Project 2: Index Fund Optimization

Authors: Casey Copeland (cmc6793), Junsu Kim (jk48353),
Brandon Pover (bnp669), Xiaohan Sun (xs3236)

Objective

We have created an optimization algorithm to show the return performance of an index fund of *m* NASDAQ-100 securities. Based on the procedures and results of our index fund, we will go over recommendations for component stock selections and weight selections for a mutual fund. The goal is to learn passive investment strategies to track the investment return of the NASDAQ-100 based on minimizing the tracking error we created using linear programming, while only using a subset of index components in the fund.

Discussion

Stock Selection

We first formulated an integer program to pick *m* stocks to best match our index, the NASDAQ-100. To pick these *m* stocks, we created a correlation matrix *p* that displayed the similarity between each component stock available. Depending on the value of *m* (ranging from 5 stocks to 100 stocks, i.e. all the stock in the index), the integer program would pick the best *m* stock based on their similarity and ability to match the index returns. The objectives and constraints below were programmed to achieve the optimal stock selection.

- **Objective:** Maximize similarity between the 100 NASDAQ stocks available to choose and their representative investments in the index fund.
    - There are a total of 10,100 decision variables in this optimization problem. The first 10,000 represent the similarity values between stock components we are optimizing over and the index performance. 100 X 100 = 10,000 total similarity values which became 10,000 binary decision variables for our program to select the optimal *m* number of similarities. If selected, the coefficient would be 1, if not selected the coefficient will be 0. The last 100 are binary variables that force the similarity value to become 0 if the stock component is not chosen in the optimal fund.
- **Optimization Constraints:**
    1. Exactly *m* stocks will be held in the fund. We cannot buy more or less than the specified m. We can track the performance of each portfolio of different levels of stocks and determine which performs the best.
    2. Each stock i has exactly one representative stock j in the fund. This constraint guarantees that each stock component in the index has only one representative stock in the fund if chosen.
    3. Stock i is best represented by stock j only if stock j is in the fund. This adds to the previous constraint by guaranteeing that the best representative index stock only exists if it is chosen in the fund.

Below are the five stocks chosen by our first iteration of the stock selection optimizer. We then ran the algorithm to select m = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 stocks to be in our index fund so we could map their various returns.

```
['LBTYK', 'MXIM', 'MSFT', 'VRTX', 'XEL']
```

Note: Liberty Global PLC ($LBTYK), MXIM ($MXIM), Microsoft Co ($MSFT), Vertex Pharmaceuticals Inc ($VRTX), Xcel Energy Inc ($XEL)

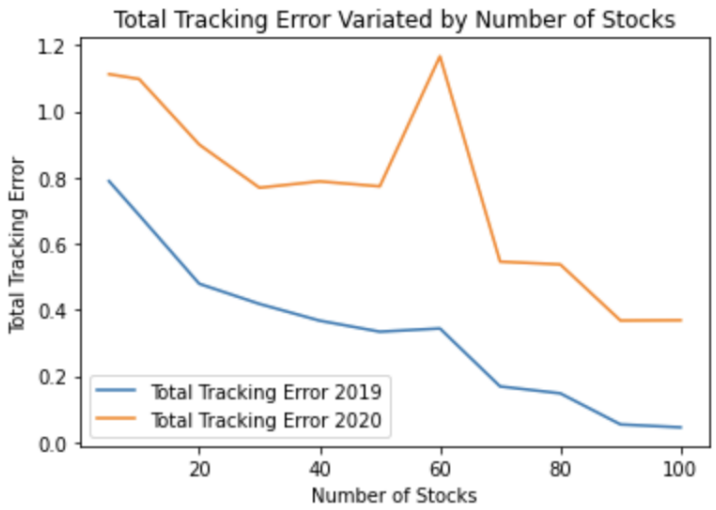
Portfolio Weights

After selecting the m stocks that best represent the NASDAQ-100, we will determine the portfolio weights that will match the index return as closely as possible.

- **Objective:** Minimize the absolute value tracking error in performance between our index fund and the market index. This is a nonlinear function, so we reformatted the problem into a linear programming problem.
- **Optimization Constraints:**
    1. The first constraint guarantees that all chosen weights will sum to 1 for the *m* chosen stocks.
    2. The second constraint reformats the absolute value tracking error as a linear problem by optimizing over a single *y* variable instead of the function of the difference between index return and fund returns. Thus, the y variable has to be greater than or equal to both signs of the absolute value function to find the optimal weights.

Fund Performance on 2-Step Optimization Problem

	m	2019	2020
0	5.0	0.789178	1.112437
1	10.0	0.686533	1.097709
2	20.0	0.478836	0.899598
3	30.0	0.418015	0.769110
4	40.0	0.367439	0.788335
5	50.0	0.334010	0.773216
6	60.0	0.343788	1.166438
7	70.0	0.168587	0.545744
8	80.0	0.147683	0.537323
9	90.0	0.053779	0.367790
10	100.0	0.044911	0.368682



Note: m = number of stocks in the index fund, 2019 = performance with 2019 NASDAQ-100 data, 2020 = performance with 2020 NASDAQ-100 data

In this automated approach of picking stock components, we can passively select the stocks and update them with new market data simply by entering a new csv of stock market data and re-running the algorithm.

