香 港 大 學
THE UNIVERSITY OF HONG KONG

STAT6013 Financial Data Analysis Project

Beyond Markowitz: Alternative Portfolio Optimization Methods

WU Kwan-yu

(UID: 3035275501)

Supervisor

Dr. Eric Li

DEPARTMENT OF STATISTICS AND ACTUARIAL SCIENCE

December 2020

Abstract

This is an abstract.

Contents

Beyond Markowitz: Alternative Portfolio Optimization Methods

WU Kwan-yu

December 2020

Abstract

The core of the modern portfolio theory is based on the ground-breaking model proposed by Harry Markowitz in 1952. The idea of the model is to solve objectives involving trade-offs between mean return and volatility. However, the model itself is less applicable due to the lack of consideration of real-world scenarios. This project introduces other types of portfolio optimization methods in recent decades and a comparison of performance with backtesting under US stocks is to be conducted.

1 Introduction

1.1 Background

In 1952, economist Harry Markowitz proposed a mean-variance optimization portfolio Markowitz, 1959 model to establish an efficient strategy for assigning weights towards a portfolio of assets with minimized risk with a specified return. The proposal becomes the core idea of modern portfolio theory with its elegant and simple closed-form solution towards the problem. However, the proposed method is rather theoretical, and the business goals and scenarios are remain unconsidered, which makes the original model inapplicable in modern real-world scenario. With the advancement of technology and computational power, more researches have been made to extend Markowitz's portfolio theory with employment of computational algorithms or machine learning techniques, particularly on weight optimization approaches with various constraints and objectives (ie. cost functions) to tackle these problems.

In this paper, two alternative approaches on determining optimal weight of portfolio will be introduced, namely the *Equal Risk Contribution portfolio (ERC)* proposed by Maillard et al., 2010, and *Most Diversified Portfolio (MDP)* proposed by Choueifaty and Coignard, 2008. The two portfolio are driven by different proposed cost function that are considered under various business strategic scenarios. We will also apply the optimization approaches towards the recent US stock market through backtesting on the US stocks with largest current market capital.

Approach	Cost Functions
Mean-variance portfolio	$\frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{x}'\boldsymbol{\mu}$
Equal-weighted portfolio	$x_i = \frac{1}{N}$ where N is number of assets
Global minimum variance portfolio	$\frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x}$
Equal risk contribution portfolio	$\frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} - \ln \boldsymbol{x_i}$
Most diversified portfolio	$-(\ln \boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} - \ln \boldsymbol{x}'\boldsymbol{\sigma})$

Table 1: Portfolio construction approaches and their respective cost functions

1.2 Project Outline

This project report is outlined as follows: The details of the traditional and recent approaches of optimization methods will be introduced in Section 2.1, followed by the application method introducing in 2.2. The specification of data is given in Section 3 and the empirical results of backtesting is given and discussed in Section 4. The source code for this project, written in Python, is provided in Appendices.

2 Methodologies

2.1 Various Portfolio Optimization Models

In this section, the traditional approaches of portfolio allocation methods are given, followed by the two alternative optimization methods. The major differences in these methods are on their respective cost functions as given from Table 1:

- discuss their objective function, approximation method (`scipy SLSQP`) - mention that some of them does not have closed form solution; approximation method suggested by Perrin et al. (2020) using iteration method - discuss their business interpretation, business use case

2.1.1 Markowitz Mean-variance Portfolio (MVO)

The mean-variance portfolio proposed by Markowitz is an optimization problem on minimizing the portfolio risk with mean return penalty:

$$\frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{x}'\boldsymbol{\mu} \text{ subject to } \boldsymbol{x}'\mathbf{1} = 1$$

It is often the case that no short-selling constraint is applied towards the optimal weighting $\boldsymbol{x}$, where

$$x_i \geq 0$$

The optimization provides an elegant closed-form solution, which can be derived using Lagrange multiplier method. Without considering the risk-free rate as in this project, the optimal weight is given by

$$\boldsymbol{x} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \text{ with normalization}$$

Alternatively to avoid edge cases, the optimal weighting can be found using Python package `scipy`, with an iterative estimation method of locating local-minima using Sequential Least Squares Programming (SLSQP).

2.1.2 Equal-weighted Portfolio

The equal-weighted portfolio serves as a base benchmark model in evaluation of the performance of other approaches. Disregarding the market capital of assets, the equal-weighted portfolio allocates equal portfolio weights to all assets under the specified universe. In this project, 20 assets are chosen and hence $x_i = 0.05$ is set for equal-weighted portfolio.

