3^{(1)} = 0.18629 \\ \gamma_3^{(1)} = -0.333\% \end{cases}$$

We obtain:

$$x_3^{(1)} = 1.15019$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.iv

Give the CCD coordinates  $x_i^{(k)}$  for  $k = 1, \dots, 10$ .

# Portfolio optimization with CCD and ADMM algorithms

Table 13: CCD coordinates ( $k = 1, \dots, 5$ )

$k$	$i$	$\alpha_i^{(k)}$	$\beta_i^{(k)}$	$\gamma_i^{(k)}$	$x_i^{(k)}$	CCD coordinates		
						$x_1$	$x_2$	$x_3$
0						0.33333	0.33333	0.33333
1	1	0.04000	0.02033	-0.33333	2.64375	2.64375	0.33333	0.33333
1	2	0.06250	0.08359	-0.33333	1.73553	2.64375	1.73553	0.33333
1	3	0.09000	0.18629	-0.33333	1.15019	2.64375	1.73553	1.15019
2	1	0.04000	0.08480	-0.33333	2.01525	2.01525	1.73553	1.15019
2	2	0.06250	0.11077	-0.33333	1.58744	2.01525	1.58744	1.15019
2	3	0.09000	0.15589	-0.33333	1.24434	2.01525	1.58744	1.24434
3	1	0.04000	0.08448	-0.33333	2.01782	2.01782	1.58744	1.24434
3	2	0.06250	0.11577	-0.33333	1.56202	2.01782	1.56202	1.24434
3	3	0.09000	0.15465	-0.33333	1.24842	2.01782	1.56202	1.24842
4	1	0.04000	0.08399	-0.33333	2.02183	2.02183	1.56202	1.24842
4	2	0.06250	0.11609	-0.33333	1.56044	2.02183	1.56044	1.24842
4	3	0.09000	0.15471	-0.33333	1.24821	2.02183	1.56044	1.24821
5	1	0.04000	0.08395	-0.33333	2.02222	2.02222	1.56044	1.24821
5	2	0.06250	0.11609	-0.33333	1.56044	2.02222	1.56044	1.24821
5	3	0.09000	0.15472	-0.33333	1.24817	2.02222	1.56044	1.24817

# Portfolio optimization with CCD and ADMM algorithms

Table 14: CCD coordinates ( $k = 6, \dots, 10$ )

$k$	$i$	$\alpha_i^{(k)}$	$\beta_i^{(k)}$	$\gamma_i^{(k)}$	$x_i^{(k)}$	CCD coordinates		
						$x_1$	$x_2$	$x_3$
0						0.33333	0.33333	0.33333
6	1	0.04000	0.08395	-0.33333	2.02223	2.02223	1.56044	1.24817
6	2	0.06250	0.11608	-0.33333	1.56045	2.02223	1.56045	1.24817
6	3	0.09000	0.15472	-0.33333	1.24816	2.02223	1.56045	1.24816
7	1	0.04000	0.08395	-0.33333	2.02223	2.02223	1.56045	1.24816
7	2	0.06250	0.11608	-0.33333	1.56046	2.02223	1.56046	1.24816
7	3	0.09000	0.15472	-0.33333	1.24816	2.02223	1.56046	1.24816
8	1	0.04000	0.08395	-0.33333	2.02223	2.02223	1.56046	1.24816
8	2	0.06250	0.11608	-0.33333	1.56046	2.02223	1.56046	1.24816
8	3	0.09000	0.15472	-0.33333	1.24816	2.02223	1.56046	1.24816
9	1	0.04000	0.08395	-0.33333	2.02223	2.02223	1.56046	1.24816
9	2	0.06250	0.11608	-0.33333	1.56046	2.02223	1.56046	1.24816
9	3	0.09000	0.15472	-0.33333	1.24816	2.02223	1.56046	1.24816
10	1	0.04000	0.08395	-0.33333	2.02223	2.02223	1.56046	1.24816
10	2	0.06250	0.11608	-0.33333	1.56046	2.02223	1.56046	1.24816
10	3	0.09000	0.15472	-0.33333	1.24816	2.02223	1.56046	1.24816

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.v

Deduce the ERC portfolio.

# Portfolio optimization with CCD and ADMM algorithms

The CCD algorithm has converged to the following solution:

$$x^* = \begin{pmatrix} 2.02223 \\ 1.56046 \\ 1.24816 \end{pmatrix}$$

Since  $\sum_{i=1}^3 x_i^* = 4.83085$ , we deduce that:

$$x_{\text{erc}} = \frac{1}{4.83085} \begin{pmatrix} 2.02223 \\ 1.56046 \\ 1.24816 \end{pmatrix} = \begin{pmatrix} 41.86076\% \\ 32.30189\% \\ 25.83736\% \end{pmatrix}$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.vi

Compute the variance of the previous CCD solution. What do you notice?  
Explain this result.

