Correlation-Driven Nonparametric Learning Decay Method

One novel online portfolio selection method
Optimization Modelling in Finance Course Project

Published by

Qin Lu (ql224)

Yudong Yang(yy723)
Sizhang Zhao(sz459)

The Cornell University

March 26, 2017

Abstract

Online portfolio selection is a dynamic asset allocation method based on online learning techniques. The portfolio selection task is transformed into a sequencial decision problem. And different approaches would be adopted to achieve the goal. In this paper, we focus on one of the "pattern-matching" based approaches, correlation-driven nonparametric learning. We improved the exsiting strategy and built up our own Correlation-driven Nonparametric Decay(CORN-D) model. We applied our algorithms to both stock dataset and index dataset and conduct strategy analysis based on MSCI index dataset. And finally we applied the strategy to ETF dataset for backtesting with the chosen parameters and selected methods. We finally drew the conclusion that our strategy is suitable for trading portfolios such as ETFs instead of trading single stocks and performs well under high volatility periods. We also found out that our strategy performed extremely well when there are no transaction costs but the performance became worse when transaction costs included.

Keywords: online portfolio selection, online learning, nonparametric learning, regularization, neariest neighbor, decay

Content

Part I

Introduction

1 Literature Review

There are traditional two portfolio selection methods. The mean-variance method brought up by Markowitz[1], which is suitable for one-period allocation. And the Capital Growth Theory[2], which aims to maximize the expected log return and is more suitable under a dynamic setting. Online portfolio selection is just originated from Capital Growth Therory.

Li et al has summarized different online portfolio selection algorithms[3] and have detailed description of nearly all the strategies in [4]. There are approximately five categories of the algorithms:

- Benchmark algorithms: simple algorithms used as benchmarks
- Follow the winner algorithms: increasing the weights of most successful experts
- Follow the loser algorithms: transfering the wealth from winners to losers
- Pattern Matching algorithms: assuming stationary and ergodic financial time series and predict based on similar past patterns
- Meta Learning: combining expert portfolio and build new portfolios

Györfi seperated the pattern-matching based strategies into two steps[5], the sample seletion and portfolio optimization steps. In the first step, the algorithms would go through all the past and find the similar historical indices, whose next-day return would be used to predict the future return. The second step is the portfolio optimization step, which would find the optimal allocation weight based on all the similar indices.

How to find similar indices? Györfi et al[5][6] firstly tried to discover the similar pattern of the markets. They introduced Euclidean distance to measure the similarity between different time windows based on relative prices. However, Li et al[7] thought the Euclidean distance cannot exploit the directional information between the two market windows. Therefore, it may detect some useful price relatives, but often includes some potentially useless or even harmful price relatives and excludes many beneficial price relatives. Thus, they introduced correlation as the measurement for similarities, which has achieved excellent performance.

As for the optimization step, Li et al[7] introduced the method to maximize expected multiplicative return over the sequence of similar price relatives, that is:

$$\mathbf{b}_{t+1} = arg\ max_{\mathbf{b}\in\Delta_m} \prod_{i\in\mathcal{C}_t} (\mathbf{b}\mathbf{x}_i)$$

where $\mathcal{C}_t$ is the similar set and $\mathbf{x}_i$ is the relative prices for all the assets.

To avoid overfitting, they included different window sizes and different thresholds for correlation calculation and similar window selection. And they took a uniformly average across all the individual weight for different window sizes and thresholds to calculate the predicted weight for next day. They designed two specific algorithms for this step. One of them is called CORN-U algorithm, and they would take the average across the weights for all the window sizes and thresholds. The other one is called CORN-K algorithm, they just chose the top performers with highest cumulative return to calculate the average.

However, based on correlation, they just take linear relationship into consideration but non-linear relationship between different window size is also important to decide the similarity. Also, although they have considered about transaction costs, they did not dig into the methods to control them. What's more, there are lots of constrains in their model such as short sell constraints. Thus, in our paper, we made several improvements for the original model and built up our Correlation-driven Nonparametric Decay(CORN-D) model .

2 Problem Setting

2.1 Basic Problem Setting

For an online portfolio selection problem, an investor would invest on more than 2 assets for more than 1 trading periods. For every period, we can observe the adjusted price data for all the assets (represented by vector $\mathbf{p}_t$). And then we calculate the the relative price for different assets($x_{t,i} = \frac{\mathbf{p}_{t,i}}{\mathbf{p}_{(t-1),i}}$).

