CONSTRUCTING INDEX FUND TO TRACK THE NASDAQ-100

AN OPTIMIZATION APPROACH

GROUP 15

GROUP MEMBERS:

Akshay Navaneetha Krishnan (an34244)
Monica Liou (cl49358)
Vaibhav Nagar (vn5339)
Ayaan Khan (ak46396)

DATE:

Oct 22, 2023

Table of Contents

1. Introduction....3
2. Background....3
3. Scope of the Report....4
4. Objective....4
5. Methodology
    - 5.1. Data Preparation and Preprocessing....5
    - 5.2. Method-1
        - 5.2.1. Stock Selection....6
        - 5.2.2. Calculating Portfolio Weights....7
    - 5.3. Method-2....9
    - 5.4. Performance Evaluation....10
6. Specifics....11
7. Summary and Recommendations....12

1. Introduction

The report provides an in-depth analysis of creating an index fund aimed at replicating the performance of the NASDAQ-100 index. Through two distinct methodologies, the study seeks to optimize factors such as stock selection and portfolio weightings, thus offering a streamlined and cost-effective investment solution.

Rationale

Given the surge in interest for passive investment vehicles, there is a growing need for cost-efficient index funds that accurately mirror market indices. This report aims to fill that gap by exploring methodologies for stock selection and portfolio weighting.

2. Background

The equity money management strategies are classified as active or passive. Index funds are a popular form of passive investing, designed to replicate the performance of specific market indices.

Importance of Index Funds

Index funds serve as an exemplary vehicle for passive investment, offering broad market exposure with lower costs and risk as compared to active funds.

Classifying Money Management Strategies

Equity money management can be broadly categorized into active and passive strategies. While active strategies aim to outperform the market, passive strategies like index funds seek to replicate market performance.

3. Scope of the Report

This report will cover:

Stock Selection Strategies:

- Criteria for stock selection
- Data sources and collection methods

Portfolio Weighting Methodologies:

- Market-cap weighting vs. equal weighting
- Pros and cons of different weighing mechanisms

Comparative Performance Evaluation:

- Performance metrics for Method-1 and Method-2
- Risk-adjusted returns

4. Objective

Cost-Effectiveness:
To offer investors a low-cost option for tracking the NASDAQ-100 index.

Efficiency:
To design an index fund that uses a reduced set of 'm' stocks but closely mimics the performance of the NASDAQ-100 index.

Comparative Analysis:
To compare and contrast the efficacy of two distinct methodologies in achieving the above objectives.

5. Methodology

5.1 Data Preparation and Preprocessing

Data Sources:
The data for this analysis has been sourced from two CSV files, containing stock and index data for the years 2019 and 2020.

Data Importing:
The Pandas library in Python is used for importing the CSV files. The date column is parsed to a datetime format for ease of manipulation.

Feature Extraction:
Only the columns representing stock prices are used for analysis. The stock names are extracted and stored in separate lists for both years.

Calculating Returns:
The percentage change function is applied to stock prices to calculate daily returns. The resulting data is then filtered to remove any 'NaN' values (first row of data).

Calculating Correlations:

Correlation matrices are then generated to understand the correlation between the returns of different stocks for both 2019 and 2020.

Code snippet below:

```
combined_filtered=data_2019.iloc[:,2:].pct_change().dropna().reset_index(drop=True)
combined_filtered.columns = [col + '_returns' for col in combined_filtered.columns]
index_returns=data_2019.iloc[:,1].pct_change().dropna().reset_index(drop=True)

corr_matrix = combined_filtered.corr()
corr_list = corr_matrix.values.tolist()
```

Pct_change for calculating returns and then correlations

5.2 Method-1

This method consists of two primary steps, Stock Selection and Calculating Portfolio Weights, which are described in detail below.

