MIE 1622H: Assignment 2 – Risk-Based and Robust Portfolio Selection Strategies

Dr. Oleksandr Romanko, Mohammadreza Mohammadi

February 10, 2025

Due: Friday, March 7, 2025, not later than 11:59 p.m.

Use Python for all MIE 1622H assignments.

You should hand in:

- Your report (pdf file and docx file). Maximum page limit is 8 pages.
- Compress all of your Python code (i.e., ipynb notebook with plots and necessary outputs intact) into a zip file and submit it via Quercus portal no later than 11:59 p.m. on March 7.

Where to hand in: Online via Quercus portal, both your code and report.

Introduction

The purpose of this assignment is to compare computational investment strategies based on selecting portfolio with equal risk contributions, using robust mean-variance optimization and applying passive investment strategy via benchmark tracking optimization. In Assignment 1, you have already implemented and compared computational investment strategies based on minimizing portfolio variance, maximizing portfolio expected return, maximizing Sharpe ratio as well as "equally weighted" and "buy and hold" portfolio strategies.

You are proposed to test nine strategies (use your implementation of strategies 1-5 from Assignment 1):

1. "Buy and hold" strategy;
2. "Equally weighted" (also known as "$1/n$") portfolio strategy;
3. "Minimum variance" portfolio strategy;
4. "Maximum expected return" portfolio strategy;
5. "Maximum Sharpe ratio" portfolio strategy;
6. "Equal risk contributions" portfolio strategy;
7. "Leveraged max Sharpe Ratio" portfolio strategy;
8. "Robust mean-variance optimization" portfolio strategy.
9. "Benchmark tracking optimization" portfolio strategy.

If a portfolio strategy relies on some targets, e.g., target return estimation error, you need to select those targets consistently for each holding period. It is up to you to decide how to select those targets.

Feel free to use Python prototypes provided in class.

Questions

1. **(50 %) Implement investment strategies in Python:**

   You need to test four portfolio re-balancing strategies:

   1. "Equal risk contributions" portfolio strategy: compute a portfolio that has equal risk contributions to standard deviation for each period and re-balance accordingly. You can use IPOPT example from the lecture notes, but you need to compute the gradient of the objective function yourself (to validate your gradient computations you may use finite differences method). The strategy should be implemented in the function `strat_equal_risk_contr`.
   2. "Leveraged max Sharpe Ratio" portfolio strategy: take long 200% position in max Sharpe ratio portfolio and short risk-free asset for each period and re-balance accordingly, implement it in the function `strat_lever_max_Sharpe`. You do not have to include risk-free asset as an additional asset when calculating optimal positions. Just make sure that you subtract amount borrowed (including interest at the risk-free rate) when calculating portfolio value in a similar way as you do for transaction cost at each time period.
   3. "Robust mean-variance optimization" portfolio strategy: compute a robust mean-variance portfolio for each period and re-balance accordingly. You can use CPLEX example from the lecture notes, but you need to select target risk estimation error and target return according to your preferences as an investor (provide justification of your choices). The strategy should be implemented in the function `strat_robust_optim`.
   4. "Benchmark tracking" portfolio strategy: for each period, select a portfolio that aims to replicate or closely track the provided benchmark portfolio (S&P30). The S&P30 benchmark (index) portfolio is composed of the same 30 stocks used in this assignment, with each stock's weight determined by its market capitalization relative to the total market cap of the 30 stocks. Asset weights in the benchmark portfolio w_b are provided in the template code. Your optimization formulation should minimize *tracking error* (squared), which is defined as the difference between your portfolio's variance of returns and the benchmark's variance of returns, subject to standard portfolio constraints plus a cardinality constraint. The cardinality constraint should allow selection of at most ten stocks for your benchmark-tracking strategy. Implement this in the function `strat_tracking_index`.

   Design and implement a rounding procedure, so that you always trade (buy or sell) an integer number of shares.

   Design and implement a validation procedure in your code to test that each of your strategies is feasible (you have enough budget to re-balance portfolio, you correctly compute transaction costs, funds in your cash account are non-negative).

   There is a file **portf_optim2.ipynb** on the course web-page. You are required to complete the code in the file and run your script for 3 CSV datasets explained below.

   Your Python code should use only CPLEX and IPOPT optimization solvers.