And the value that we are interested in is the allocation weight (represented by vector $\mathbf{b}_t = (b_{t,1}, ..., b_{t,m})$), where m is the number of the assets. And $\mathbf{b}_t$ would satisfy the constraints $\sum_{i=1}^{m} b_{t,i} = 1$. The output for a OLPS strategy would be $\mathbf{b}_1^n = (\mathbf{b}_1, ..., \mathbf{b}_n)$. Each $\mathbf{b}_i$ is the allocation weight for different periods.

And the result we are interested is the portfolio cumulative wealth, which is:

$$W_n(\mathbf{b}_1^n, \mathbf{x}_1^n) = W_0 \prod_{i=1}^{n} (\mathbf{b}_t^T \mathbf{x}_t)$$

Practically, there would be transaction costs for every trade. To simplify, we just assume the transcation cost is a propostional one. Firstly, we need to calculate the price adjusted porfolio for last period, which is $\hat{b}_{t-1,i} = \frac{b_{t-1,i} * x_{t-1,i}}{\mathbf{b}_{t-1}^T \mathbf{x}_{t-1}}$. Thus if we take transaction costs into consideration (let γ denote the transaction cost rate), the cumulative wealth would be:

$$W_n(\mathbf{b}_1^n, \mathbf{x}_1^n) = W_0 \prod_{i=1}^{n} [(\mathbf{b}_t^T \mathbf{x}_t)(1 - \frac{\gamma}{2}) \sum_{i-1}^{m} |b_{t,i} - \hat{b_{t-1,i}}|]$$

To construct an online portfolio selection strategy, the goal for an investor is to achieve certain objectives (such as higher cumulative return or higher Sharpe Ratio) through a sequencial manner. The investor can obeserve all the past relative price and compute the new portfolio $\mathbf{b}_i$ for next day. At the next day, the investor would rebalance his positions. And at the end of the day, he will observe the new relative price and he can calculate his P&L based on the information.

2.2 Evaluation

2.2.1 Cumulative Wealth

We often assume we have $ 1 at the beginning, and then we can calculate the cumulative wealth for our strategy according to the formula mentioned in the last section. Generally, higher cumulative wealth, better the strategy is. It should be mentioned that when we want to compare the results for different datasets, we'd better use annulized percentage yield, which is defined as $APY = \sqrt[y]{W_n}$ because the final wealth is dependent with time.

2.2.2 Sharpe Ratio

Annualized sharpe ratio is one of the good ways to measure the risk-adjusted return for portfolios. Sharpe ratio is defined as:

$$SR = \frac{APY - R_f}{\sigma_p}$$

The higher the sharpe ratio, the higher return per risk taken.

2.2.3 Maximum Draw Down

Maximum draw down is a good method to measure the tail risk of the portfolio, which measure the decline from a historical peak of portfolio cumulative wealth. Maximum draw down is define as:

$$MDD = \sup_{t\in(0,n)} \{\sup[0, \sup_{i\in(0,n)} S_i - S_t]\}$$

The smaller the maximum draw down, the more drawdown risk the strategy can tolerate.

2.3 Benchmark Algorithms

2.3.1 Uniform Buy and Hold Algorithm

Buy and hold algorithm is a commonly-used benchmark algorithm. The investor start his position at the beginning of the testing period and hold it till the end of the testing period without rebalances in the middle. For an uniform buy and hold algorithm (denotes by UBAH), the investor's starting position would just be evenly distributed acorss all the assets (i.e. $\mathbf{b} = (\frac{1}{m}, ..., \frac{1}{m})$) and hold the postions till the end of the testing period.

2.3.2 Best Stock Algorithm

The best stock is a hindsight strategy. The investor would choose the stock with the best performance during the period. That is to say, for a testing period, the investor would choose the stock whose cumulative return is the highest although he cannot anticipate the result at the beginning of the period.

Part II

Strategy

In this part, we focus on the logic of the CORN-D strategy. The strategy serially assigns optimal weights to assets during the trading window by matching previous relative price patterns based on constantly updated information.

General settings and notations are specified in 1. As a pattern matching strategy, one of the main issues is to find similar patterns between recent time series and previous data before rebalancing the portfolio. Thus our choice for different ways to match similar patterns is stated in 2, followed by the algorithm given in 3. All the results, robustness tests and sensitivity analysis are given in Part III.