5.2.1 Stock Selection

- Formulated a Binary Programming problem using Gurobi to select stocks.
- Decision variables: `best_rep` and `selected_stock`.
- Objective: **Maximize** the **correlation** among selected stocks.
- Constraints: Number of stocks (`m`) and a representative for each stock.

$$\max_{x,y} \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{n} y_j = m.$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad for\ i = 1,2,\dots,n$$

$$x_{ij} \leq y_j \qquad for\ i,j = 1,2,\dots,n$$

$$x_{ij}, y_j \in \{0,1\}$$

Code snippet below:

```
# Initializing Stock Selection Model
sf19=gp.Model()
# Decision variable that represents the best representative stock out of the selected stocks
best_rep = sf19.addMVar((number_stocks,number_stocks),vtype="B")
# Decision variable that represents whether the stock was selected for the fund
selected_stock= sf19.addMVar((number_stocks),vtype="B")
# Objective function to maximizing the correlation
sf19.setObjective(gp.quicksum(corr_list[i][j]*best_rep[i][j] for i in range(number_stocks) for j in \
                  range(number_stocks)),sense=gp.GRB.MAXIMIZE)
# Constraint to only select m stocks
sf19.addConstr(gp.quicksum(selected_stock[i] for i in range(number_stocks))==m)
# Constraint to make sure every stock has a representative stock
sf19.addConstrs(gp.quicksum(best_rep[i][j] for j in range(number_stocks))==1 for i in range(number_stocks))
# Constraint to make sure the representative stock is also a selected stock
sf19.addConstrs(best_rep[i][j]<=selected_stock[j] for i in range(number_stocks) for j in \
                  range(number_stocks))
sf19.setParam('OutputFlag', 0)
sf19.optimize()
# List containing the selected stocks after optimizing the model
selected_list=[int(x) for x in selected_stock.x]

# Returns of the stocks that were selected above for 2019
selected_columns = combined_filtered.loc[:, [i==1 for i in selected_list]]
```

Decision variable, Objective function, constraints and "m" stocks selected

5.2.2 Calculating Portfolio Weights

Initialize another optimization model to assign weights to selected stocks.

- Decision variable: `w` (weights).
- Objective: **Minimize the absolute return difference** between the portfolio and the index.
- Constraint: Sum of weights should be 1.