2.1.3 Global Minimum Variance Portfolio (GMV)

Similar to MVO, the portfolio seeks for optimum weighting such that the least portfolio risk of all weighting combination is achieved. The optimization problem is to minimize the portfolio risk:

$$\frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} \text{ subject to } \boldsymbol{x}'\boldsymbol{1} = 1$$

It is often the case that no short-selling constraint is applied towards the optimal weighting $\boldsymbol{x}$, where

$$x_i \geq 0$$

The closed-form solution can be easily derived using Langrange multiplier method, where

$$\boldsymbol{x} = \boldsymbol{\Sigma}^{-1}\boldsymbol{1} \text{ with normalization}$$

With a simple closed-form solution, the optimal weighting can be calculated easily without any iterative estimation methods.

2.1.4 Equal Risk Contribution Portfolio (ERC)

The objective of ERC is to obtain a portfolio weighting such that the level of risk for every assets under the portfolio are the same. The optimization problem is to minimize the cost function

$$\frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} - \ln \boldsymbol{x_i} \text{ subject to } \boldsymbol{x}'\boldsymbol{1} = 1$$

In the scope of this project, the no-short-selling constraint is applied in order to provide consistent assumptions with MVO.

The model does not provide a closed-form solution for the optimization problem. The optimal weighting can be found using `scipy` with SLSQP, similar to MVO.

2.1.5 Maximum Diversification Portfolio (MDP)

The objective of MDP is to obtain a portfolio weighting such that the diversification ratio DR is maximized. The diversification ratio is given by

$$\text{DR} = \frac{x'\sigma}{\sqrt{x'\Sigma x}}$$

and the cost function is given by

$$-(\ln \boldsymbol{x}'\boldsymbol{\Sigma}\boldsymbol{x} - \ln \boldsymbol{x}'\boldsymbol{\sigma})$$

In the scope of this project, the no-short-selling constraint is applied in order to provide consistent assumptions with MVO.

The model does not provide a closed-form solution for the optimization problem. The optimal weighting can be found using `scipy` with SLSQP, similar to MVO and ERC.

2.2 Backtesting Methodology

A simple backtesting is to be carried out in order to evaluate the performance of various portfolio construction approaches and compare the performance metrics. In this project, a backtesting engine is constructed using Python with the following configurations:

- Universe: The 20 largest US stocks ranked in market capital and traded in NASDAQ exchange is chosen. The 20 tickers are namely `['AAPL', 'MSFT', 'AMZN', 'GOOG', 'GOOGL', 'FB', 'TSLA', 'NVDA', 'PYPL', 'ADBE', 'CMCSA', 'NFLX', 'PEP', 'INTC', 'CSCO', 'AVGO', 'QCOM', 'TMUS', 'COST', 'TXN']`.

- Time period: The time frame of observation starts from January 2016 to December 2020. The backtesting period starts from January 2017, where the period January 2016 to December 2016 is used for observation only.
- Portfolio re-balancing frequency: Each of the portfolio except for equal-weighted portfolio are re-balanced in the beginning of every 1 month starting from the backtesting period January 2017.
- Rolling window: A rolling window of 1 year is considered. At each time point of re-balance, the portfolio weighting is determined by the recent 1 year of stock data only.
- Empirical estimation: The estimation of the moments of stock returns (mean return and volatility) for each stock are estimated assuming uni-variate normality. The mean returns and volatility are calculated using simple mean and standard deviation.
- Performance metrics: The generation of performance metrics are with aid of the Python package `pyfolio`. The annualized mean, volatility, and Sharpe ratio are the main metrics in investigation.

With the above backtesting configurations, the daily stock returns are calculated using different portfolio weightings. An index for each portfolio construction approaches for each day is then generated. The result of backtesting will be presented in Section 4.

3 Data

The information of top US stocks is obtained on NASDAQ Stock Screener. The daily historical price data from January 2016 to December 2020 of the selected stocks are obtained from Yahoo Finance through Python package `yfinance`. The close price of the stock is used to represent the daily price. With the daily historical price data, the daily log-return is calculated. There are 1255 time points in total. For a rare occasion of missing values of price data, the missing data are filled with zero in log-return. The data are not aggregated as the frequency of data matches the backtesting needs.

4 Empirical Results and Analysis

4.1 Backtesting Results

The backtesting results are shown in Figure 1 and the performance metrics are shown in Table 2.