   For this assignment you need to use Google Colab and install the trial version of CPLEX (available on Colab) with `!pip install cplex`. To install IPOPT solver on Google Colab, please install necessary libraries as shown in `ipopt_example.ipynb` file from Lecture 1, and after that install `cyipopt` module with `!pip install cyipopt`.

2. **(15 %) Analyze your results:**

- Produce the following output for the 12 periods (years 2020 and 2021):

```
Period 1: start date 01/02/2020, end date 02/28/2020
  Strategy "Buy and Hold", value begin = $ 1000016.96, value end = $ 887595.87, cash account = $0.00
  Strategy "Equally Weighted Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Minimum Variance Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Maximum Expected Return Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Maximum Sharpe Ratio Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Equal Risk Contributions Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Leveraged Max Sharpe Ratio Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Robust Optimization Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Benchmark Tracking Portfolio", value begin = ... , value end = ... , cash account = ...
  ...

Period 12: start date 11/01/2021, end date 12/31/2021
  Strategy "Buy and Hold", value begin = $ 964589.81, value end = $ 942602.39, cash account = $0.00
  Strategy "Equally Weighted Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Minimum Variance Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Maximum Expected Return Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Maximum Sharpe Ratio Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Equal Risk Contributions Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Leveraged Max Sharpe Ratio Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Robust Optimization Portfolio", value begin = ... , value end = ... , cash account = ...
  Strategy "Benchmark Tracking Portfolio", value begin = ... , value end = ... , cash account = ...
```

- Plot one chart in Python that illustrates the daily value of your portfolio (for each of the nine trading strategies) over the years 2020 and 2021 using daily prices provided. Include the chart in your report.
- Plot one chart in Python that illustrates maximum drawdown of your portfolio (for each of the nine trading strategies) for each of the 12 periods (years 2020 and 2021) using daily prices provided. Include the chart in your report.
- Plot one chart in Python to show dynamic changes in portfolio allocations under strategy 8. In each chart, x-axis represents the rolling up time horizon, y-axis denotes portfolio weights between 0 and 1, and distinct lines display the position of selected assets over time periods. Does your robust portfolio selection strategy reduce trading as compared with strategies 3, 4 and 5 that you have implemented in Assignment 1?
- Compare your "equal risk contributions", "leveraged max Sharpe ratio", "robust mean-variance optimization" and "benchmark tracking" trading strategies between each other and to five strategies implemented in Assignment 1 and discuss their performance relative to each other. Which strategy would you select for managing your own portfolio and why?

3. **(20 %) Test your trading strategies for years 2008 and 2009:**

- Download daily closing prices for the same thirty stocks for years 2008 and 2009 (file **adjclose_2008_2009.csv**). If necessary, transform data into a format that your code can read.
- Test nine strategies that you have implemented for the 12 periods (years 2008 and 2009). Use the same initial portfolio that you are given in Assignment 1. Produce the same output for your strategies as in Question 2.
- Plot one chart in Python that illustrates the daily value of your portfolio (for each of the nine trading strategies) over the years 2008 and 2009 using daily prices that you have downloaded. Include the chart in your report.

- Plot one chart in Python that illustrates maximum drawdown of your portfolio (for each of the nine trading strategies) for each of the 12 periods (years 2008 and 2009) using daily prices provided. Include the chart in your report.
- Plot four charts in Python for strategies 3, 4, 5 and 8 to show dynamic changes in portfolio allocations using the new data set. Does your robust portfolio selection strategy reduce trading as compared with the strategies 3, 4 and 5?
- Compare and discuss relative performance of your nine trading strategies during 2020-2021 and 2008-2009 time periods. Which strategy would you select for managing your own portfolio during 2008-2009 time period and why?

4. **(15 %) Test your trading strategies for year 2022:**

   - Download daily closing prices for the same thirty stocks for year 2022 (file **adjclose_2022.csv**). If necessary, transform data into a format that your code can read.
   - Test nine strategies that you have implemented for the 6 periods (year 2022). Use the same initial portfolio that you are given in Assignment 1. Produce the same output for your strategies as in Question 2.
   - Plot one chart in Python that illustrates the daily value of your portfolio (for each of the nine trading strategies) over the year 2022 using daily prices that you have downloaded. Include the chart in your report.
   - Plot one chart in Python that illustrates maximum drawdown of your portfolio (for each of the nine trading strategies) for each of the 6 periods (year 2022) using daily prices provided. Include the chart in your report.
   - Compare the maximum drawdown for 2022 to the 2008-2009 maximum drawdown? How do the two periods called "recessions" compare?