# Portfolio optimization with CCD and ADMM algorithms

We remind that the CCD solution is:

$$x^* = \begin{pmatrix} 2.02223 \\ 1.56046 \\ 1.24816 \end{pmatrix}$$

We have:

$$\sigma^2(x^*) = x^{*\top} \Sigma x^* = 1$$

We notice that:

$$\sigma^2(x^*) = \lambda$$



# Portfolio optimization with CCD and ADMM algorithms

At the optimum, we remind that:

$$\lambda = \frac{x_i^* \cdot (\Sigma x^*)_i}{b_i} = \frac{x_i^* \cdot (\Sigma x^*)_i}{n^{-1}}$$

We deduce that:

$$\begin{aligned} \lambda &= \frac{1}{n} \sum_{i=1}^n \frac{x_i^* \cdot (\Sigma x^*)_i}{n^{-1}} \\ &= \sum_{i=1}^n x_i^* \cdot (\Sigma x^*)_i \\ &= x^{*\top} \Sigma x^* \\ &= \sigma^2(x^*) \end{aligned}$$

It follows that the portfolio variance of the CCD solution is exactly equal to  $\lambda$ .

# Portfolio optimization with CCD and ADMM algorithms

## Question 1.d.vii

Verify that the CCD solution converges faster to the ERC portfolio when we assume that  $\lambda = x_{\text{erc}}^\top \Sigma x_{\text{erc}}$ .

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\sigma(x_{\text{erc}}) = \sqrt{x_{\text{erc}}^{\top} \Sigma x_{\text{erc}}} = 20.70029\%$$

and:

$$\sigma^2(x_{\text{erc}}) = 4.28502\%$$

We obtain the results given in Table 15 when  $\lambda = 4.28502\%$ . If we compare with those given in Tables 13 and 14, it is obvious that the convergence is faster in the present case.

# Portfolio optimization with CCD and ADMM algorithms

Table 15: CCD coordinates ( $k = 1, \dots, 5$ )

$k$	$i$	$\alpha_i^{(k)}$	$\beta_i^{(k)}$	$\gamma_i^{(k)}$	$x_i^{(k)}$	CCD coordinates		
						$x_1$	$x_2$	$x_3$
0						0.33333	0.33333	0.33333
1	1	0.04000	0.02033	-0.01428	0.39521	0.39521	0.33333	0.33333
1	2	0.06250	0.02738	-0.01428	0.30680	0.39521	0.30680	0.33333
1	3	0.09000	0.03033	-0.01428	0.26403	0.39521	0.30680	0.26403
2	1	0.04000	0.01718	-0.01428	0.42027	0.42027	0.30680	0.26403
2	2	0.06250	0.02437	-0.01428	0.32133	0.42027	0.32133	0.26403
2	3	0.09000	0.03200	-0.01428	0.25847	0.42027	0.32133	0.25847
3	1	0.04000	0.01734	-0.01428	0.41893	0.41893	0.32133	0.25847
3	2	0.06250	0.02404	-0.01428	0.32295	0.41893	0.32295	0.25847
3	3	0.09000	0.03204	-0.01428	0.25835	0.41893	0.32295	0.25835
4	1	0.04000	0.01737	-0.01428	0.41863	0.41863	0.32295	0.25835
4	2	0.06250	0.02403	-0.01428	0.32302	0.41863	0.32302	0.25835
4	3	0.09000	0.03203	-0.01428	0.25837	0.41863	0.32302	0.25837
5	1	0.04000	0.01738	-0.01428	0.41861	0.41861	0.32302	0.25837
5	2	0.06250	0.02403	-0.01428	0.32302	0.41861	0.32302	0.25837
5	3	0.09000	0.03203	-0.01428	0.25837	0.41861	0.32302	0.25837

# Portfolio optimization with CCD and ADMM algorithms

## Question 2

We recall that the ADMM algorithm is based on the following optimization problem:

$$\begin{aligned} \{x^*, y^*\} &= \arg \min f_x(x) + f_y(y) \\ \text{s.t. } & Ax + By = c \end{aligned}$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.a

Describe the ADMM algorithm.

# Portfolio optimization with CCD and ADMM algorithms

The ADMM algorithm consists in the following iterations:

$$\begin{cases} x^{(k+1)} = \arg \min_x \left\{ f_x(x) + \frac{\varphi}{2} \|Ax + By^{(k)} - c + u^{(k)}\|_2^2 \right\} \\ y^{(k+1)} = \arg \min_y \left\{ f_y(y) + \frac{\varphi}{2} \|Ax^{(k+1)} + By - c + u^{(k)}\|_2^2 \right\} \\ u^{(k+1)} = u^{(k)} + (Ax^{(k+1)} + By^{(k+1)} - c) \end{cases}$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.b

