MIE 1622H: Assignment 1 – Mean-Variance Portfolio Selection Strategies

Dr. Oleksandr Romanko, Mohammadreza Mohammadi

January 27, 2025

Due: Saturday, February 15, 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 4 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 February 15.

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 minimizing portfolio variance, maximizing expected return and maximizing Sharpe ratio. You will need to take into account the effect of trading costs.

We start with the classical Markowitz model of portfolio optimization. We are trading only stocks $S_1, \ldots, S_n$, and the total number of stocks is $n = 30$. At holding period t the expected (daily) return of stock i is μ_i^t, the standard deviation of the return is σ_i^t. The correlation between the returns of stocks $i \neq j$ is ρ_{ij}^t. To abbreviate notation we introduce the return vector $\mu^t = (\mu_1^t, \ldots, \mu_n^t)^T$ and the covariance matrix $Q^t = [\rho_{ij}^t \sigma_i^t \sigma_j^t]_{i,j=1,\ldots,n}$. At time period t we estimate μ^t and Q^t from the period $t-1$ time series. There are 12 holding periods and each holding period lasts 2 months (2 years in total). As a result, you can re-balance your portfolio at most 12 times.

Each transaction has a cost composed of a variable portion only. The variable fee is due to the difference between the selling and bidding price of a stock, and is 0.5% of the traded volume. This means that if a stock is quoted at 30 dollars and the investor buys 10 shares, then they are going to pay $0.005 \cdot 30 \cdot 10 = 1.50$ dollars on top of the 300 dollars the 10 shares would cost otherwise. If they sell 20 shares at the same 30 dollar price then they get 600 dollars, minus the $0.005 \cdot 600 = 3$ dollar transaction cost. Your may not need to account for transaction fees in your optimization, but you need to take them into account when computing portfolio value.

The goal of your optimization effort is to create a tool that allows the user to make regular decisions about re-balancing their portfolio and compare different investment strategies. The user wants to consider the total return, the risk and Sharpe ratio. They may want to minimize/maximize any of these components, while limiting one or more of the other components. The basic building block is a decision made at the first trading day of each 2-month holding period: given a current portfolio, the market prices on that day, and the estimates of the mean and covariance of the daily returns, make a decision about what to buy and sell according to a strategy. You are proposed to test five strategies:

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.

To test your bi-monthly decisions, you are given adjusted closing prices of 30 stocks over a two-year period. Obviously, you cannot use the actual future prices when making a decision, only the price at that day, the expected return and covariance estimates. These estimates were derived from the actual data and are quite accurate. Once you are done with trades for a current holding period, you should move on to the next period, use new prices (and new estimates), and make another decision. You also need to track the value of your portfolio on each day, so that you have it handy at the first trading day of a new period. You re-balance your portfolio on the first trading day of each 2-month holding period (up to 12 re-balances during 2 years).

The 30 stocks for this assignment are from U.S. stock market, priced in U.S. dollars. Their quotes are given on trading days as adjusted closing prices. The data set is from January 2020 to December 2021. The list of assets and their sectors is:

1. **Technology**: AAPL, MSFT, NVDA, IBM, AMD, CSCO, HPQ, SONY
2. **Financial Services**: JPM, BAC, MS
3. **Healthcare**: JNJ, UNH, PFE
4. **Consumer Cyclical**: AMZN, F, HOG
5. **Consumer Defensive**: KO, PG
6. **Energy**: XOM, CVX
7. **Real Estate**: AMT, PLD
8. **Communication Services**: GOOG, T, VZ
9. **Industrials**: HON, CAT
10. **Utilities**: NEE, DUK

The initial portfolio is as follows:"HOG" = 3447 shares, "VZ" = 19385 shares.