1 General Settings and Notations

For ease of notation, we use relative price series, instead of return series, to observe wealth changes in our setting. Denote the price of an asset i at time t as $S_t^{(i)}$.

The relative price series is defined as,

$$x_s^u = (x_s, x_{s+1}, ..., x_u)^T,$$

where

$$x_t = \left(\frac{S_t^{(1)}}{S_{t-1}^{(1)}}, \frac{S_t^{(2)}}{S_{t-1}^{(2)}}, ..., \frac{S_t^{(d)}}{S_{t-1}^{(d)}} \right).$$

Note that x_s^u is a relative price matrix when there are multiple assets.

At each time point t during the trading period, we have a recent return series denoted by x_{t-w+1}^t, where w is the window size for the selected recent data. Then, looking back at all the historical data x_1^{t-1} before t, certain dates i are selected into a similar indices set C_t,

$$C_t = \left\{ w < i \leq t \,\middle|\, x_{i-w}^{i-1} \text{ is similar with } x_{t-w+1}^t \right\}.$$

Part 2 gives details on how to decide C_t based on correlation.

2 Similar Pattern Selection

Choosing past date with similar patterns can be translated into defining appropriate distance between two relative price matrices. Naive guess may be Euclidean distance. However, there might be problems using Euclidean distance.

Consider a special scenario where there is only one asset. Two relative price series $X_{i-1}^i = (0.8, 0.9)$ and $X_{j-1}^j = (0.8, 0.7)$ are to be compared with the recent price series $X_{t-1}^t = (1.2, 1.1)$. Intuitively, X_{j-1}^j should be selected by our algorithm since it depicts a descending trend of relative prices, while X_{i-1}^i shows an increasing one. However, according to Euclidean distance criteria, X_{i-1}^i is selected since it has a distance of 0.2 to $X_{t-1}^t = (1.2, 1.1)$, while X_{j-1}^j has a distance of 0.41 which is greater than 0.2. If such distance is used for selecting the pattern-similar set, the performance is surely to be undermined. This counter-intuitive behavior motivates us to use a new measure for distance of two matrices.

Another important distance used between matrices is the Pearson correlation. Here we define Pearson correlation for two matrices the same way as it is defined for two vectors, by connecting different columns within the same matrix to form a vector.

$$\mathrm{Cor}_p(x,y) = \frac{\Sigma_{i=1}^{w}\Sigma_{j=1}^{d}(x_{ij}-\bar{x})(y_{ij}-\bar{y})}{\sqrt{\Sigma_{i=1}^{w}\Sigma_{j=1}^{d}(x_{ij}-\bar{x})^2}\sqrt{\Sigma_{i=1}^{w}\Sigma_{j=1}^{d}(y_{ij}-\bar{y})^2}}.$$

However, another issues arrises when considering Pearson correlation. Since the Pearson correlation only considers linear relationship between two vectors, it cannot capture higher order connection. For instance, generate 1000 sample X from $U(0,1)$, denote $Y = X^2$ element-wise. The true relationship is that they are perfectly related, while Pearson correlation gives a value less than 1. To take advantage of higher order effect, we seek to use rank correlations such as Spearman correlation and Kendall-tau correlation.

The Spearman correlation is defined as,

$$\mathrm{Cor}_s(x,y) = \frac{\Sigma_{i=1}^{w}\Sigma_{j=1}^{d}(\mathrm{rank}(x_{ij})-\frac{wd+1}{2})(\mathrm{rank}(y_{ij})-\frac{wd+1}{2})}{\sqrt{\Sigma_{i=1}^{w}\Sigma_{j=1}^{d}(\mathrm{rank}(x_{ij})-\frac{wd+1}{2})^2}\sqrt{\Sigma_{i=1}^{w}\Sigma_{j=1}^{d}(\mathrm{rank}(y_{ij})-\frac{wd+1}{2})^2}}.$$

And Kendall's tau verifies relative trends. Denote (x_{ij}, y_{ij}) as a pair of set. Kendall's correlation examines every combination of two pairs:

$$\mathrm{Cor}_\tau(x,y) = \frac{4}{wd(wd-1)}\Sigma_{(i,j),\ (i',j')}\mathrm{sign}\left\{\left(x_{ij}-x_{i'j'}\right)\left(y_{ij}-y_{i'j'}\right)\right\}$$

Using these two distances to calculate the correlation in the above instance, the higher order effect could be revealed. Thus we take all three distances into account. Define the similar indices set as,

$$C_t(w,c) = \left\{w < i \le t \,\middle|\, \mathrm{Cor}\left(x_{i-w}^{i-1}, x_{t-w+1}^{t}\right) \ge c\right\}.$$

for every choice of window size w and threshold c.