We consider the following optimization problem:

$$w^*(\gamma) = \arg \min \frac{1}{2} (w - b)^\top \Sigma (w - b) - \gamma (w - b)^\top \mu$$

$$\text{s.t.} \quad \begin{cases} \mathbf{1}_n^\top w = 1 \\ \sum_{i=1}^n |w_i - b_i| \leq \tau^+ \\ \mathbf{0}_n \leq w \leq \mathbf{1}_n \end{cases}$$



# Portfolio optimization with CCD and ADMM algorithms

## Question 2.b.i

Give the meaning of the symbols  $w$ ,  $b$ ,  $\Sigma$ , and  $\mu$ . What is the goal of this optimization program? What is the meaning of the constraint

$$\sum_{i=1}^n |w_i - b_i| \leq \tau^+?$$

# Portfolio optimization with CCD and ADMM algorithms

- $w$  is the vector of portfolio weights:

$$w = (w_1, \dots, w_n)$$

- $b$  is the vector of benchmark weights:

$$b = (b_1, \dots, b_n)$$

- $\Sigma$  is the covariance matrix of asset returns
- $\mu$  is the vector of expected returns

# Portfolio optimization with CCD and ADMM algorithms

The goal of the optimization problem is to tilt a benchmark portfolio by controlling the volatility of the tracking error:

$$\sigma(w | b) = \sqrt{(w - b)^\top \Sigma (w - b)}$$

and improving the expected excess return:

$$\mu(w | b) = (w - b)^\top \mu$$

This is a typical  $\gamma$ -problem when there is a benchmark

# Portfolio optimization with CCD and ADMM algorithms

We remind that the turnover between the benchmark  $b$  and the portfolio  $w$  is equal to:

$$\tau(w | b) = \sum_{i=1}^n |w_i - b_i|$$

Therefore, we impose that the turnover is less than an upper limit:

$$\tau(w | b) \leq \tau^+$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.b.ii

What is the best way to specify  $f_x(x)$  and  $f_y(y)$  in order to find numerically the solution. Justify your choice.

# Portfolio optimization with CCD and ADMM algorithms

The best way to specify  $f_x(x)$  and  $f_y(y)$  is to split the QP problem and the turnover constraint:

$$\begin{aligned} \{x^*, y^*\} &= \arg \min_{x,y} f_x(x) + f_y(y) \\ \text{s.t. } &x - y = \mathbf{0}_n \end{aligned}$$

where:

$$\begin{aligned} f_x(x) &= \frac{1}{2} (x - b)^\top \Sigma (x - b) - \gamma (x - b)^\top \mu + \mathbb{1}_{\Omega_1}(x) + \mathbb{1}_{\Omega_3}(x) \\ f_y(y) &= \mathbb{1}_{\Omega_2}(y) \\ \Omega_1(x) &= \{x : \mathbf{1}_n^\top x = 1\} \\ \Omega_2(y) &= \left\{ y : \sum_{i=1}^n |y_i - b_i| \leq \tau^+ \right\} \\ \Omega_3(x) &= \{x : \mathbf{0}_n \leq x \leq \mathbf{1}_n\} \end{aligned}$$

Indeed, the  $x$ -update step is a standard QP problem whereas the  $y$ -update step is the projection onto the  $\ell_1$ -ball  $\mathcal{B}_1(b, \tau^+)$ .

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.b.iii

Give the corresponding ADMM algorithm.

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\begin{aligned}
 (*) &= \frac{1}{2} (x - b)^\top \Sigma (x - b) - \gamma (x - b)^\top \mu \\
 &= \frac{1}{2} x^\top \Sigma x - x^\top \Sigma b + \frac{1}{2} b^\top \Sigma b - \gamma x^\top \mu + \gamma b^\top \mu \\
 &= \frac{1}{2} x^\top \Sigma x - x^\top (\Sigma b + \gamma \mu) + \underbrace{\left( \gamma b^\top \mu + \frac{1}{2} b^\top \Sigma b \right)}_{\text{constant}}
 \end{aligned}$$



# Portfolio optimization with CCD and ADMM algorithms

If we note  $v_x^{(k+1)} = y^{(k)} - u^{(k)}$ , we have:

$$\begin{aligned}
 \left\| x - y^{(k)} + u^{(k)} \right\|_2^2 &= \left\| x - v_x^{(k+1)} \right\|_2^2 \\
 &= \left( x - v_x^{(k+1)} \right)^\top \left( x - v_x^{(k+1)} \right) \\
 &= x^\top I_n x - 2x^\top v_x^{(k+1)} + \underbrace{\left( v_x^{(k+1)} \right)^\top v_x^{(k+1)}}_{\text{constant}}
 \end{aligned}$$