$$\min_{w} \sum_{t=1}^{T} \left| q_t - \sum_{i=1}^{m} w_i r_{it} \right|$$

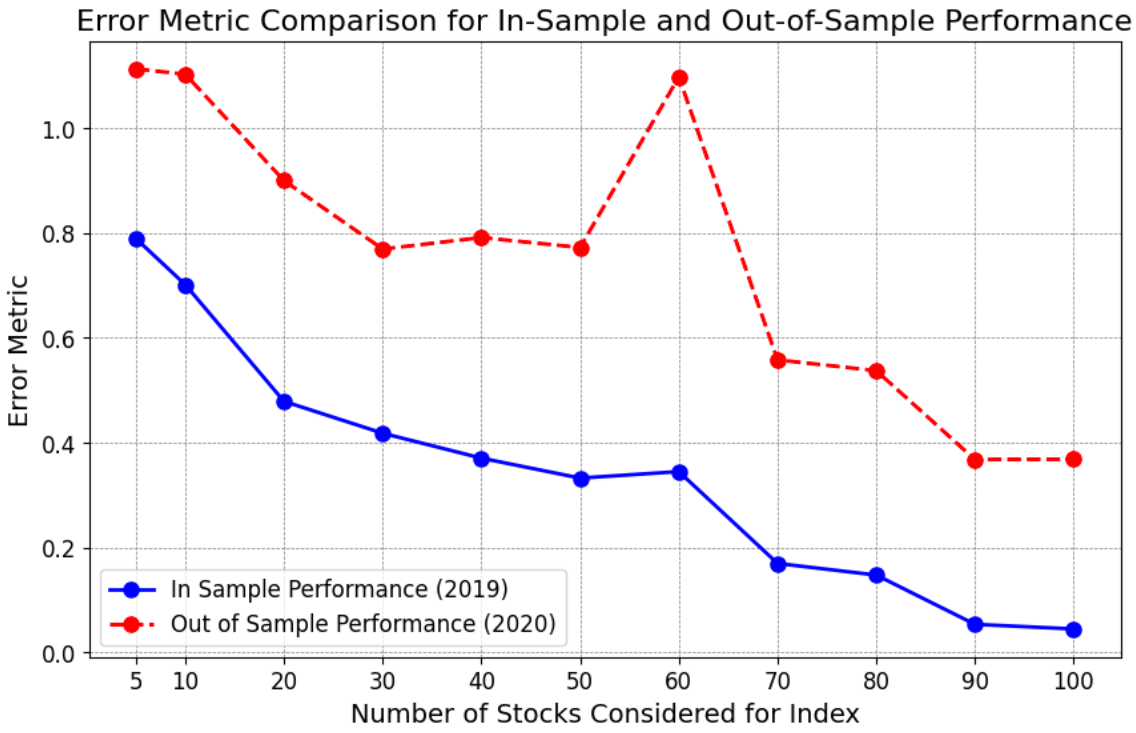
$$s.t. \sum_{i=1}^{m} w_i = 1$$

$$w_i \geq 0.$$

Code snippet below:

```
# Initializing the model to select the weights for the above selected stocks
weights=gp.Model()
# Decision variable that represents the weight of each stock
w = weights.addMVar((sum(selected_list)),vtype="C")
# Proxy variable to represent absolute values in the objective function
bounds = weights.addMVar(len(selected_columns),vtype="C")
#Objective function to minimize the daily absolute return difference between fund and index
weights.setObjective(gp.quicksum(bounds[i] for i in range(len(selected_columns))))
# Constraint to make the sum of weights to be 1
weights.addConstr(gp.quicksum(w[i]  for i in range(sum(selected_list)))==1)
# Adding the lower and upper bounds to imitate an absolute difference
weights.addConstrs(bounds[j]>=(index_returns[j]-gp.quicksum(selected_columns.iloc[j,i]*w[i] for i\
                           in range(sum(selected_list)))) for j in range(len(selected_columns)))
weights.addConstrs(bounds[j]>=-1*(index_returns[j]-gp.quicksum(selected_columns.iloc[j,i]*w[i] for i\
                           in range(sum(selected_list)))) for j in range(len(selected_columns)))
weights.setParam('OutputFlag', 0)
weights.optimize()
# Returns of the stocks that were selected above for 2020
selected_columns_2020=combined_filtered_2020[[x+"_returns" for x in stocks_list[[i==1 for i in selected_list]]]
```

Decision variable, Objective function, constraints and "w" weights



Observation: The chart illustrates the error metric comparison between in-sample performance from 2019 and out-of-sample performance from 2020. As the number of stocks considered for the index increases, the out-of-sample performance peaks around 70 stocks, while the in-sample error gradually decreases.

5.3 Method-2

In Method-2, the strategy emphasizes selecting an optimal combination of stocks dynamically and then determining their weights to create a customized fund that mirrors a predefined index. Here's a concise summary:

- **Stock Selection:** For each iteration, corresponding to a different number of stocks, the method doesn't begin with a predetermined stock selection but instead, decides on which stocks to include based on optimization results.
- **Optimization Model:** The model consists of decision variables: weights for each stock (`w`), an absolute difference proxy (`bounds`), and a binary variable indicating if a stock is utilized (`utilized`).
- **Objective:** To minimize the absolute daily return difference between the custom fund and the index.
- **Constraints are added to ensure:**
  - Total weights sum to 1.
  - A certain number of stocks (`m`) are selected.
  - If a stock has non-zero weight, it's marked as utilized.

