Machine Learning for Asset Management
Tutorial 3

Contents

1 Build portfolios with a risk budgeting approach

Tutorial 2 allowed you to create portfolios using mean-variance optimization. You saw that the turnover could be (very) high. An alternative approach to portfolio construction is risk budgeting (RB – see Lecture 5), which (indirectly) helps manage the turnover problem since only the covariance matrix is used. This approach typically leads to portfolios that are more stable than those derived from mean-variance optimization, especially when the expected returns are constructed from rolling past performance.

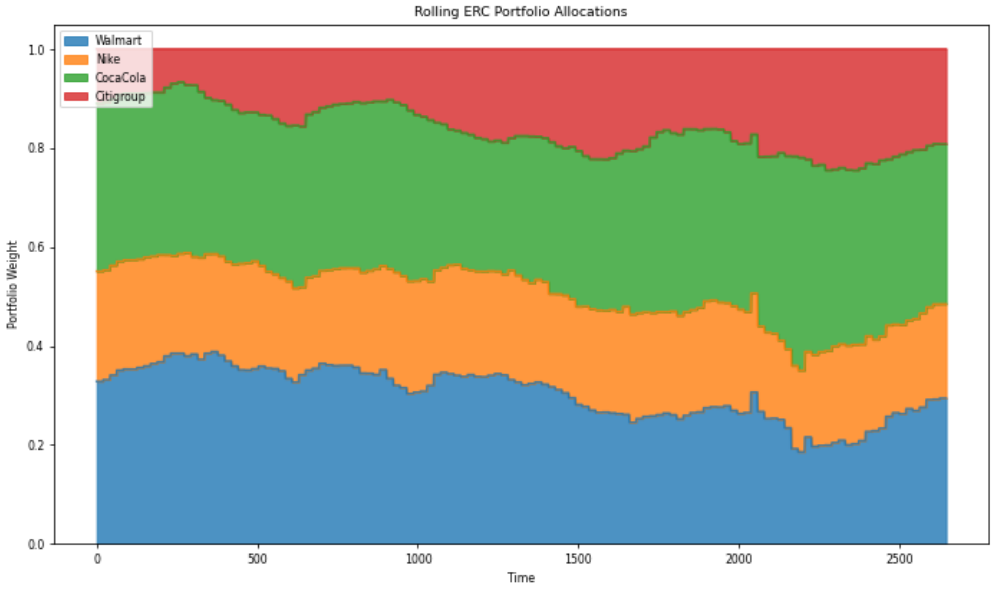
In the first part of this tutorial, the goal is to find optimal portfolios using various simple criteria for a given covariance matrix. This section concludes with the construction of rolling ERC (Equal Risk Contribution) portfolios. You will complete the program *Tutorial3_RB_Students.py*. All the questions (there are 12) are embedded in this code.

You will use the *minimize* function from package *scipy.optimize*. If you are not familiar with this function, the first step is to check the documentation to see how it is parameterized. This advice applies to any new function you encounter.

You will finish this part by plotting the stock weights (using the same stocks as in Tutorial 2) in the portfolios, and your chart should be similar to the following:

More mathematical details are provided in the appendix.

Figure 1: Portefeuilles de risk budgeting simple (ERC)



2 Random Forests

2.1 Data

We will use a simple dataset containing 40 of large companies (by market capitalization) along with several quantitative factors. In practice, models are typically estimated on past (lagged) data and then applied to the most recent factor data (or features) to generate predictions; here, note that the estimation is in-sample.
In general, the first step in any modeling problem is cleaning the input data. Fortunately, random forests are robust to outliers in the data space, making this step relatively painless. So what we need to do is impute the missing value based on its similarity to other observations using the *IterativeImputer* function.