The value of the initial portfolio is about one million dollars. The initial cash account is zero USD. The cash account must be nonnegative at all times. Your optimization algorithm may suggest buying or selling a non-integer number of shares, which is not allowed. You need to round the number of shares bought or sold to integer values and subtract transaction fees. If the value of your portfolio after re-balancing plus the transaction fees are smaller than the portfolio value before re-balancing, the remaining funds are accumulated in the cash account. Cash account does not pay any interest, but cash funds should be used toward stock purchases when portfolio is re-balanced next time.

An estimate of the daily returns of the stocks (for trading days) and the covariances of the returns are computed for you. These estimates are based on the previous two-month data and are updated every second month. You can only use the current estimate for the calculations and are not allowed to change or update the estimates in any way.

Note that for buying and selling asset shares, you need to optimize over asset weights. Current weight of asset i in a portfolio is $w_i = \frac{v_i \cdot x_i}{V}$, where V is the current portfolio value, v_i is current price of asset i and x_i is the number of units (shares) of asset i in your portfolio. As the initial portfolio value for the 2020-2021 period is \$1,000,016.96 USD, the price of the HOG stock on the first trading day of holding period 1 is 34.08 USD, and we hold 3447 shares, then $w_{12}^1 = \frac{v_{12}^1 \cdot x_{12}^1}{V^1} = \frac{34.08 \cdot 3447}{1000016.96} = 0.1175$.

Questions

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

   You need to test five portfolio re-balancing strategies:

   1. "Buy and hold" strategy: it is the simplest strategy where you hold initial portfolio for the entire investment horizon of 2 years. The strategy is already implemented in the function `strat_buy_and_hold`.
   2. "Equally weighted" (also known as "$1/n$") portfolio strategy: asset weights are selected as $w_i^t = 1/n$, where n is the number of assets. You may need to re-balance your portfolio in each period as the number of shares x_i^t changes even when $w_i^t = 1/n$ stays the same in each period. The strategy should be implemented in the function `strat_equally_weighted`.
   3. "Minimum Variance" portfolio strategy: compute minimum variance portfolio for each period and re-balance accordingly. The strategy should be implemented in the function `strat_min_variance`.
   4. "Maximum Expected Return" portfolio strategy: build a portfolio that maximizes expected return for each period and re-balance accordingly. The strategy should be implemented in the function `strat_max_return`.
   5. "Maximum Sharpe ratio" portfolio strategy: compute a portfolio that maximizes Sharpe ratio for each period and re-balance accordingly. The strategy should be implemented in the function `strat_max_Sharpe`.

   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_optim.ipynb** on the course web-page. You are required to complete the code in the file. Make sure to restart the notebook and re-run all cells on Google Colab before submitting, and have all outputs and plots intact when saving the notebook.

   Your Python code should use only CPLEX optimization solver without CVXPY module. You are required to briefly comment on your code to explain important logic and implementation details. Some marks may be deducted for lack of explanation or poorly commented code.

   You are required to provide a mathematical formulation to the optimization problem you solve in strategy 3, 4, and 5 and explain relevant information (variables, constraints, objective function, data parameters, etc). This should be done in the report and can refer to comments/code from the ipynb file.

   For this assignment you need to use Google Colab and install the trial version of CPLEX (available on Colab) as sizes of the optimization problems are small. To install CPLEX on Google Colab just run `!pip install cplex`.

2. **(25 %) Analyze your results:**

   - Produce the following output for the 12 periods (two years):

```
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 = ...
  ...

Period 12: start date 11/1/2021, 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 = ...
```

- Plot one chart in Python that illustrates the daily value of your portfolio (for each trading strategy) over the two years using daily adjusted closing prices provided. Include the chart in your report.
- Plot three charts in Python for strategy 3, 4, and 5 to show dynamic changes in portfolio allocations. 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. You may use these figures to support your analysis or discussion.
- Compare your trading strategies and discuss their performance relative to each other. Which strategy would you select for managing your own portfolio and why?
- Compute the risk measures (variance, maximum drawdown, and Sharpe ratio) for the "Maximum Expected Return" portfolio strategy over the 2020-2021 time period and compare those to measures for other strategies.

