Optimization
Project 2 – Integer Programming

Deliverables
One Python code file (.py or .ipynb) and one well-written PDF file, submitted to Canvas. Your report should go into some detail about how you solved the problem, include some graphs that explain your results, and include relevant code chunks in the final output. 66% of your grade will be based on whether you get the problem right or not, the remaining 34% will be based on the quality of your analysis. We will re-run your code with a new data set. If you don't get the right answer or your Python file doesn't run, we will go through your code and give partial credit accordingly. The easier it is to read your code the easier it is for us to understand what you're doing, so use a lot of comments in your code!

Problem Overview

Equity money management strategies are largely classified as either 'active' or 'passive'. The most common passive strategy is that of "indexing" where the 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.

Constructing an index fund that tracks a specific broad market index could be done by simply purchasing all n stocks in the index, with the same weights as in the index. However, this approach is impractical (many small positions) and expensive (rebalancing costs may frequently be incurred, price response to trading). An index fund with m stocks, where m is substantially smaller than the size of the target population, n, seems desirable.

In this project, we will create an Index fund with m stocks to track the NASDAQ-100 index. We will do this in multiple steps. First, we will formulate an integer program that picks exactly m 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, ρ_{ij}, represent similarity between stock i and 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, you will solve a linear program to decide how many of each chosen stock to buy for your portfolio and finally evaluate how well your index fund does as compared to the NASDAQ-100 index, out of sample. You will examine the performance for several values of m.

Stock Selection

The binary decision variables y_j indicates which stocks j from the index are present in the fund ($y_j = 1$ if j is selected in the fund, 0 otherwise). For each stock in the index, i = 1, . . . , n, the binary decision variable x_{ij} indicates which stock j in the index is the best representative of stock i ($x_{ij} = 1$ if stock j in index is the most similar stock i, 0 otherwise).

The first constraint selects exactly m stocks to be held in the fund. The second constraint imposes that each stock i has exactly 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 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 better 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.

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

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

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

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

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

Calculating Portfolio Weights

To get the portfolio weights you will try to match the returns of the index as closely as possible. Let's call r_{it} the return of stock i (where stock i is one of the chosen stocks from above) at time period t, q_t the return of the index at time t, and w_i the weight of stock i in the portfolio. You want to solve

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

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

$$w_i \geq 0.$$

The absolute value function is non-linear, so we must reformulate this optimization problem as a linear program. I'm not going to tell you exactly how to do this, but I will give you a very big hint.

Suppose I want to solve

$$\min_{x} \quad |x-1| + |x-2| + |x-3|$$

It should be apparent that the answer to this problem is x=2. However, this is a non-linear program; here's how you could formulate it as an LP.

$$\min_{x,y_1,y_2,y_3} y_1 + y_2 + y_3$$

$$\begin{aligned} s.t.\quad & y_1 \geq x-1 \\ & y_1 \geq -(x-1) \\ & y_2 \geq x-2 \\ & y_2 \geq 2-x \\ & y_3 \geq x-3 \\ & y_3 \geq 3-x. \end{aligned}$$

Let's think about what y_1 does here. We are trying to minimize over y_1 and we constrain y_1 to be bigger than both x-1 and 1-x. Only one of x-1 or 1-x can be bigger than zero, so if y_1 is bigger than both of them then y_1 must be bigger than |x-1|. If we're trying to make y_1 as small as possible and the only constraint on y_1 is that $y_1 \geq |x-1|$, then y_1 will be exactly equal to |x-1|. Now since we're also optimizing over x, and x shows up in all the constraints for y_1, y_2, y_3 then these two optimization problems are exactly the same. The first formulation is non-linear, and the second formulation is an LP!!! Take this hint and apply it to the portfolio weight construction.