3 Algorithm

Since past data is utilized to forecast, we build the following optimization model to select optimal portfolio weights for every $C_t(w,c)$,

$$\begin{aligned} \min_{b_{t+1}} \quad & -\eta_i \Sigma_{i \in C_t(w,c)} \log\left(x_i \bullet b_{t+1}\right) + \lambda \left\| b_{t+1} - b_t \right\|_2 \\ s.t. \quad & b_{t+1,i} \geq -0.1, \ \forall i \\ & b_{t+1,i} \leq 1, \qquad \forall i \end{aligned}$$

where $\lambda \left\| b_{t+1} - b_t \right\|_2$ is a smoothing regularizer that adds penalty to solutions with too much position change. λ is called the multiplier for the regularizer. η_i is a decay parameter on the past returns which emphasizes the more relevant situations in $C_t(w, c)$. The model serves to find the optimal weights that maximizes the past returns as a whole, meanwhile tradeoff between transaction cost and overall performance.

The dacay parameter has two definations, the exponential decay is defined as

$$\eta_i = (1 - \gamma)^{\text{rank}(i)}$$

and the linear decay is defined as

$$\eta_i = \rho_i.$$

where ρ_i is the correlation parameter.

The algorithm at time t to find weight $b_{t+1}(w, c)$ for each pair of (w, c) is given by,

Table 1: Algorithm 2.1

CORN-D Algorithm: weight_para$_{t+1}(t, w, c, b_t(w, c))$

Input: distance_type = "Pearson", "Spearman" or "Kendall"; window size w; correlation threshold c, current time t; Last year's weights $b_t(w, c)$

Output: optimal weight under parameter w and c: $b_{t+1}(w, c)$

Algorithm Start:

$C_t = \Phi$

if $t < w$:

    $b_{t+1}(w, c) = \left(\frac{1}{d}, \frac{1}{d}, ..., \frac{1}{d}\right)$

end

for $i = w + 1, w + 2, ..., t - 1$:

    Calculate distance: $\rho(w) = \text{Cor}\left(x_{i-w}^{i-1}, x_{t-w+1}^{t}, \text{distance\_type}\right)$

    if $\rho(w) \geq c$:

        Add i into similar index set C_t

    end

end

if $C_t \neq \Phi$:

    Select weight $b_{t+1}(w, c)$ using the optimal solution for the optimization model:

    $\min_{b_{t+1}(w,c)} \ -\eta_i \Sigma_{i \in C_t(w,c)} \log\left(x_i \bullet b_{t+1}(w, c)\right) + \lambda \left\| b_{t+1}(w, c) - b_t(w, c) \right\|_2$

end

else :

    $b_{t+1}(w, c) = \left(\frac{1}{d}, \frac{1}{d}, ..., \frac{1}{d}\right)$

end

Algorithm End.

Now we consider different combination of parameters (w, c). Our way to deal with it is to assign a 2-dimensional prior probability distribution to the parameter set. The prior probability comes from previous data beyond the start of our strategy, by normalizing the performance during prior periods, or can be simply assumed to be uniform when we have no prior information. Denote the probability mass function as $p(w, c)$, where $w = 1, 2, ..., W$, and $c = 0, \frac{1}{L}, \frac{2}{L}, ..., \frac{L-1}{L}$.

