ASHISH DHIMAN

+1(404) 509-0254 | Atlanta, GA

ashish1610dhiman@gmail.com | linkedin.com/in/ashish1610dhiman | ashish1610dhiman.github.io/ad_cv/

Introduction

The following work deals with approaches to solve a particular class of Portfolio Optimisation problems known as Enhanced Index Tracking, a variation of the classical Index Tracking. Index Tracking problem deals with determining a portfolio of assets (henceforth referred to as tracking portfolio) whose performance replicates, as closely as possible, that of a financial market index (or any arbitrary benchmark chosen), measured by tracking error.

Enhanced index tracking improves upon the original problem of Index Tracking by additionally trying to maximise the excess return of the tracking portfolio (over the benchmark) while limiting tracking error, or in other words the optimal portfolio is expected to outperform the benchmark with minimal additional risk over the index. Thus Enhanced Index tracking deals with two competing objectives i.e. the expected excess return of the portfolio over the benchmark and the tracking error from the benchmark.

Problem Statement

In this work, we evaluate if, the application of dimension reduction techniques (namely NPCA and NMF) to reduce the temporal dimensioanlity of data is helpful in the context of Enhanced Index Tracking. The hypothesis here is that dimension reduction would help replicate the index only at a macro level by minimising the minute (and futile) fluctuations. This essentially translates to limiting the resolution for tracking of benchmark to user decided period along with added benefit of decrease in computational complexity of the original problem.

Dataset used

We evaluate out approach using two popular index funds:

1. Hang Seng:
   31 stocks | March 1992 to September 1997 | Gradual up trajectory of market | weekly, NPCA reduced, NMF reduced

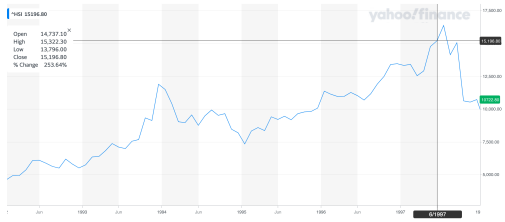

Figure 1: Hang Seng Trend

2. S&P500:
   500 stocks | Feb 2013 to Mar 2018 | Explosive up trajectory, following static market | daily, weekly, NPCA reduced, NMF reduced

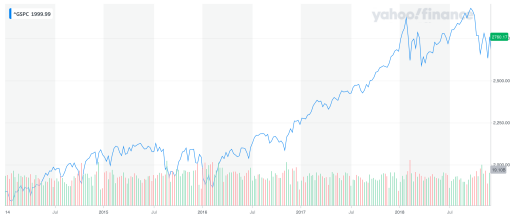

Figure 2: S&P500 Trend

Linear Formulation of EIT

A Mixed Integer Linear Problem (MILP) formulation of the Enhanced Index Tracking problem has been used here. Since Integer Programming is an NP-complete problem, even solving relatively small problems might become hard. For our use case we employ the Heuristic Kernel Search algorithm, to arrive at a solution of the problem, which works by iteratively expanding the searching space of securities.

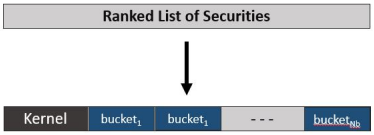

Figure 3: Sorting and Bucketing the Stocks

We define the excess return (z_1) of tracking portfolio over benchmark as the absolute excess value of portfolio over benchmark averaged over time, i.e:

$$z_1 = \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=1}^{n} r_{j,t}q_{j,T}X_j^1 - r_{I,t}C\right]$$

Tracking Error (TrE) is defined as absolute deviation of portfolio from benchmark averaged over time. Note the linear nature of Tracking Error here compared to quadratic in **??**.

$$TrE = \sum_{t=1}^{T}\left|\theta I_t - \sum_{j=1}^{n} q_{j,t}X_j^1\right|$$

where $\theta = \frac{C}{I_T}$ is used to scale the value of Benchmark. Let d_t and u_t be the variables depicting downside and upside deviation of tracking portfolio from benchmark at time t. Hence it follows, $d_t - u_t = \theta I_t - \sum_{j=1}^{n} q_{j,t}X_j^1$ for $t = 1, 2, \cdots, T$. Thus Tracking Error can be expressed as:

$$TrE = \sum_{t=1}^{T}(d_t + u_t)$$

The above two metrics are then used to formulate the final optimisation problem.