# Portfolio optimization with CCD and ADMM algorithms

It follows that:

$$\begin{aligned}
 f_x^{(k+1)}(x) &= f_x(x) + \frac{\varphi}{2} \left\| x - y^{(k)} + u^{(k)} \right\|_2^2 \\
 &= \frac{1}{2} (x - b)^\top \Sigma (x - b) - \gamma (x - b)^\top \mu + \\
 &\quad \mathbb{1}_{\Omega_1}(x) + \mathbb{1}_{\Omega_3}(x) + \frac{\varphi}{2} \left\| x - y^{(k)} + u^{(k)} \right\|_2^2 \\
 &= \frac{1}{2} x^\top (\Sigma + \varphi I_n) x - x^\top \left( \Sigma b + \gamma \mu + \varphi v_x^{(k+1)} \right) + \\
 &\quad \mathbb{1}_{\Omega_1}(x) + \mathbb{1}_{\Omega_3}(x) + \text{constant}
 \end{aligned}$$

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$\begin{aligned} f_y^{(k+1)}(y) &= \mathbb{1}_{\Omega_2}(y) + \frac{\varphi}{2} \left\| x^{(k+1)} - y + u^{(k)} \right\|_2^2 \\ &= \mathbb{1}_{\Omega_2}(y) + \frac{\varphi}{2} \left\| y - v_y^{(k+1)} \right\|_2^2 \end{aligned}$$

where  $v_y^{(k+1)} = x^{(k+1)} + u^{(k)}$ . We deduce that:

$$\begin{aligned} y^{(k+1)} &= \arg \min_y f_y^{(k+1)}(y) \\ &= \mathcal{P}_{\Omega_2} \left( v_y^{(k+1)} \right) \end{aligned}$$

where:

$$\Omega_2 = \mathcal{B}_1(b, \tau^+)$$

# Portfolio optimization with CCD and ADMM algorithms

We remind that:

$$\begin{aligned} \mathcal{P}_{\mathcal{B}_1(c,\lambda)}(v) &= \mathcal{P}_{\mathcal{B}_1(\mathbf{0}_n,\lambda)}(v - c) + c \\ \mathcal{P}_{\mathcal{B}_1(\mathbf{0}_n,\lambda)}(v) &= v - \text{sign}(v) \odot \mathbf{prox}_{\lambda \max_x}(|v|) \\ \mathbf{prox}_{\lambda \max_x}(v) &= \min(v, s^*) \end{aligned}$$

where  $s^*$  is the solution of the following equation:

$$s^* = \left\{ s \in \mathbb{R} : \sum_{i=1}^n (v_i - s)_+ = \lambda \right\}$$

# Portfolio optimization with CCD and ADMM algorithms

We deduce that:

$$\begin{aligned}
 \mathcal{P}_{\Omega_2} \left( v_y^{(k+1)} \right) &= \mathcal{P}_{\mathcal{B}_1(b, \tau^+)} \left( v_y^{(k+1)} \right) \\
 &= \mathcal{P}_{\mathcal{B}_1(\mathbf{0}_n, \tau^+)} \left( v_y^{(k+1)} - b \right) + b \\
 &= v_y^{(k+1)} - \text{sign} \left( v_y^{(k+1)} - b \right) \odot \mathbf{prox}_{\tau^+ \max x} \left( \left| v_y^{(k+1)} - b \right| \right) \\
 &= v_y^{(k+1)} - \text{sign} \left( v_y^{(k+1)} - b \right) \odot \min \left( \left| v_y^{(k+1)} - b \right|, s^* \right)
 \end{aligned}$$

where  $s^*$  is the solution of the following equation:

$$s^* = \left\{ s \in \mathbb{R} : \sum_{i=1}^n \left( \left| v_{y,i}^{(k+1)} - b_i \right| - s \right)_+ = \tau^+ \right\}$$

# Portfolio optimization with CCD and ADMM algorithms

The ADMM algorithm becomes:

$$\left\{ \begin{array}{l} v_x^{(k+1)} = y^{(k)} - u^{(k)} \\ Q^{(k+1)} = \Sigma + \varphi I_n \\ R^{(k+1)} = \Sigma b + \gamma \mu + \varphi v_x^{(k+1)} \\ x^{(k+1)} = \arg \min_x \left\{ \frac{1}{2} x^\top Q^{(k+1)} x - x^\top R^{(k+1)} + \mathbb{1}_{\Omega_1}(x) + \mathbb{1}_{\Omega_3}(x) \right\} \\ v_y^{(k+1)} = x^{(k+1)} + u^{(k)} \\ s^* = \left\{ s \in \mathbb{R} : \sum_{i=1}^n \left( \left| v_{y,i}^{(k+1)} - b_i \right| - s \right)_+ = \tau^+ \right\} \\ y^{(k+1)} = v_y^{(k+1)} - \text{sign} \left( v_y^{(k+1)} - b \right) \odot \min \left( \left| v_y^{(k+1)} - b \right|, s^* \right) \\ u^{(k+1)} = u^{(k)} + x^{(k+1)} - y^{(k+1)} \end{array} \right.$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.c

We consider the following optimization problem:

$$\begin{aligned} w^* &= \arg \min \|w - \tilde{w}\|_1 \\ \text{s.t.} & \begin{cases} \mathbf{1}_n^\top w = 1 \\ \sqrt{(w - b)^\top \Sigma (w - b)} \leq \sigma^+ \\ \mathbf{0}_n \leq w \leq \mathbf{1}_n \end{cases} \end{aligned}$$

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.c.i

What is the meaning of the objective function  $\|w - \tilde{w}\|_1$ ? What is the meaning of the constraint  $\sqrt{(w - b)^\top \Sigma (w - b)} \leq \sigma^+$ ?