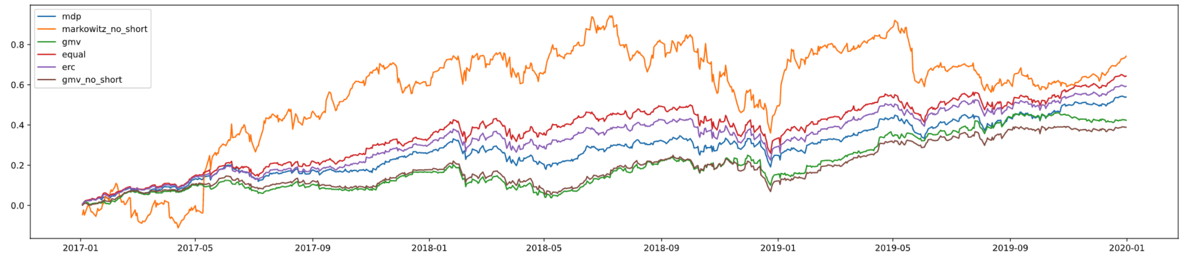

Figure 1: Return plot for each portfolio construction approaches

	MVO	Equal-weighted	GMV	ERC	MDP
Annual return	18.8%	22.0%	14.2%	20.3%	18.5%
Cumulative returns	67.4%	81.1%	48.6%	73.9%	66.0%
Annual volatility	38.9%	18.2%	13.4%	16.1%	14.6%
Sharpe ratio	0.64	1.18	1.06	1.23	1.23
Maximum drawdown	-47.3%	-22.3%	-15.6%	-19.5%	-14.9%
Skew	-0.10	-0.54	-0.54	-0.56	-0.52
Kurtosis	4.65	3.33	4.01	3.50	2.76

Table 2: Performance metrics

4.2 Analysis

It can be observed that the results of backtesting performance metrics are generally consistent with the underlying basis of portfolio construction approaches. As shown in Figure 2, GMV portfolio exhibits the lowest annual volatility, yet with the lowest annual return of all five portfolio construction approaches. This shows that GMV provides a low risk approach of portfolio construction, with a trade-off of low return. In contrast, MVO portfolio has a remarkably higher annual volatility and maximum drawdown comparing with the other four portfolio, and a similar annual return with equal-weighted, ERC, and MDP portfolio. Under a similar level of annual return with these portfolio, MVO has a lower Sharpe ratio which reflects that MVO might not be a robust nor good choice of portfolio construction approach. On the other hand, ERC and MDP provides a similar performance result. In fact, the differences in actual portfolio weighting allocation is not significantly different between the two approaches. Both approaches aims at constructing portfolio with consideration of business risk management.

In terms of annual return and cumulative returns, despite its naive method of construction, the equal-weighted portfolio out-performs the other four portfolios with a slightly higher annual volatility. This is actually reasonable, given that the other four approaches may not be robust, and practical constraints such as turnover, transaction fees, short-selling availability, asset class limits, and leverage limits. Furthermore, the pre-selection method in this project also benefits the equal-weighted portfolio as it can be seen that the overall US stock market exhibits a continuous bullish trend with only small downsides, and the mega stocks with top market capital values are surely the stock companies that a benefits from it.

5 Conclusion

In this project, we have introduced some recent extensions from Markowitz's optimal portfolio model, namely Equal Risk Contribution portfolio and Most Diversified Portfolio. In order to compare the usability of the two newer approaches, a backtesting engine on top US stocks from 2016 to 2020 is constructed and a simple backtesting is conducted to examine the recent performances of various portfolio construction approaches. The performance metrics shows a reasonable, yet not significantly well-performing result of backtesting under these portfolio.

References

Choueifaty, Y., & Coignard, Y. (2008). Toward maximum diversification. *The Journal of Portfolio Management*, *35*(1), 40–51.

Jagannathan, R., & Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraints helps. *The Journal of Finance*, *58*(4), 1651–1683.

Maillard, S., Roncalli, T., & Teıletche, J. (2010). The properties of equally weighted risk contribution portfolios. *The Journal of Portfolio Management*, *36*(4), 60–70.

Markowitz, H. (1959). Portfolio selection. *Investment under Uncertainty*.

Perrin, S., & Roncalli, T. (2020). Machine learning optimization algorithms & portfolio allocation. *Machine Learning for Asset Management: New Developments and Financial Applications*, 261–328.

Richard, J.-C., & Roncalli, T. (2019). Constrained risk budgeting portfolios: Theory, algorithms, applications & puzzles. *arXiv preprint arXiv:1902.05710*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature methods*, *17*(3), 261–272.