Moreover, for each parameter, the better past performance, the greater weight should be applied. Therefore, we calculate the normalized weighted average taking both prior distribution and past performance into account.

$$b_{t+1} = \frac{\Sigma_{w,c}\ p(w,c)W_t(w,c)b_{t+1}(w,c)}{\Sigma_{w,c}\ p(w,c)W_t(w,c)}$$

where $W_t(w,c)$ is the cumulative past performance for parameter choice (w,c). The overall cumulative return is denoted as,

$$W_{t+1} = W_t \centerdot x_{t+1} \bullet b_{t+1}$$

Note that,

$$W_{t+1}\,(w,c) = W_t\,(w,c) \centerdot x_{t+1} \bullet b_{t+1}\,(w,c)$$

Therefore,

$$W_t = \Sigma_{w,c} p(w,c) W_t(w,c).$$

The general algorithm is described as below,

Table 2: Algorithm 2.2

CORN-D Algorithm: CORND(x_1^T, dist_type)

Input: distance_type = "Pearson", "Spearman" or "Kendall"; Trading Period T;
Data: x_1^T

Output: Cumulative wealth W_1^T, Sharpe ratio, Maximum Drawdown

Algorithm Start:

$W_0 = W_0(w,c) = 1, b_1 = b_1(w,c) = \left(\frac{1}{d}, \frac{1}{d}, ..., \frac{1}{d}\right)$,

for $t = 1, 2, ..., T$:

Calculate cumulative wealth:

for w, c in $\{1, 2, ..., W\} \times \left\{0, \frac{1}{L}, \frac{2}{L}, ..., \frac{L-1}{L}\right\}$

$W_t(w,c) = W_{t-1}(w,c) \centerdot x_t \bullet b_t\,(w,c)$

end

$W_t = W_{t-1} \centerdot x_t \bullet b_t$

Calculate new weights for each parameter combination:

for w, c in $\{1, 2, ..., W\} \times \left\{0, \frac{1}{L}, \frac{2}{L}, ..., \frac{L-1}{L}\right\}$

$b_{t+1}(w,c) = \text{weight\_para}_{t+1}(t, w, c, b_t(w,c))$

end

Calculate final weights for next period:

$b_{t+1} = \frac{\Sigma_{w,c}\ p(w,c)W_t(w,c)b_{t+1}(w,c)}{\Sigma_{w,c}\ p(w,c)W_t(w,c)}$

end

Calculate Sharpe ratio, Maximum Drawdown using W_1^T.

Algorithm End.

Part III

Empirical Analysis

1 Data Collection

In this part, we would discuss about the datasets we selected. All of the datasets are chosen from Yahoo Finance.

1.1 Stock Data

Small stocks are more subject to individual shocks, thus there would hardly be patterns among them. Thus we choose value stocks which are components of DJIA (Dow Jones Industrial Average). Our stock pool contains 29 stocks from April 1, 2006 to March 31, 2010 because we think there are different market conditions during that period, both the steadily growing stage and financial crisis stage. And we can test the robustness of our strategy during that period.

1.2 Index Data

Even large and value stocks would be influenced by lots of idiosyncratic shocks, and their patterns are hardly to persist. Thus we include index data for our strategy because index is "portfolio" of stocks and the idiosyncratic shocks could be diversified so that the pattern behind prices would be more possible to be detected. We choose a collection of global equity indices that constitute the MSCI World Index. It contains 24 indices that represent the equity markets of 24 countries. And the period is the same as the stock data set to faciliate the comparisons.

1.3 ETF Data

Although indices could represent a portfolio of stocks, they cannot be directly traded. Thus we choose ETF data to lastly test our algorithms. There are 15 ETFs chosen by us, and some of them are ETFs for different sectors in USA and some of them are equity ETFs for other regions across the world. We choose a longer period to finally back test our algorithms. The period starts at Feb 1, 2006 and ends at May 1, 2012, totally 1573 trading days.

2 Performance

2.1 Performance without Transaction Cost

2.1.1 Data Set Selection

In this section, we compared the performance of CORN-D with some benchmark algorithms in the most important metrics, cumulative return, sharpe ratio and maximum draw down and we observed that CORN-D has a better performace than both benchmark algorithms, especially when the underlying asset is index-based.

First, we experimented CORN-D with benchmark algorithms (uniform buy-and-hold, best-stock) in an index-based dataset MSCI and a stock-based dataset DJIA, the empirical results is in Table 3.

The compared metrics in the table is a simplified version without considering the daily transaction cost, and we observed that in the dataset of MSCI, CORN-D greatly outperformed the benchmark strategies. In the dataset of DJIA, CORN-D still had slightly better performance than other benchmarks.