# Portfolio optimization with CCD and ADMM algorithms

The objective function  $\|w - \tilde{w}\|_1$  is the turnover between a given portfolio  $\tilde{w}$  and the optimized portfolio  $w$

The constraint  $\sqrt{(w - b)^\top \Sigma (w - b)} \leq \sigma^+$  is a tracking error limit with respect to a benchmark  $b$

# Portfolio optimization with CCD and ADMM algorithms

## Question 2.c.ii

Propose an equivalent optimization problem such that  $f_x(x)$  is a QP problem. How to solve the  $y$ -update?

# Portfolio optimization with CCD and ADMM algorithms

The optimization problem is equivalent to solve the following program:

$$\begin{aligned} w^* &= \arg \min \frac{1}{2} (w - b)^\top \Sigma (w - b) + \lambda \|w - \tilde{w}\|_1 \\ \text{s.t.} & \begin{cases} \mathbf{1}_n^\top w = 1 \\ \mathbf{0}_n \leq w \leq \mathbf{1}_n \end{cases} \end{aligned}$$

# Portfolio optimization with CCD and ADMM algorithms

We deduce that:

$$f_x(x) = \frac{1}{2} (x - b)^\top \Sigma (x - b) + \mathbb{1}_{\Omega_1}(x) + \mathbb{1}_{\Omega_2}(x)$$

where:

$$\Omega_1(x) = \{x : \mathbf{1}_n^\top x = 1\}$$

and:

$$\Omega_2(x) = \{x : \mathbf{0}_n \leq x \leq \mathbf{1}_n\}$$

# Portfolio optimization with CCD and ADMM algorithms

We have:

$$f_y(y) = \lambda \|w - \tilde{w}\|_1$$

We remind that:

$$\mathbf{prox}_{\lambda\|x\|_1}(v) = \mathcal{S}(v; \lambda) = \text{sign}(v) \odot (|v| - \lambda \mathbf{1}_n)_+$$

and:

$$\mathbf{prox}_{f(x+b)}(v) = \mathbf{prox}_f(v + b) - b$$

The  $y$ -update step is then equal to:

$$\begin{aligned} y^{(k+1)} &= \mathbf{prox}_{\lambda\|w-\tilde{w}\|_1} \left( x^{(k+1)} + u^{(k)} \right) \\ &= \tilde{w} + \text{sign} \left( x^{(k+1)} + u^{(k)} - \tilde{w} \right) \odot \left( \left| x^{(k+1)} + u^{(k)} - \tilde{w} \right| - \lambda \mathbf{1}_n \right)_+ \end{aligned}$$

because  $f_y(y)$  is fully separable<sup>5</sup>

---

<sup>5</sup>Otherwise the scaling property does not work!

# Regularized portfolio optimization

## Exercise

We consider an investment universe with 6 assets. We assume that their expected returns are 4%, 6%, 7%, 8%, 10% and 10%, and their volatilities are 6%, 10%, 11%, 15%, 15% and 20%. The correlation matrix is given by:

$$\rho = \begin{pmatrix} 100\% & & & & & \\ 50\% & 100\% & & & & \\ 20\% & 20\% & 100\% & & & \\ 50\% & 50\% & 80\% & 100\% & & \\ 0\% & -20\% & -50\% & -30\% & 100\% & \\ 0\% & 20\% & 30\% & 0\% & 0\% & 100\% \end{pmatrix}$$

# Regularized portfolio optimization

## Question 1

We restrict the analysis to long-only portfolios meaning that  $\sum_{i=1}^n x_i = 1$  and  $x_i \geq 0$ .

# Regularized portfolio optimization

## Question 1.a

We consider the Herfindahl index  $\mathcal{H}(x) = \sum_{i=1}^n x_i^2$ . What are the two limit cases of  $\mathcal{H}(x)$ ? What is the interpretation of the statistic  $\mathcal{N}(x) = \mathcal{H}^{-1}(x)$ ?



