Project 2: Integer Programming

February 20, 2020

Index Tracking

Money management strategies are primarily classified as either 'active' or 'passive.' The most common passive approach is that of "indexing." Its goal is to choose a portfolio that mirrors the movements of the broad market population or a market index. Such a portfolio is called an index fund. For example, the QQQ Index fund tracks the NASDAQ-100 index.

In recent years, index funds become more and more popular:

> *With index funds outperforming their actively managed counterparts on a large scale, asset flows have grown significantly in index fund products. For 2017, investors poured more than $692 billion into index funds across all asset classes. For the same period, actively managed funds experienced $7 billion in outflows.* Investopedia

Constructing an index fund that tracks a specific broad market index could be done simply purchasing all the stocks in the index, with the same weights as in the index. However, this approach is impractical (many small positions) and expensive. An index fund with q stocks, where q is substantially lower than the size of the target population (n), seems desirable.

The Project

In this project, we will create an Index fund with **25** stocks to track the NASDAQ-100 index. We will do this in multiple steps. First, we will formulate an integer program that picks exactly q out of n stocks for our portfolio. This integer program will take as input a 'similarity matrix,' which we will call ρ. The individual elements of this matrix represent:

$$\rho_{ij} = \text{similarity between stock } i \text{ and stock } j$$

An example of this is the correlation between the returns of stocks i and j. But one could choose other similarity metrics ρ_{ij}. Next, we will construct a portfolio by selecting appropriate weights for the **25** stocks and finally evaluate how well our index fund does as compared to the NASDAQ-100 index.

The Integer Program

$$Z = \max \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} x_{ij}$$

$$\begin{aligned}
\text{Subject to} \quad & \sum_{j=1}^{n} y_j = q \\
& \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, \ldots, n \\
& x_{ij} \leq y_j \quad \text{for } i = 1, \ldots, n; \text{ for } j = 1, \ldots, n \\
& x_{ij}, y_j \text{ are binary} \quad \text{for } i = 1, \ldots, n; \text{ for } j = 1, \ldots, n
\end{aligned}$$

The binary decision variables y_j indicates the stock j from the index are present in the fund ($y_j = 1$) or not ($y_j = 0$). For each stock in the index, $i = 1, \ldots, n$, the binary decision variable x_{ij} indicates the stock j in the index is the best representative of stock i ($x_{ij} = 1$) or not ($x_{ij} = 0$).

The first constraint selects exactly q stocks to be held in the fund. The second constraint imposes that each stock i has precisely one representative stock j in the index. The third constraint guarantees that stock i is best represented by stock j only if j is in the fund. The objective of the model maximizes the similarity between the n stocks and their representatives in the fund.

Another way of thinking is that you have two sets. Set I has all stocks in the index. Set F has fund stocks. You want to create a link that maps elements of set I to elements of set F. The first constraint in the formulation above is equivalent to saying that not more than q elements of F can be mapped from set I. The second constraint suggests that each element in the set I will map to a single element in set F. The third constraint suggests that if an element from set I gets mapped to an element in set F, then the element of set F must be present in the fund. The binary constraint on x_{ij} indicates if a link between element i in set I to element j in set F exists. The weight on that link is the correlation ρ_{ij} between stock i in set I and stock j in set F. Many such mappings which satisfy the above constraints exist. Your objective gives you the best mapping. This is the basic idea of bipartite matching.

Calculating Weights

Once the model has been solved, and a set of q stocks has been selected for the index fund, a weight w_j is calculated for each stock j in the fund:

$$w_j = \sum_{i=1}^{n} V_i x_{ij},$$

where V_i is the market capitalizatin of stock i at some fixed time. So w_j is the toal market value of the stocks represented by stock j in the fund. The fraction of the index fund to be invested in stock j is proportional to the stock's weight w_j, i.e.,

$$\frac{w_j}{\sum_{f=1}^{n} w_f}.$$

Notice that the re-weighted sum is 1.

Files Provided on Canvas

a. *N100StkPrices.csv* contains daily stock prices for the NASDAQ 100 stocks in 2012. You will use this to calculate the similarity matrix ρ.

b. *N100Monthly.csv* is the given monthly stock prices for 2013. You will use them to evaluate how closely your portfolio did in 2013. The monthly NASDAQ 100 in 2013 is 2731.53, 2738.58, 2818.69, 2887.44, 2981.76, 2909.60, 3090.19, 3073.81, 3218.20, 3377.73, 3487.82, and 3592.00. You may also use the index value, 2660.93, in Dec 2012

c. *readData.R* is a script to read the stock prices into R and clean the data (handles missing values, changed stock tickers, etc.).

- For the daily data, it computes a price matrix (rows standing for dates and columns for the 100 stocks), the list of stocks' tickers, the dates and a shares matrix (similar to the price matrix in structure) that contains the total number of shares outstanding in the company (you will need this to compute market capitals of each company).
- For the monthly data, it calculates the same things except for the total number of shares.

The Specifics

1. Calculate the daily returns for each stock using the 2012 price data.
2. As our initial candidate for the similarity matrix, find the correlation matrix for the returns of the 100 stocks. Note that there will be missing data in the price matrix (NA which stands for Not Available). You need to specify 'use' argument in the 'cor' function to handle NAs.
3. Code the integer program above as another function that returns the weights for each of the stock that needs to be in your portfolio.
   - This will amount to simply formulating the integer program, solving it and then using the market capitalization of each company on the last date to compute weights. The output weights will be a vector of size n with only q non-zero elements denoting the weights.
   - (Note: this will have a vast number of variables and constraints. As you program, you may want to avoid printing out your output. **In the final code you submit, please make sure to have no output at all– especially the x values and your coefficient matrix!**)

```
weights = constructFund(rho, q, priceMat, sharesMat, unique_tickers, unique_dates)
```

4. Use your weights to construct an index portfolio at the end of 2012.
   - Compare how this index portfolio performs monthly in 2013 as compared to the NASDAQ 100 index using the 2013 stock data provided. Here you may assume that you can directly invest in the Index as if it is stock. First, calculate the number of shares for every stock in your fund at the end of 2012 using 1 million dollars in cash. Then calculate the value of your fund starting December 2012. Next using the value of the index in December 2012 and 1 million cash, calculate the units of index you will buy. Then calculate the value of the index. Present your findings using any visualizations or tabulations. You can leave the shares as non-integers because the effect that the non-integer parts of shares have should be marginal.
5. Earlier you used correlation as the similarity measure. Now instead create your similarity measure and put it in a function similarityMat that has the same inputs and outputs
   - Use this rho in your function call to constructFund and as in Step 4, evaluate the performance of this fund as well. Please compare the new fund to the previous fund. Explain why the performance is better (or worse).

```
rho =  similarityMat(priceMat, sharesMat, unique_tickers,unique_dates)
```

The Deliverables: 1. The following R files: a. similarityMat.R b. constructFund.R

2. A report that contains
   a. A description of the similarity metric you used.
   b. All your visualizations and tabulations for the two funds you constructed. To run your scripts, we will use this sequence:

```
source('readData.R')

q=25;

source('similarityMat.R')

source('constructFund.R')

rho =  similarityMat(priceMat, sharesMat, unique_tickers,unique_dates);

weights = constructFund(rho, q, priceMat, sharesMat, unique_tickers, unique_dates);

#Followed by our evaluation code
```

Submission Instructions:

Name your report as **project2__gZ.pdf** (Z is your group number). Upload your report, the file similarity-Mat.R, and the file constructFund.R as three separate files[1] to Canvas.

[1]We want to use speedgrader which requires different files.