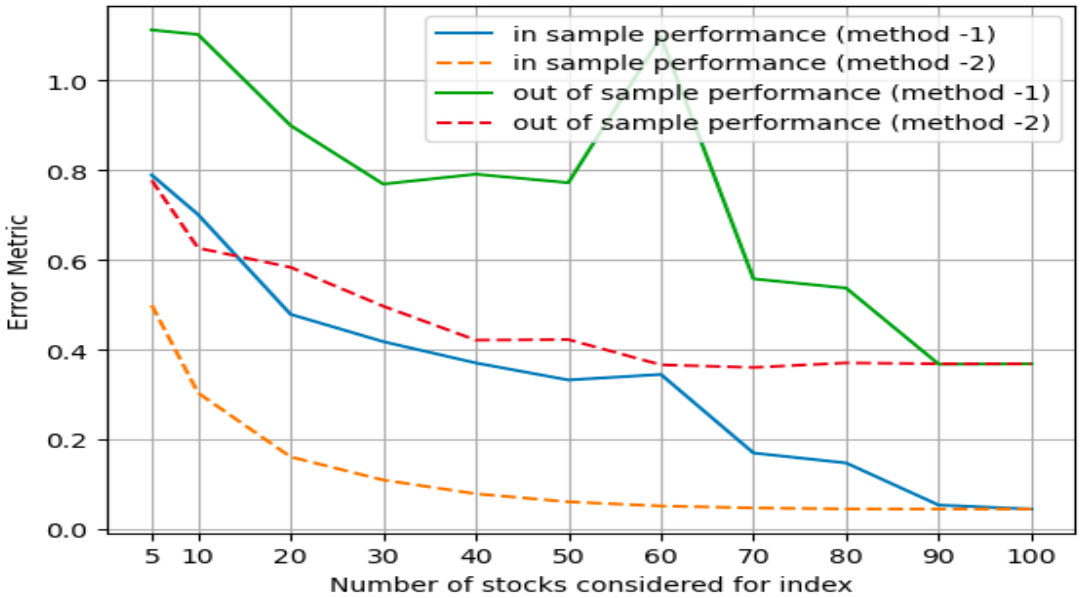
Code snippet below:

```
#List with all stocks (we don't select the stocks initially)
selected_list=[1 for x in range(len(combined_filtered.columns))]

selected_columns = combined_filtered.loc[:, [i==1 for i in selected_list]]

#Initializing the model to both select the weights and pick the stocks
weights=gp.Model()
# Decision variable that represents the weight of each stock
w = weights.addMVar((sum(selected_list)),vtype="C")
# Proxy variable to represent absolute values in the objective function
bounds = weights.addMVar(len(selected_columns),vtype="C")
# Variable that represents whether the stock was used in the fund or not
utilized=weights.addMVar((sum(selected_list)),vtype="B")

#Objective function to minimize the daily absolute return difference between fund and index
weights.setObjective(gp.quicksum(bounds[i] for i in range(len(selected_columns))))
# Big M constraint with M=1, to ensure that if a stock has non zero weight, it is utilized
weights.addConstrs(utilized[i]>=w[i] for i in range(sum(selected_list)))
# Constraint to make the sum of weights to be 1
weights.addConstr(gp.quicksum(w[i]  for i in range(sum(selected_list)))==1)
# Constraint to ensure than m stocks are selected
weights.addConstr(gp.quicksum(utilized[i]  for i in range(sum(selected_list)))==m)
# Adding the lower and upper bounds to imitate an absolute difference
weights.addConstrs(bounds[j]>=(index_returns[j]-gp.quicksum(selected_columns.iloc[j,i]*w[i] for i\
                           in range(sum(selected_list))) for j in range(len(selected_columns)))
weights.addConstrs(bounds[j]>=-1*(index_returns[j]-gp.quicksum(selected_columns.iloc[j,i]*w[i] for i\
                           in range(sum(selected_list))) for j in range(len(selected_columns)))
weights.setParam('OutputFlag', 0)
weights.setParam('TimeLimit', time_limit)
weights.optimize()
weights.objVal

# Returns of the stocks that were selected above for 2020
selected_columns_2020=combined_filtered_2020[[x+"_returns" for x in stocks_list[[i==1 for i in selected_list]]]]
```

Decision variable, Objective function, Big M constraint (M=1) and "w" weights

5.3 Performance Evaluation

Post-optimization, the returns for the chosen stocks in the subsequent year (2020) are computed. The returns for the created fund and the error metrics for both 2019 (in-sample) and 2020 (out-of-sample) are subsequently determined.



Observation: The chart above illustrates the comparison of in-sample and out-of-sample performance errors for two methods across varying numbers of stocks considered for the index. Notably, Method-1 shows a sharp decline in errors with increasing stocks, while Method-2 maintains a relatively stable error rate across the spectrum.