Appendices

A Source Codes for the Project

```
import pandas as pd
import numpy as np
import yfinance as yf
from datetime import datetime
from dateutil.relativedelta import relativedelta
from tqdm import tqdm
from collections import OrderedDict
from scipy.optimize import minimize
import matplotlib.pyplot as plt
import pyfolio as pf


TOPN = 20
OBS_START = 'datetime(2016,1,1)' # string micmicking datetime object
BACKTEST_START = datetime(2017,1,1) # datetime
BACKTEST_END = '2019-12-31' # string


def empirical_estimate(data, tickers, start, end):
    start_dt = f'datetime({start[:4]},{int(start[5:7])},{int(start[8:10])})'
    end_dt = f'datetime({end[:4]},{int(end[5:7])},{int(end[8:10])})'
    ret = []
    for x, t in enumerate(tickers):
        df = data[t]
        df = df.query(f'Date␣>=␣{start_dt}␣and␣Date␣<=␣{end_dt}')
        ret.append(np.array((df['Logret']*100).fillna(0).to_list()))
    mean = np.matrix([sum(stock_ret) for stock_ret in ret]).transpose()
    cov = np.matrix(np.cov(ret))
    return ret, {'mean': mean, 'cov': cov}


def col_mat_to_list(z):
    return np.array((z/z.sum()).transpose())[0]


def gmv_portfolio(stat):
    covar = stat['cov']
    cov_inv = np.linalg.inv(covar)
    I = np.matrix(np.ones((len(tickers),1)))
    weights = cov_inv * I
    weights = col_mat_to_list(weights)
    return weights


def equal_portfolio(stat):
    return [1/len(tickers)]*len(tickers)
```

```
def gmv_no_short_portfolio(stat):
    covar = stat['cov']
    bnd = [(0, 1)]*len(tickers) # only positive weights
    w0 = [1/len(tickers)]*len(tickers)

    # min var optimization
    def calculate_portfolio_var(w,covar):
        w = np.matrix(w)
        return (w*covar*w.T)[0,0]

    cons = ({'type': 'eq', 'fun': lambda x:  np.sum(x)-1.0})
    weights = minimize(calculate_portfolio_var,
        w0, args=covar, bounds=bnd, method='SLSQP',constraints=cons)['x']
    return weights


def markowitz_no_short_portfolio(stat):
    mean = stat['mean']
    covar = stat['cov']

    w0 = [1/len(tickers)]*len(tickers)

    def calculate_cost(w, pars):
        mean = pars[0]
        covar = pars[1]
        w = np.matrix(w)
        return ((np.sqrt(w*covar*w.T)[0,0])*0.5 - (w*mean)[0,0])

    bnd = [(0, 1)]*len(tickers) # only positive weights
    cons = ({'type': 'eq', 'fun': lambda x:  np.sum(x)-1.0})
    weights = minimize(calculate_cost, w0, args=[mean,covar],
        bounds=bnd, method='SLSQP',constraints=cons)
    return weights['x']


def erc_portfolio(stat):
    covar = stat['cov']

    def calculate_portfolio_var(w,V):
        w = np.matrix(w)
        return (w*V*w.T)[0,0]

    def calculate_risk_contribution(w,V):
        w = np.matrix(w)
        sigma = np.sqrt(calculate_portfolio_var(w,V))
        MRC = V*w.T
        RC = np.multiply(MRC,w.T)/sigma
        return RC