# Regularized portfolio optimization

We consider the following optimization problem:

$$\begin{aligned} x^* &= \arg \min \mathcal{H}(x) \\ \text{s.t. } &\sum_{i=1}^n x_i = 1 \end{aligned}$$

We deduce that the Lagrange function is:

$$\begin{aligned} \mathcal{L}(x; \lambda) &= \mathcal{H}(x) - \lambda \left( \sum_{i=1}^n x_i - 1 \right) \\ &= x^\top x - \lambda (\mathbf{1}_n^\top x - 1) \end{aligned}$$

# Regularized portfolio optimization

The first-order condition is:

$$\frac{\partial \mathcal{L}(x; \lambda)}{\partial x} = x - \lambda \mathbf{1}_n = \mathbf{0}_n$$

Since we have  $\mathbf{1}_n^\top x - 1 = 0$ , we deduce that:

$$\lambda = \frac{1}{\mathbf{1}_n^\top \mathbf{1}_n} = \frac{1}{n}$$

We conclude that the lower bound is reached for the equally-weighted portfolio:

$$x_{\text{ew}} = \frac{1}{n} \cdot \mathbf{1}_n$$

and we have:

$$\mathcal{H}(x_{\text{ew}}) = \frac{1}{n^2} \cdot \mathbf{1}_n^\top \mathbf{1}_n = \frac{1}{n}$$

# Regularized portfolio optimization

Since the weights are positive, we have:

$$\begin{aligned}\mathcal{H}(x) &= \sum_{i=1}^n x_i^2 \\ &\leq \left( \sum_{i=1}^n x_i \right)^2 \\ &\leq 1\end{aligned}$$

The upper bound is reached when the portfolio is concentrated on one asset:

$$\exists i : x_i = 1$$

# Regularized portfolio optimization

We conclude that:

$$\frac{1}{n} \leq \mathcal{H}(x) \leq 1$$

The statistic  $\mathcal{N}(x) = \mathcal{H}^{-1}(x)$  is the effective number of assets

# Regularized portfolio optimization

## Question 1.b

We consider the following optimization problem ( $\mathcal{P}_1$ ):

$$\begin{aligned} x^*(\lambda) &= \arg \min \frac{1}{2} x^\top \Sigma x + \lambda x^\top x \\ \text{s.t.} & \begin{cases} \sum_{i=1}^n x_i = 1 \\ x_i \geq 0 \end{cases} \end{aligned}$$

What is the link between this constrained optimization program and the weight diversification based on the Herfindahl index?

# Regularized portfolio optimization

The optimization problem ( $\mathcal{P}_1$ ) is equivalent to:

$$x^*(\mathcal{H}^+) = \arg \min \frac{1}{2} x^\top \Sigma x$$

$$\text{s.t.} \quad \begin{cases} \sum_{i=1}^n x_i = 1 \\ x_i \geq 0 \\ x^\top x \leq \mathcal{H}^+ \end{cases}$$

We obtain a long-only minimum variance portfolio with a diversification constraint based on the Herfindahl index:

$$\mathcal{H}(x) \leq \mathcal{H}^+$$

We have the following correspondance:

$$\mathcal{H}^+ = \mathcal{H}(x^*(\lambda)) = x^*(\lambda)^\top x^*(\lambda)$$

Given a value of  $\lambda$ , we can then compute the implicit constraint  $\mathcal{H}(x) \leq \mathcal{H}^+$ .

# Regularized portfolio optimization

## Question 1.c

Solve Program  $(\mathcal{P}_1)$  when  $\lambda$  is equal to respectively 0, 0.001, 0.01, 0.05, 0.10 and 10. Compute the statistic  $\mathcal{N}(x)$ . Comment on these results.

# Regularized portfolio optimization

**Table 16:** Solution of the optimization problem ( $\mathcal{P}_1$ )

$\lambda$	0.000	0.001	0.010	0.050	0.100	10.000
$x_1^*(\lambda)$ (in %)	44.60	35.66	23.97	18.71	17.76	16.68
$x_2^*(\lambda)$ (in %)	9.12	14.60	18.10	17.08	16.89	16.67
$x_3^*(\lambda)$ (in %)	25.46	26.57	19.96	16.89	16.71	16.67
$x_4^*(\lambda)$ (in %)	0.00	0.00	7.64	14.46	15.52	16.65
$x_5^*(\lambda)$ (in %)	20.40	22.11	22.38	19.31	18.21	16.69
$x_6^*(\lambda)$ (in %)	0.43	1.07	7.94	13.55	14.92	16.65
$\mathcal{H}(x^*(\lambda))$	0.3137	0.2680	0.1923	0.1693	0.1675	0.1667
$\mathcal{N}(x^*(\lambda))$	3.19	3.73	5.20	5.91	5.97	6.00



# Regularized portfolio optimization

## Question 1.d

Using the bisection algorithm, find the optimal value of  $\lambda^*$  that satisfies:

$$\mathcal{N}(x^*(\lambda^*)) = 4$$

Give the composition of  $x^*(\lambda^*)$ . What is the interpretation of  $x^*(\lambda^*)$ ?