3. **(25 %) Discuss possible improvements to your trading strategies:**

- Test your Python program for different variations of your strategies, e.g., select "1/n" portfolio at the beginning of period 1 and hold it till the end of period 12 (as if the re-balancing strategy required large transaction costs). Discuss if you are able to achieve better results.
- Can you suggest any improvements of the trading strategies that you have implemented?
- To potentially reduce risk measures for the "Maximum Expected Return" portfolio strategy, apply sector diversification constraints by ensuring that the total weight of stocks in each sector is between 0% and 25%. Then, compare your results with those in the previous section.
- Re-test all of your strategies for the 2023-2024 time period and plot the daily value of your portfolio over these two years using the daily adjusted closing prices provided in the new dataset (`adjclose_2023_2024.csv` file). Use the new initial portfolio consisting of "HOG" = 10,904 shares and "CVX" = 3,542 shares, whose total value is again about one million dollars, and apply the updated risk-free rate of 4.5% (instead of 1.5% for 2020-2021). Then, considering the economic contexts of these two distinct two-year periods, compare the performance of all strategies and clearly discuss how market conditions may have influenced performances, rewards and risks of your portfolios during 2020–2021 versus 2023–2024 time periods.
- **(Optional Question)** Using a generative AI tool (e.g., ChatGPT), produce a one-page, executive-style summary of your portfolio modeling results and analyses. Include this AI-generated summary in your report and briefly discuss its clarity, accuracy, and any important points it may have missed relative to your own written conclusions. You can get additional 5 % for answering this optional question, but your maximal mark for the assignment cannot exceed 16 points.

Python Code to be Completed (available on Quercus)

```
# Import libraries
import pandas as pd
import numpy as np
import math

"""# Strategies
 For strategies 3,4 and 5 include mathematical formulation of optimization problem in report and/or in notebook
"""

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_max_return(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_Sharpe(x_init, cash_init, mu, Q, cur_prices):

   return x_optimal, cash_optimal

"""# Data loading and initialization of portfolios"""

# Input file
input_file_prices='adjclose_2020_2021.csv' # path to close_2020_2021 file

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

# 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
if '2021' in input_file_prices:
    day_ind_end0 = len(np.where(dates_array[:,0]==2019)[0])
elif '2022' in input_file_prices:
    day_ind_end0 = len(np.where(dates_array[:,0]==2020)[0])

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, 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

# Number of strategies
strategy_functions = ['strat_buy_and_hold', 'strat_equally_weighted', 'strat_min_variance',
                      'strat_max_return', 'strat_max_Sharpe']
strategy_names     = ['Buy and Hold', 'Equally Weighted Portfolio', 'Minimum Variance Portfolio',
                      'Maximum Expected Return Portfolio', 'Maximum Sharpe Ratio 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]

"""# Running computations and printing results"""

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
   if '2020' in input_file_prices:
      if dates_array[0, 0] == 20:
         cur_year  = 20 + math.floor(period/7)
      else:
         cur_year  = 2020 + math.floor(period/7)

   elif '2021' in input_file_prices:
      if dates_array[0, 0] == 21:
         cur_year  = 21 + math.floor(period/7)
      else:
         cur_year  = 2021 + 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"""

# Plot results
###################### Insert your code here ############################

"""# Repeat for years 2023-2024"""

# Input file
input_file_prices_new='adjclose_2023_2024.csv' # path to close_2023_2024 file

# Read data into a dataframe
df_new = pd.read_csv(input_file_prices_new)

# Initial positions in the portfolio
init_positions_new = np.array([0, 0, 0, 0, 0, 0, 0, 3542, 0, 0, 0, 10904, 0, 0,
                               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

# Annual risk-free rate for years 2023-2024 is 4.5%
r_rf2023_2024 = 0.045

# Repeate the previous calculation for the time period 2023-2024
###################### Insert your code here ############################

"""# Plot the results for years 2023-2024"""

# Plot results for 2023-2024
###################### 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
```