Table 3: Comparison Between Stock Data Set and Index Data Set

Asset	MSCI			DJIA		
Criterion	CW	SR	MDD	CW	SR	MDD
CORN-D	11977	4.6027	0.2284	6.52	0.9119	0.4059
UBAH	0.9064	-0.2583	0.6474	1.30	0.1050	0.4737
BEST	1.5040	0.2937	0.3935	3.75	0.6847	0.6087

In the Figure 1 we noticed that CORN-D performed extremely greatly during the period when the market has high volatilities. We can see from Figure 1 that the algorithm performed extremely well when financial crisis happened (the graph has comparatively higher slope during the 600-800 trading days).

We can see from Figure 1 and Figure 2 that our algorithms perform better for MSCI dataset than DJIA dataset. And the reason we think is because single stock is exposed more to the random shocks, which would make stock price maintain fewer patterns. However, for indices, its stationarity and ergodicity would be more likely to persist, which made it more possible to find its pattern. Thus in the future studies, we chose index dataset for our analysis.

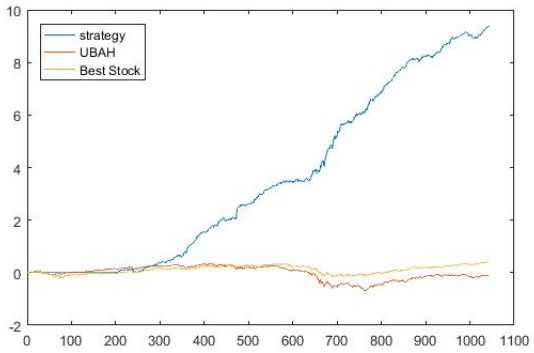

Figure 1: Log P&L for Index Data Sets

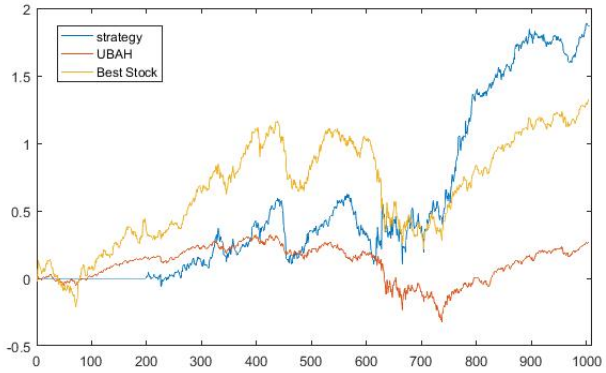

Figure 2: Log P&L for Stock Data Sets

Also as the time went by, CORN-D performed better and better, which might be explained by the fact that CORN-D is a pattern-matching strategies, and with more historical training data, it is more likely to learn a better allocation of the underlying assets based on the previous market performance.

2.1.2 Parameter Selection

Starting Date As we mentioned in the last section, we think the more historical training data, the better for pattern-matching objective. Thus we decided to make the starting data at the 200th trading day instead of the first trading day for all the analysis.

Window Size CORN-D aims at seeking for the most correlated sequences of price relatives in the historical data to the most recent price relatives before the next investment. And the size of the sequence(window) is not obvious. Since we have no pre-knowledge about the window size of similar price relatives, it's reasonable to have a uniform prior over the window size. In the experiments, we tried four uniform distribution: U[1,5], U[6,10], U[11,15] and U[16,20]. From Table 4, we can find among all four uniform distributions, the first distribution has the better performance, which might be explained by the fact that larger the window size is, the correlation might be smaller due to a longer duration. And since the financial market always exhibits random behavior, the noise may greatly reduce the feasibility to pick up the most similar windows.

Table 4: Comparison Between Different Window Sizes

Algorithms	Cumulative Wealth	Sharpe Ratio	Maximum Draw Down
UBAH	0.9064	-0.2583	0.6474
BEST	1.5040	0.2937	0.3935
CORN-D(U[1,5])	11977	4.6027	0.2284
CORN-D(U[6,10])	164.6148	2.6287	0.4258
CORN-D(U[11,15])	106.2472	2.3057	0.3590
CORN-D(U[16,20])	75.1027	2.1406	0.3256

Upper Bound for Single Asset When choosing the allocation for the pool of underlying assets, usually we have the upper bound for each asset as 1, which means the strategy is allowed to invest all wealth into one specific asset. However, from the perspective of risk control, as a remedy, we can lower the upper bound to encourage the diversification. Though lower the upper bound add more constraints to the objective function and returns a lower cumulative returns, when it comes to including the transaction cost, a lower upper bound may help reduce the variation of the allocation, which to some extent can save the transaction cost. So the empirical results will be showed in the next section when considering transaction cost.