2.2 Random Forests - Part 1

The source code for this section is *Tutorial3_RF1_Students.py*; it contains 13 questions. Right from the start, we will estimate a random forest. Later, we will see how to:

- Use an estimated model to make predictions
- Visualize the density of the estimates made by the individual trees
- Identify the important variables
- Understand how parameters affect the results by varying the number of trees and the number of features used in each tree

2.3 Random Forests - Part 2

The source code for this section is *Tutorial3_RF2_Students.py*; it contains 13 questions. Here, we will continue to examine how to fine-tune the random forest model. This includes:

- Preprocessing and splitting the data
- Evaluating and optimizing models using cross validation
- Building a gradient boosting model

Appendix

Risk budgeting approach

The fundamental principle of the risk budgeting (RB) approach is to allocate funds based on risk, rather than capital, as stated in Roncalli[1]. To achieve this, we introduce the concept of risk contribution, which is characterized as the contribution of each asset in the portfolio to the portfolio's overall risk. The portfolio manager defines a set of risk budgets and then determines the weights of the portfolio such that the risk contributions are in line with the budgets.

From a mathematical point of view, a risk budgeting portfolio is defined as follows:

$$\begin{cases} RC_i(x) = b_i R(x) \\ b_i > 0, x_i \geq 0, \text{for all i} \\ \sum_{i=1}^{n} b_i = 1, \sum_{i=1}^{n} x_i = 1 \end{cases} \tag{1}$$

where x_i is the allocation of Asset i, $R(x)$ is the risk of the portfolio, which is typically the volatility of the portfolio, $RC_i(x)$ and b_i are respectively the risk contribution and the risk budget of Asset i.

A route to solving Problem (1) is to transform the non-linear system into an optimization problem:

$$x^\star = \arg\min \sum_n \big(RC_i(x) - b_i R(x)\big)^2$$

s.t.

$$\begin{cases} x_i \geq 0, b_i \geq 0, \text{for all i} \\ 1^T x = 1 \\ 1^T b = 1 \end{cases} \tag{2}$$

However, Problem (2) is not a convex problem, and the optimization has some numerical issues, particularly in the high-dimensional case, that is, when the number of assets is large. A different approach to solving Problem (1) with the help of the logarithmic barrier method and the solution is:

$$x_{RB} = \frac{y^\star}{1^T y^\star}$$

[1] Roncalli, T. (2013), *Introduction to Risk Parity and Budgeting*

where $y^\star(c)$ is the solution of the alternative optimization problem:

$$y^\star(c) = \arg\min R(y)$$

s.t.

$$\begin{cases} \sum_{i=1}^{n} b_i \log(y_i) \geq c \\ y_i \geq 0 \text{ for all i} \end{cases} \tag{3}$$

where c is an arbitrary scalar.This problem can be solved by the Newton algorithm and the Cyclical Coordinate Descent (CCD) algorithm. In the special case of the equal risk contribution (ERC) portfolio, i.e. the risk budgets are the same ($b_i = b_j$ for all i, j), we have:

$$y^\star(c) = \arg\min R(y)$$

s.t.

$$\begin{cases} \sum_{i=1}^{n} \log(y_i) \geq c \\ y_i \geq 0 \text{ for all i} \end{cases} \tag{4}$$

Finally:

$$x_{ERC} = \frac{y^\star}{1^T y^\star}$$

This portfolio allocation strategy is also known as the risk parity approach, which is a special case of Risk Budgeting, an alternative method to the traditional mean-variance portfolio optimization.

Formulation of the opimization problems in Tutorial 3

Part I : Equal risk contribution

$$x^\star = \arg\min -\sum_{i=1}^{n} \log(x_i)$$

s.t.

$$\begin{cases} R(x) \geq R_{target} \\ x_i \geq 0 \text{ for all i} \end{cases} \tag{5}$$

Finally:

$$x_{ERC} = \frac{x^\star}{1^T x^\star}$$

Part II : Long / short optimization

$$x^\star = \arg\min -\sum_{i=1}^{n} |\mu_i| \log(|x_i|)$$

s.t.

$$\begin{cases} R(x) \geq R_{target} \\ x_i \geq 0 \text{ for all i} \end{cases} \tag{6}$$

Finally: where μ_i is the expected return of Asset i Finally, $x_{LS} = x\star$

Part III : Long / short optimization with market neutrality constraint

$$x^\star = \arg\min - \sum_{i=1}^{n} |\mu_i| \log(|x_i|)$$

s.t.

$$\begin{cases} R(x) \geq R_{target} \\ x_i \geq 0 \text{ for all i} \\ \sum_{i=1}^{n} x_i = 0 \end{cases} \tag{7}$$

Finally: where μ_i is the expected return of Asset i Finally, $x_{LSMN} = x\star$