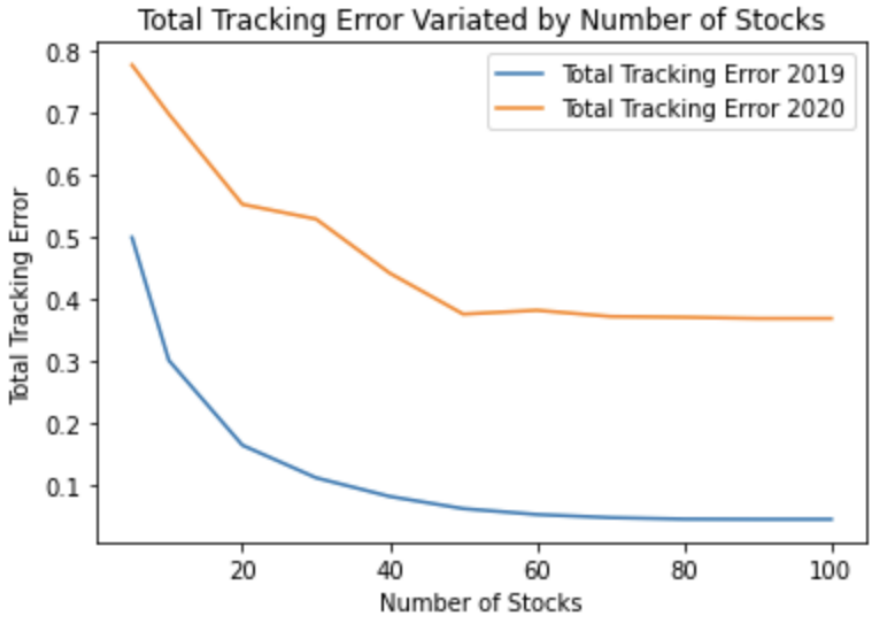
The 2-step approach, selecting the representative stocks and then the portfolio weights, was much more computationally efficient than the Integer Programming approach. Clearly, 100 stocks minimize the tracking error substantially. The five and 60 stock funds seem to maximize the tracking error. There is a total tracking error for both these funds that is over 100%! That is extremely bad if the purpose of the fund was only to passively track the NASDAQ 100. There does not seem to be a place of diminishing returns for an arbitrary M value as the error consistently reduces as more stocks are added.

Although, there seems to be some disconnect between which stocks are chosen and what weights they will have. This can be fixed by running an optimization problem over both stock selection and weights at the same time, which will be done below (i.e., the Integer Programming approach).

The tracking error on 2019 data is much less when compared to the 2020 error. This is not surprising as the stocks and weights are optimized on 2019 data (i.e., training error vs. test error). The training error will always be lower as that is what the model is fitted on to predict or in this case minimize. The test error is more important when considering passively tracking an index, so even with 100 components in the fund the total tracking error is still 40%.

Integer Programming Approach

A different approach to create an optimal index fund is through integer programming and the use of "big M" constraints. This constraint makes it where if stock selection = 0 (meaning that stock was not selected) then the weight of the stock will also be 0. This combines stock selection and weight selection into one step in order to craft the index fund. This method is extremely computational heavy and requires massive amounts of computational power to even find the optimal solution. As a result, this analysis will cap optimization time to an hour for each *m* stock fund. Even though this is an integer programming problem, the objective is still to minimize the absolute value of tracking error between the index and fund, so the same reformatted linear programming problem will be used as the one described above for the weight optimizer.



The graph above shows a much smoother decline of fund tracking error with the increase of index components in the tracking fund. The in-sample error goes almost to zero for 2019 index data. The out of sample tracking error seems to have diminishing benefits after about 50 stocks.

There is also higher variance in returns of our index fund on the 2020 data versus the 2019 data. This is common when you have a lag in information for time-series data. Our index fund was created using 2019 data and evaluated its performance on both the 2019 and 2020 index. Because this is a passive stock selection method, you can easily update your index fund selections annually to closely follow the current index performance.

Recommendation

From this analysis, we can see that using the integer programming method creates a more accurate result to minimize tracking error. There is an obvious trend after 50 stocks are added to the fund there is minimal or no decrease to out-of-sample tracking error. According to this process, it is optimal and more efficient to have about 50 stocks that have the highest similarity to the index than it is to include more stocks. When considering direct and indirect trading costs, there is no reason to have more than 50 stocks in the fund if we plan on passively tracking the index. Although, we would recommend running this optimization problem monthly or quarterly with the last 12 months of data continuously as the tracking error for out of sample data is quite high for tracking an index passively. Lastly, we recommend investing in a highly efficient computer to run this optimization problem as mentioned above because integer programming problems can have extreme computation times such as this one.