Specifics

1) Two csv files are included with this assignment. One of those files contains daily prices of the index as well as the component stocks of the NASDAQ-100 in 2019. The first column is the date, the second column is the index price (NDX), and columns 3-102 are the prices of the component stocks. The second file is the same, except it's 2020 prices. (We are using pre- and post-shutdown data to see if anything fishy happens!) There were actually 103 stocks in the NASDAQ-100 in 2020, but 3 of them are new stocks that don't have data for all of 2019, so I just removed them. You're going to use the 2019 file for all your portfolio construction tasks and then analyze the performance on the 2020 file; that is, how well does your portfolio, constructed with 2019 data, track the index in 2020? When we grade your assignment, we will use different csv files with potentially a different number of days and a different index with a different number of component stocks. You will need to calculate the returns of the stocks in the 2019 file to calculate the correlation matrix for stock selection and weight construction, and you will need the returns of the index for weight construction. You will also need the daily returns in 2020 for both the index and the stocks to evaluate the performance of your portfolio in 2020. Use the correlation matrix of returns as ρ.

2) Start with m=5. Find the best 5 stocks to include in your portfolio and the weights of those 5 stocks, using the 2019 data. How well does this portfolio track the index in 2020? That is, calculate $\sum_{t=1}^{T}\left|q_t - \sum_{i=1}^{5} w_i r_{it}\right|$ using the 2020 data (except w_i is from your 2019 solution…).
3) Redo step (2) with m = 10, 20, …, 90, 100 (obviously when m=100 you don't need to solve for which stocks to include, because they're all included). Analyze the performance of the portfolio for each value of m. How does the performance change? Is there some value of m, where there are diminishing returns of including more stocks in the portfolio? You can also look at the in-sample performance. That is, evaluate the performance in 2019 using 2019 portfolio construction and 2019 data. How is performance in 2019 different than performance in 2020? Why is it different? Be sure to write your code so that if there are more or fewer than 100 stocks in the csv file it stops at the right place.
4) Another way you could solve this problem is to completely ignore the stock selection IP and re-formulate the weight selection problem to be an MIP that constrains the number of non-zero weights to be an integer. To do this take the weight selection problem and replace m with n so that you are optimizing over ALL weights: $\min_{w} \sum_{t=1}^{T}\left|q_t - \sum_{i=1}^{n} w_i r_{it}\right|$. Now define some binary variables $y_1, y_2, \ldots, y_n$ and add some constraints that force $w_i = 0$ if $y_i = 0$ using the 'big M' technique (What's the smallest value of big M you could use?). You also need to add a constraint that the sum of y's is equal to m (m and M are different things here…). It turns out that this is a VERY hard problem for gurobi to solve. After 24 hours running on my desktop gurobi didn't find a solution! We can force gurobi to quit looking for a solution after a specific amount of time by setting the TimeLimit value, in units of seconds, in the params list (the place where we tell gurobi to shut up). Redo parts 2 and 3 with this new method to find weights. For each value of m, limit gurobi to work for 1 hour. Which method works better on the 2020 data, the original method or this new method? Note that your code will need to run for up to 10 hours to create your final output. I would suggest that you plan ahead and set this to run overnight. You can set your Python file to save your results for this part to a csv file, and then you can also have it check to see if that csv file exists, if it exists grab those results without spending the 10 hours, and if it doesn't then re-solve the problem. That way you only have to do the big solution once and you can still work on the formatting of your Python file. In your Python file create a clearly obvious variable at the top, that is equal to 3600, that you reference to limit gurobi's time. This way when we grade your solutions, we can set it to something smaller to make sure your code works without having to wait 10 hours for everyone's code to run.
5) Pretend you are an analyst at a mutual fund. You boss has asked your team to come up with a recommendation for how many component stocks to include in the fund and how to pick their weights going forward. Write this project as if this is what you're going to deliver to your boss. Your boss is pretty technical and understands optimization, so don't be afraid to include quantitative material. Your boss is also busy, so be sure to include some visualizations to get the important points across. For the purpose of your recommendations, you can assume that your boss is interested in the data posted with the project.