Python Code to be Completed (available on Quercus)

```
# Import libraries
import pandas as pd
import numpy as np
import math
from scipy import sparse
import matplotlib.pyplot as plt
%matplotlib inline

# Install CPLEX and IPOPT, if necessary.
import cplex
import cyipopt as ipopt

# Complete the following functions
def strat_buy_and_hold(x_init, cash_init, mu, Q, cur_prices):
   x_optimal = x_init
   cash_optimal = cash_init
   return x_optimal, cash_optimal

def strat_equally_weighted(x_init, cash_init, mu, Q, cur_prices):

    return x_optimal, cash_optimal

def strat_min_variance(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

def strat_max_return(x_init, cash_init, mu, Q, cur_prices):

    return x_optimal, cash_optimal

def strat_max_Sharpe(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

def strat_equal_risk_contr(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

def strat_lever_max_Sharpe(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

def strat_robust_optim(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

def strat_tracking_index(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

# Input file
input_file_prices = 'adjclose_2020_2021.csv'

# Read data into a dataframe
df = pd.read_csv(input_file_prices)

# Convert dates into array [year month day]
# Convert dates into array [year month day]
def convert_date_to_array(datestr):
    temp = [int(x) for x in datestr.split('/')]
    return [temp[-1], temp[0], temp[1]]

dates_array = np.array(list(df['Date'].apply(convert_date_to_array)))
data_prices = df.iloc[:, 1:].to_numpy()
dates = np.array(df['Date'])
# Find the number of trading days in Nov-Dec 2019 and
```

```
# compute expected return and covariance matrix for period 1
day_ind_start0 = 0
day_ind_end0 = len(np.where(dates_array[:,0]==2019)[0])   # for 2020-2021 csv

#day_ind_end0 = len(np.where(dates_array[:,0]==2007)[0])   # for 2008-2009 csv
#day_ind_end0 = len(np.where(dates_array[:,0]==2021)[0])   # for 2022 csv

cur_returns0 = data_prices[day_ind_start0+1:day_ind_end0,:] / data_prices[day_ind_start0:day_ind_end0-1,:] - 1
mu = np.mean(cur_returns0, axis = 0)
Q = np.cov(cur_returns0.T)

# Remove datapoints for year 2019
data_prices = data_prices[day_ind_end0:,:]
dates_array = dates_array[day_ind_end0:,:]
dates = dates[day_ind_end0:]

# Initial positions in the portfolio
init_positions = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3447, 0, 0, 0,
                           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 19385, 0])

# Initial value of the portfolio
init_value = np.dot(data_prices[0,:], init_positions)
print('\nInitial portfolio value = $ {}\n'.format(round(init_value, 2)))

# Initial portfolio weights
w_init = (data_prices[0,:] * init_positions) / init_value

# Number of periods, assets, trading days
N_periods = 6*len(np.unique(dates_array[:,0])) # 6 periods per year
N = len(df.columns)-1
N_days = len(dates)

# Annual risk-free rate for years 2020-2021 is 1.5%
r_rf = 0.015
# Annual risk-free rate for years 2008-2009 is 4.5%
r_rf2008_2009 = 0.045
# Annual risk-free rate for year 2022 is 3.75%
r_rf2022 = 0.0375

# Weights of assets in the benchmark portfolio S&P30 for years 2020-2021
w_b = np.array([0.14832533, 0.15556291, 0.01990254, 0.05079846, 0.01302685, 0.01030985, 0.02252249, 0.00227124, ...])
# Weights of assets in the benchmark portfolio S&P30 for years 2008-2009
w_b2008_2009 = np.array([0.04515391, 0.09628167, 0.00962156, 0.04751553, 0.02738386, 0.00612178, 0.03599232, ...])
# Weights of assets in the benchmark portfolio S&P30 for year 2022
w_b2022 = np.array([0.1994311, 0.18518391, 0.05464191, 0.06021769, 0.00822679, 0.01771958, 0.01735487, ...])

# Number of strategies
strategy_functions = ['strat_buy_and_hold', 'strat_equally_weighted', 'strat_min_variance', 'strat_max_return',
                      'strat_max_Sharpe', 'strat_equal_risk_contr', 'strat_lever_max_Sharpe', 'strat_robust_optim',