2.1.3 Refinement of CORN model

Measurement for Distance Our objective function to to maximize the expected multiplicative return over the sequence of similar price relatives. So our first task is to define the similarity of the price relatives (or the "distance" between two trading days). With a given window of relative prices, we defined the similarity ("distance") as the correlation between two windows of relative prices.

As we have mentioned in Strategy part, there are several defination for correlation. The most common one is the Pearson correlation, and there are another two correlation, so-called Spearman and Kendall correlation which take non-linear effects into consideration. Thus, in this part, we compare the performance under different correlation settings.

From Table 5, we can find that Spearman and Kendall correlation actually did not improve the performance as we expected (however they did lower the maximum drawdown and Kendall actually imporve the sharpe ratio). The reason we think is because Spearman and Kendall are actually suitable to measure the monotonic higher-order correlation (like quadratic and cubic) and they can hardly tell

the difference between concavity and convexity, thus may still be not a appropriate measure for higher-order correlation. And they may include some noise and make the performance worse. But because they did take some higher order effects into consideration, they actually help with the risk management.

However, we would still choose Pearson Correlation for our following analysis because it give out significantly higher cumulative wealth.

Table 5: Comparison Between Different Correlation Definations

Matrics	Pearson	Spearman	Kendall
Cumulative Wealth	11977	8256	10690
Sharpe Ratio	4.6027	4.5692	4.6199
Maximum Draw Down	0.2284	0.1860	0.1760

The Dacay for correlation We proposed two approaches to correlating the different sequence of similar relative prices to the most recent sequence before the next investment. The first approach is to ranking the sequences by the decreasing correlation and using an exponential decay when estimating the expected multiplicative return (the detailed description is shown in the strategy part). The second approach is to ranking the sequences by the decreasing correlation and using an linear decay (the detailed description is shown in the strategy part).

As we can obeserve in Table 6, both decay methods greatly improve the performace for our algorithms as we expected. It is understandable because we lower the contribution for less similar windows and improve the weight for more similar windows. What we have done is similar to a KNN algorithm and the experts (who is more similar) should have higher power for prediction. And linear decay slightly outperform exponential decay under all the comparison metrics. Thus, we would choose linear decay for our future research.

Table 6: Comparison Between Different Decay Methods

Matrics	No Dacay	Exponential Decay	Linear Decay
Cumulative Wealth	8085	11977	12879
Sharpe Ratio	4.5879	4.6027	4.6839
Maximum Draw Down	0.1762	0.2284	0.1780

Relaxation of short sale constraints When maximizing the expected multiplicative return, we relaxed the constraint of lower bound of the allocation to -0.1, which means the strategies is allowed to short sale each asset up to 10% of the current wealth and invest the money to other asset in the pool to ensure the total investment in one period keep 1. We can observed from Table 7 that the relaxing the constraint can help increasing the cumulative wealth greatly. It is easy to understand because through short selling, we impose high leverage on our portolio, thus we can achieve really high return if our algorithms can work out. But there exists practical issues like higher transaction costs for short selling.

Table 7: Relaxation of short sale constraints

Matrics	Short Sell Constraints	No Short Sell Constraints
Cumulative Wealth	25.0587	12879
Sharpe Ratio	2.7211	4.6839
Maximum Draw Down	0.2012	0.1780

2.2 Performance with Transaction Cost

In this section, we considered more practical situation when we add the transaction cost as 0.2%. We can observe from Figure 3 and Table 8, all strategies have a significantly lower cumulative return compare with the situation when there is no transaction costs.

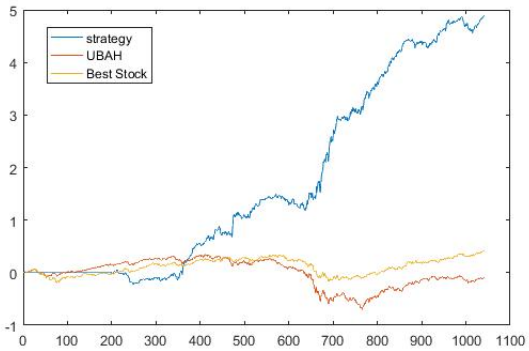

Figure 3: Log P&L when there are transaction costs

Table 8: Trading with transaction costs

Matrics	CORN-D	UBAH	Best
Cumulative Wealth	133.6632	0.9064	1.5040
Sharpe Ratio	2.3935	0.2937	-0.2583
Maximum Draw Down	0.2660	0.6474	0.3935

To better reduce the impact of the transaction cost, we introduced two improvements to the above strategy.