$$\begin{aligned}
\underset{x\ \in\ \mathcal{X}}{\text{Maximize}} \quad & z_1 = \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=1}^{n} r_{j,t}q_{j,T}X_j^1 - r_{I,t}C\right] \\
\text{subject to} \quad & \sum_{t=1}^{T}(d_t + u_t) \le \xi C \\
& d_t - u_t = \left(\theta I_t - \sum_{j=1}^{n} q_{j,t}X_j^1\right) \forall t = 1, 2, \cdots, T
\end{aligned} \tag{1}$$

**Additional constraints abstracted for brevity here*

Dimension Reduction

As the number of observations increase, oscillations between their returns increase and thus reductions start producing outliers. To avoid these outliers, the first step of the dimension reduction methodology requires dividing the unreduced dataset into k equidistant time windows X_{ti} as described in 4.

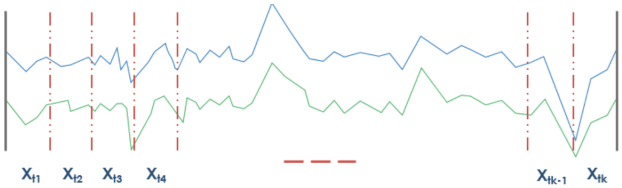

Figure 4: Dividing original time series to equal size windows

Dimensionality Reduction is applied to decrease the time-dimension of each of these X_{ti}. The reduced dimensions for each of the X_{ti} are then combined into one on the basis of SVP (Statistical Variance Procedure) which essentially takes a weighted sum of the reduced dimensions with weights being the proportion of variance explained in the original data. Hence we derive a single vector f_i of stock prices for each of the k subsets of data. These reduced prices then become the input (Reduced Data=$[f_i]_{k*n}$) for our tracking problem.

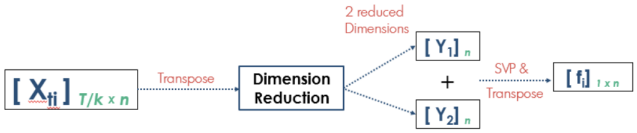

Figure 5: Dimension Reduction Framework

Results

We identify that using reduced data of NMF has the effect of improving risk/return characteristics with increasing k, i.e we see Higher Slope of *return_pu_risk* vs k, i.e. with increase in size of portfolio, higher increase in *return_pu_risk*. All the results are validated on both in sample and out of time data.

EIT_type	dual			basic		
Dataset — Hang Seng	*1*	*npca*	*nmf*	*1*	*npca*	*nmf*
return_pu_risk_sample_transpose	1.356284e-07	1.961903e-06	-7.874494e-07	1.962252e-05	3.507791e-05	6.306736e-07
return_pu_risk_oot_transpose	-4.498208e-07	1.359339e-06	2.753823e-06	1.032645e-04	2.727054e-05	-3.150138e-05

Table 2: Slope of return/risk vs k (16-25)

From the above table, we see data reduced by NPCA, helps both dual and basic EIT approach, but more so for dual approach. This might be because bi-weekly reduced data has lowest amount of information available, adding extra vars in form of securities best increases performance. However basic might already have significant information from weekly data, hence adding more securities to portfolio provides lesser marginal utility.

Computational Environment

The above problem is solved using a combination of programming languages including Python and R. While dimension reduction is performed in R, the Kernel Search framework is implemented in Python. The final EIT Optimisation problem is solved using MIP library in python. Also since EIT problem needs to be run with different set of input parameters, and is compuationally expensive, we use Joblib library to run parallel jobs on Kaggle Kernels.
https://www.kaggle.com/code/ashish1610dhiman/fork-of-eit-dual-vs-basic-unreduced-sp500-nmf3/notebook