                      'strat_tracking_index']
strategy_names     = ['Buy and Hold', 'Equally Weighted Portfolio', 'Mininum Variance Portfolio',
                      'Maximum Expected Return Portfolio', 'Maximum Sharpe Ratio Portfolio',
                      'Equal Risk Contributions Portfolio', 'Leveraged Max Sharpe Ratio Portfolio',
                      'Robust Optimization Portfolio', 'Benchmark Tracking Portfolio']

N_strat = 1  # comment this in your code
#N_strat = len(strategy_functions)  # uncomment this in your code
fh_array = [strat_buy_and_hold, strat_equally_weighted, strat_min_variance, strat_max_return, strat_max_Sharpe,
            strat_equal_risk_contr, strat_lever_max_Sharpe, strat_robust_optim, strat_tracking_index]


portf_value = [0] * N_strat
x = np.zeros((N_strat, N_periods),  dtype=np.ndarray)
cash = np.zeros((N_strat, N_periods),  dtype=np.ndarray)
for period in range(1, N_periods+1):
   # Compute current year and month, first and last day of the period

   # Depending on what data/csv (i.e time period) uncomment code
   if dates_array[0, 0] == 20:
```

```
        cur_year  = 20 + math.floor(period/7)
    else:
        cur_year  = 2020 + math.floor(period/7)

    # example for 2008-2009 data
    #if dates_array[0, 0] == 8:
    #    cur_year  = 8 + math.floor(period/7)
    #else:
    #    cur_year  = 2008 + math.floor(period/7)

    cur_month = 2*((period-1)%6) + 1
    day_ind_start = min([i for i, val in enumerate((dates_array[:,0] == cur_year) &
                                                   (dates_array[:,1] == cur_month)) if val])
    day_ind_end = max([i for i, val in enumerate((dates_array[:,0] == cur_year) &
                                                 (dates_array[:,1] == cur_month+1)) if val])
    print('\nPeriod {0}: start date {1}, end date {2}'.format(period, dates[day_ind_start], dates[day_ind_end]))

    # Prices for the current day
    cur_prices = data_prices[day_ind_start,:]

    # Execute portfolio selection strategies
    for strategy  in range(N_strat):

        # Get current portfolio positions
        if period == 1:
            curr_positions = init_positions
            curr_cash = 0
            portf_value[strategy] = np.zeros((N_days, 1))
        else:
            curr_positions = x[strategy, period-2]
            curr_cash = cash[strategy, period-2]

        # Compute strategy
        x[strategy, period-1], cash[strategy, period-1] = fh_array[strategy](curr_positions, curr_cash, mu, Q, cur_prices)

        # Verify that strategy is feasible (you have enough budget to re-balance portfolio)
        # Check that cash account is >= 0
        # Check that we can buy new portfolio subject to transaction costs

        ###################### Insert your code here ############################

        # Compute portfolio value
        p_values = np.dot(data_prices[day_ind_start:day_ind_end+1,:], x[strategy, period-1]) + cash[strategy, period-1]
        portf_value[strategy][day_ind_start:day_ind_end+1] = np.reshape(p_values, (p_values.size,1))
        print('  Strategy "{0}", value begin = $ {1:.2f}, value end = $ {2:.2f}, cash account = ${3:.2f}'.format(
                 strategy_names[strategy], portf_value[strategy][day_ind_start][0],
                 portf_value[strategy][day_ind_end][0], cash[strategy, period-1]))


    # Compute expected returns and covariances for the next period
    cur_returns = data_prices[day_ind_start+1:day_ind_end+1,:] / data_prices[day_ind_start:day_ind_end,:] - 1
    mu = np.mean(cur_returns, axis = 0)
    Q = np.cov(cur_returns.T)
# Plot results
###################### Insert your code here ############################
```

Sample Python Function for Trading Strategy

```
def strat_buy_and_hold(x_init, cash_init, mu, Q, cur_prices):

   x_optimal = x_init
   cash_optimal = cash_init

   return x_optimal, cash_optimal
```