# Regularized portfolio optimization

The optimal solution is:

$$\lambda^* = 0.002301$$

The optimal weights (in %) are equal to:

$$x^* = \begin{pmatrix} 31.62\% \\ 17.24\% \\ 26.18\% \\ 0.00\% \\ 22.63\% \\ 2.33\% \end{pmatrix}$$

The effective number of bets  $\mathcal{N}(x^*)$  is equal to 4

# Regularized portfolio optimization

## Question 2

We consider long/short portfolios and the following optimization problem ( $\mathcal{P}_2$ ):

$$\begin{aligned} x^*(\lambda) &= \arg \min \frac{1}{2} x^\top \Sigma x + \lambda \sum_{i=1}^n |x_i| \\ \text{s.t.} \quad & \sum_{i=1}^n x_i = 1 \end{aligned}$$

# Regularized portfolio optimization

## Question 2.a

Solve Program  $(\mathcal{P}_2)$  when  $\lambda$  is equal to respectively 0, 0.0001, 0.001, 0.01, 0.05, 0.10 and 10. Comment on these results.

# Regularized portfolio optimization

**Table 17:** Solution of the optimization problem ( $\mathcal{P}_2$ )

$\lambda$	0.000	0.0001	0.001	0.010	0.050	0.100	10.000
$x_1^*(\lambda)$ (in %)	35.82	37.17	44.50	44.60	44.60	44.60	44.60
$x_2^*(\lambda)$ (in %)	33.08	30.26	11.48	9.12	9.12	9.12	9.12
$x_3^*(\lambda)$ (in %)	77.62	71.77	31.28	25.46	25.46	25.46	25.46
$x_4^*(\lambda)$ (in %)	-53.48	-47.97	-7.16	0.00	0.00	0.00	0.00
$x_5^*(\lambda)$ (in %)	20.83	20.56	19.90	20.40	20.40	20.40	20.40
$x_6^*(\lambda)$ (in %)	-13.87	-11.78	0.00	0.43	0.43	0.43	0.43
$\mathcal{L}(x)$ (in %)	234.69	219.50	114.33	100.00	100.00	100.00	100.00

# Regularized portfolio optimization

## Question 2.b

For each optimized portfolio, calculate the following statistic:

$$\mathcal{L}(x) = \sum_{i=1}^n |x_i|$$

What is the interpretation of  $\mathcal{L}(x)$ ? What is the impact of Lasso regularization?

# Regularized portfolio optimization

$\mathcal{L}(x) = \sum_{i=1}^n |x_i|$  is the leverage ratio. Their values are reported in Table 17.

# Regularized portfolio optimization

## Question 3

We assume that the investor holds an initial portfolio  $x^{(0)}$  defined as follows:

$$x^{(0)} = \begin{pmatrix} 10\% \\ 15\% \\ 20\% \\ 25\% \\ 30\% \\ 0\% \end{pmatrix}$$

We consider the optimization problem ( $\mathcal{P}_3$ ):

$$x^*(\lambda) = \arg \min \frac{1}{2} x^\top \Sigma x + \lambda \sum_{i=1}^n |x_i - x_i^{(0)}|$$

$$\text{s.t. } \sum_{i=1}^n x_i = 1$$



# Regularized portfolio optimization

## Question 3.a

Solve Program  $(\mathcal{P}_3)$  when  $\lambda$  is equal respectively to 0, 0.0001, 0.001, 0.0015 and 0.01. Compute the turnover of each optimized portfolio. Comment on these results.

# Regularized portfolio optimization

**Table 18:** Solution of the optimization problem ( $\mathcal{P}_3$ )

$\lambda$	0.000	0.000	0.001	0.002	0.010
$x_1^*(\lambda)$ (in %)	35.82	35.55	27.90	24.28	10.00
$x_2^*(\lambda)$ (in %)	33.08	30.61	15.00	15.00	15.00
$x_3^*(\lambda)$ (in %)	77.62	72.35	33.36	22.86	20.00
$x_4^*(\lambda)$ (in %)	-53.48	-48.00	-5.20	7.87	25.00
$x_5^*(\lambda)$ (in %)	20.83	21.51	28.94	30.00	30.00
$x_6^*(\lambda)$ (in %)	-13.87	-12.02	0.00	0.00	0.00
$\tau(x^*(\lambda)   x^{(0)})$ (in %)	203.04	187.02	62.51	34.27	0.00

# Regularized portfolio optimization

## Question 3.b

Using the bisection algorithm, find the optimal value of  $\lambda^*$  such that the two-way turnover is equal to 60%. Give the composition of  $x^*(\lambda^*)$ .

# Regularized portfolio optimization

The optimal solution is:

$$\lambda^* = 0.00103$$

The optimal weights (in %) are equal to:

$$x^* = \begin{pmatrix} 27.23\% \\ 15.00\% \\ 32.77\% \\ -4.30\% \\ 29.30\% \\ 0.00\% \end{pmatrix}$$

The turnover  $\tau(x^* | x^{(0)})$  is equal to 60%

# Regularized portfolio optimization

## Question 3.c

Same question when the two-way turnover is equal to 50%.