```

```
    def risk_budget_objective(x,pars):
        V = pars[0]
        x_t = pars[1]
        sig_p =  np.sqrt(calculate_portfolio_var(x,V)) # portfolio sigma
        risk_target = np.asmatrix(np.multiply(sig_p,x_t))
        asset_RC = calculate_risk_contribution(x,V)
        J = sum(np.square(asset_RC-risk_target.T))[0,0] # sum of squared error
        return J

    def total_weight_constraint(x):
        return np.sum(x)-1.0

    def long_only_constraint(x):
        return x

    w0 = [1/len(tickers)]*len(tickers)
    x_t = [1/len(tickers)]*len(tickers)
    cons = ({'type': 'eq', 'fun': total_weight_constraint},
     {'type': 'ineq', 'fun': long_only_constraint})
    weights = minimize(risk_budget_objective, w0,
        args=[covar,x_t], method='SLSQP',constraints=cons)
    return weights['x']


def mdp_portfolio(stat):
    covar = stat['cov']
    def calculate_portfolio_var(w,V):
        w = np.matrix(w)
        return (w*V*w.T)[0,0]

    def calc_diversification_ratio(w, covar):
        w_vol = np.dot(np.sqrt(np.diag(covar)), w.T)
        port_vol = np.sqrt(calculate_portfolio_var(w, covar))
        diversification_ratio = w_vol/port_vol
        return -diversification_ratio

    def total_weight_constraint(x):
        return np.sum(x)-1.0

    def long_only_constraint(x):
        return x

    w0 = [1/len(tickers)]*len(tickers)
    cons = ({'type': 'eq', 'fun': total_weight_constraint},
        {'type': 'ineq', 'fun': long_only_constraint})
    weights = minimize(calc_diversification_ratio, w0,
        args=covar, method='SLSQP', constraints=cons)
    return weights['x']
```

```
def weight_construction_portfolio(stat, mode):
    """
    specify method of weight_construction
    choices:
    gmv, markowitz, equal, gmv_no_short, markowitz_no_short, erc
    """
    if mode == 'gmv':
        return gmv_portfolio(stat)
    if mode == 'equal':
        return equal_portfolio(stat)
    if mode == 'gmv_no_short':
        return gmv_no_short_portfolio(stat)
    if mode == 'markowitz_no_short':
        return markowitz_no_short_portfolio(stat)
    if mode == 'erc':
        return erc_portfolio(stat)
    if mode == 'mdp':
        return mdp_portfolio(stat)


def backtest(mode):
    for m in mode:
        weights_backtest = {}
        for i in tqdm(range(37)): # 37 for 2019-01-01
            observation_start_date = datetime.strftime(
                BACKTEST_START+relativedelta(months=i)-relativedelta(years=1)
                    ,
                '%Y-%m-%d')
            rebalance_date = datetime.strftime(
                BACKTEST_START+relativedelta(months=i), '%Y-%m-%d')
            _, stat = empirical_estimate(data,
                tickers,
                observation_start_date,
                rebalance_date)
            weights_backtest[rebalance_date] = weight_construction_portfolio(
                stat, m)
            # print(weights_backtest[rebalance_date])

        datemap = OrderedDict(weights_backtest)
        # print(datemap)
        # get all trading dates
        df_backtest = pd.DataFrame({'Date': list(data['MSFT'].index)})
        df_backtest = df_backtest[df_backtest['Date']>=datetime.strftime(
                BACKTEST_START,'%Y-%m-%d')]
        df_backtest = df_backtest[df_backtest['Date']<=BACKTEST_END]
        df_backtest['Date'] = df_backtest['Date'].apply(
                lambda x: datetime.strftime(x,'%Y-%m-%d'))

        def find_weight(dt):
            result = None
            for dm, w in datemap.items():
```

```
                if dt >= dm:
                    result = w
                if dt < dm:
                    return result
            else:
                return result

        df_backtest['Weight'] = df_backtest['Date'].apply(find_weight)
        df_backtest['Logret'] = df_backtest['Date'].apply(
                lambda x: [data[t].loc[x,'Logret'] for t in tickers])
        df_backtest['Daily_Logret'] = df_backtest.apply(
                lambda x: sum(r*w for r,w in zip(x['Weight'],x['Logret'])) , axis=1)

        # Benchmark viz
        df_backtest['Date_dt'] = df_backtest['Date'].apply(
                lambda x: datetime.strptime(x, '%Y-%m-%d'))
        plt.plot(df_backtest['Date_dt'],df_backtest['Daily_Logret'].cumsum(), label=m)

        #Performance metrics
        if len(mode) == 1:
            df_metrics = df_backtest[['Date_dt','Daily_Logret']]
            df_metrics = df_metrics.set_index('Date_dt')
            pf.create_simple_tear_sheet(df_metrics['Daily_Logret'])
    plt.legend()
    plt.show()


if __name__ == "__main__":

    # obtain data and preprocessing
    nasdaq_data = pd.read_csv('./data/nasdaq_stocks.csv')
    mc = {row['Symbol']: row['Market␣Cap'] for _, row in nasdaq_data.iterrows() \
     if row['Market␣Cap'] != 0 and not np.isnan(row['Market␣Cap'])}
    mcdf = pd.DataFrame.from_dict(mc, orient='index', columns=['marketcap'])
    mcdf_chosen = mcdf.nlargest(TOPN,'marketcap', keep='first')
    tickers = mcdf_chosen.index.to_list()
    data = yf.download(
        tickers='␣'.join(tickers) ,
        period='max',
        interval='1d',
        threads=True,
        group_by = 'ticker')
    data = {t:data[t] for t in tickers}

    for v in data.values():
        v['Return'] = v['Close']/v['Close'].shift(1)
        v['Logret'] = v['Return'].apply(lambda x: np.log(x))
    data = {k:v.query(f'Date␣>=␣{OBS_START}') for k,v in data.items()}
```

```

    # generate weights and backtesting
    backtest(mode=['mdp','markowitz_no_short','gmv','equal','erc','
        gmv_no_short'])
```