Method 1 vs Method 2:

- Time: Method 1 is significantly faster, often completing in a short span, whereas Method 2 runtime was limited to one hour to get a reasonable solution.
- Error: Method 2 consistently demonstrates lower error rates, both in-sample and out-of-sample, compared to Method 1.

6. Specifics

1. **How well does your portfolio, constructed with 2019 data, track the index in 2020? (Method 1 and Method 2)**
   The portfolio's tracking performance, constructed using 2019 data, for the NASDAQ-100 index in 2020 can be gauged by the "out-of-sample" performance metrics provided in the report:
   Method 1
   Method 2

2. **With m = 5, 10, 20, ..., 90, 100, analyze the performance of the portfolio for each value of m. How does the performance change?**
   In most cases, as the value of 'm' increases, we tend to see a decrease in the error. Nevertheless, an anomaly occurs between 'm' values of 50 and 60, where the error increases before resuming its downward trend. This peculiar behavior might be attributed to the fact that the strategy employed for 'm' equal to 60 assigned greater importance to a particular stock that displayed pronounced outlier behavior during the COVID-19 period.

3. **Is there some value of m, where there are diminishing returns of including more stocks in the portfolio?**
   Diminishing returns become evident when expanding the number of stocks in the portfolio. By examining the out-of-sample error for Method_1, it becomes clear that the error plateaus once ‘m’ reaches 30 and doesn't exhibit any significant improvement until ‘m’ reaches 70. Therefore, having a portfolio of 70 stocks is a highly recommended choice. However, if a lower value for m is desired, ‘m’=30 represents the next best alternative.

4. **What’s the smallest value of big M you could use?**
   We utilized the smallest possible value for **"big M"**, which is **1**. This choice is dictated by the requirement that the sum of weights for all stocks in the portfolio must equal 1. Consequently, no individual stock's weight can surpass 1.

7. Summary and Recommendations

Methodologies Overview:

- Method-1 involves a binary programming approach for stock selection and an optimization model to assign portfolio weights. The objective is to maximize correlation among selected stocks while minimizing the return difference between the portfolio and the NASDAQ-100 index.

- Method-2 focuses on dynamically selecting an optimal combination of stocks and determining their weights. This model aims to minimize the absolute daily return difference between our custom fund and the NASDAQ-100 index.

Performance Evaluation:

Error rate across different stock counts for both in-sample and out-of-sample data.

- Comparison: Method-1 is more time-efficient, while Method-2 consistently showcases lower error rates, making it a superior choice in terms of accuracy.

Optimal Number of Stocks:

Around **40** stocks seem to be the sweet spot where the out-of-sample performance peaks (refer to out of sample performance method 2 here- **Method 2**), indicating diminishing returns beyond this point. This is the point where adding more stocks does not significantly improve the tracking of the NASDAQ-100 index.

Recommendations:

1. **Number of Stocks:** We recommend including approximately 40 stocks in the mutual fund portfolio. This number strikes a balance between diversification and cost-efficiency while closely mirroring the NASDAQ-100 index.

2. **Stock Selection and Weighting:** Adopt Method-2 for both stock selection and weight determination. Its dynamic stock selection based on optimization results and the focus on minimizing absolute daily return differences makes it a superior choice. Although it might be computationally intensive, the accuracy benefits outweigh the time costs, especially when making significant investment decisions.

3. **Future Analysis:** Continuously monitor the portfolio's tracking error and adjust the stock selection and weights semi-annually or annually to account for market changes and ensure consistent tracking of the NASDAQ-100 index. Another possibility is to increase the runtime from 1hr, this would lead to different stock selections and weights that could potentially track the index more accurately.

Conclusion:

Our in-depth analysis indicates that a portfolio composed of around 40 stocks, selected and weighted using the Method-2 optimization approach, will most closely track the NASDAQ-100 index. This strategy offers a balanced blend of accuracy, efficiency, and cost-effectiveness, ensuring maximum value for our investors.

End of Report