# Regularized portfolio optimization

The optimal solution is:

$$\lambda^* = 0.00119$$

The optimal weights (in %) are equal to:

$$x^* = \begin{pmatrix} 25.53\% \\ 15.00\% \\ 29.47\% \\ 0.00\% \\ 30.00\% \\ 0.00\% \end{pmatrix}$$

The turnover  $\tau(x^* | x^{(0)})$  is equal to 50%

# Regularized portfolio optimization

## Question 3.d

What becomes the portfolio  $x^*(\lambda)$  when  $\lambda \rightarrow \infty$ ? How do you explain this result?

# Regularized portfolio optimization

We notice that:

$$\lim_{\lambda \rightarrow \infty} x^*(\lambda) = x^{(0)}$$

This is normal since we have:

$$x^*(\lambda) = \arg \min \frac{1}{2} x^\top \Sigma x + \lambda \sum_{i=1}^n |x_i - x_i^{(0)}|$$

$$\text{s.t. } \sum_{i=1}^n x_i = 1$$

We deduce that:

$$x^*(\infty) = \arg \min \sum_{i=1}^n |x_i - x_i^{(0)}|$$

$$\text{s.t. } \sum_{i=1}^n x_i = 1$$

The solution is  $x^*(\infty) = x^{(0)}$



# Main references



BECK, A. (2017)

*First-Order Methods in Optimization*, MOS-SIAM Series on Optimization, 25, SIAM.



COQUERET, G., and GUIDA, T. (2020)

*Machine Learning for Factor Investing*, Chapman and Hall/CRC Financial Mathematics Series.



PERRIN, S., and RONCALLI, T. (2020)

Machine Learning Algorithms and Portfolio Optimization, in Jurczenko, E. (Ed.), *Machine Learning in Asset Management: New Developments and Financial Applications*, Wiley, pp. 261-328, [arxiv.org/abs/1909.10233](https://arxiv.org/abs/1909.10233).

# References I



BOURGERON, T., LEZMI, E., and RONCALLI, T. (2018)

Robust Asset Allocation for Robo-Advisors, *arXiv*,  
[arxiv.org/abs/1902.07449](https://arxiv.org/abs/1902.07449).



BOYD, S., PARIKH, N., CHU, E., PELEATO, B., and ECKSTEIN, J.  
(2010)





Distributed Optimization and Statistical Learning via the Alternating  
Direction Method of Multipliers, *Foundations and Trends® in  
Machine learning*, 3(1), pp. 1-122.



GABAY, D., and MERCIER, B. (1976)

A Dual Algorithm for the Solution of Nonlinear Variational Problems  
via Finite Element Approximation, *Computers & Mathematics with  
Applications*, 2(1), pp. 17-40.

## References II

-  [GONZALVEZ, J., LEZMI, E., RONCALLI, T., and XU, J. \(2019\)](#)  
Financial Applications of Gaussian Processes and Bayesian Optimization, *arXiv*, [arxiv.org/abs/1903.04841](https://arxiv.org/abs/1903.04841).
-  [GRIVEAU-BILLION, T., RICHARD, J-C., and RONCALLI, T. \(2013\)](#)  
A Fast Algorithm for Computing High-dimensional Risk Parity Portfolios, *SSRN*, [www.ssrn.com/abstract=2325255](https://www.ssrn.com/abstract=2325255).
-  [JURCZENKO, E. \(2020\)](#)  
*Machine Learning in Asset Management: New Developments and Financial Applications*, Wiley.
-  [KONDRATYEV, A., and SCHWARZ, C. \(2020\)](#)  
The Market Generator, *SSRN*, [www.ssrn.com/abstract=3384948](https://www.ssrn.com/abstract=3384948).

## References III



LEZMI, E., ROCHE, J., RONCALLI, T., and XU, J. (2020)  
Improving the Robustness of Trading Strategy Backtesting with Boltzmann Machines and Generative Adversarial Networks, *arXiv*, <https://arxiv.org/abs/2007.04838>.



PARIKH, N., and BOYD, S. (2014)  
Proximal Algorithms, *Foundations and Trends® in Optimization*, 1(3), pp. 127-239.



TIBSHIRANI, R. (1996)  
Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society B*, 58(1), pp. 267-288.

## References IV



[TIBSHIRANI, R.J. \(2017\)](#)

Dykstra's Algorithm, ADMM, and Coordinate Descent: Connections, Insights, and Extensions, in Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (Eds), *Advances in Neural Information Processing Systems*, 30, pp. 517-528.



[TSENG, P. \(2001\)](#)

Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization, *Journal of Optimization Theory and Applications*, 109(3), pp. 475-494.