2.2.1 Regularization for optimization

The first adjustment is adding a smoothing regularizer on the allocation weight to the objective function, (as the equation in part 2.3 algorithm) where λ is a regularization multipler that adds penalty to solutions with too much position change. The model serves to find the optimal weights that maximizes the past returns as a whole, meanwhile tradeoff between transaction cost and overall performance.

Here we used telescopic search to pick the best λ. We did two rounds search. First, we chose the period from 1 to 300 days as the validation data set with different values of λ =[0.05,0.1,0.15,0.2,0.25,0.30]. Comparing the best cumulative returns, we found that λ should between [0.05,0.1]. In the second round search, we did a more find-grained search around the best so-far, and we got the $\lambda = 0.08$.

With the selected multipler, we tested whether regularizer could make an improvement for or algorithms and the results for the test set is shown in Figure 4 and Table 9. And we can find that by adding regularizer into our algorithm, the performance is improved.

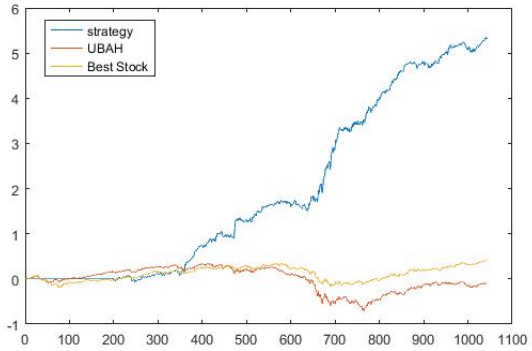

Figure 4: Log P&L when there are transaction costs and regularizer

Table 9: Trading with transaction costs and regularizer

Matrics	With no regularizer	With regularizer
Cumulative Wealth	133.6632	207.2189
Sharpe Ratio	2.3935	2.6987
Maximum Draw Down	0.2660	0.2036

2.2.2 Upper Bound Constraints

The second adjustment is lowering the upper bound of the maximum weight for each asset to reduce the variation of the allocation, and therefore saving the transaction costs. Also a lower upper bound encourages the diversification and reduces the risk of investing all wealth into one asset. We experimented different upper bounds ub = [0.6 ,0.7, 0.8, 0.9, 1] (see Table 10). And we found that larger the upper bound, the better all metrics performed. The result is kind of counter-intuitive. And one reason we think is because there are only 24 underlying assets in our pool. The benefit for diversification may not be significant. One the other hand, the benefits from lowering the transaction costs through setting the upper bounds is not enough to offset the losts stemming from changing the global optimal solution the the local optimal solution.

Table 10: Comparison of Performaces for Different Upper Bound

Upper Bound	0.6	0.7	0.8	0.9	1.0
Cumulative Wealth	37.7396	47.0352	77.4029	102.3717	207.2189
Sharpe Ratio	2.1383	2.0752	2.2491	2.3350	2.6987
Maximum Draw Down	0.2446	0.2438	0.2430	0.2532	0.2036

2.3 Backtesting with Tradable Assets

In the above analysis, we focused on the MSCI indices. However, indices are not tradable, making our algprithms less realistic. Thus in this part, we include a new data set with 15 tradable ETFs to test the robustness of our strategy. Because the new data set nearly lie in the same period of the MSCI data set, we assume all the chosen parameters would probably remain the same. And test whether our algorithms can work out.

2.3.1 Trading without Transaction Costs

Firstly, we consider the situation when there is no transaction costs. The result is shown in Table 11 and Figure 5. We can find our strategy can still be profitable, however, the risks has also been increased and thus the sharpe ratio becomes less. And actually the overall performance is much lower than our previous MSCI dataset.

Table 11: Trading with transaction costs

Matrics	CORN-D	UBAH	Best
Cumulative Wealth	16.3329	1.4007	2.8488
Sharpe Ratio	1.3553	0.0619	0.5680
Maximum Draw Down	0.3156	0.5147	0.2923

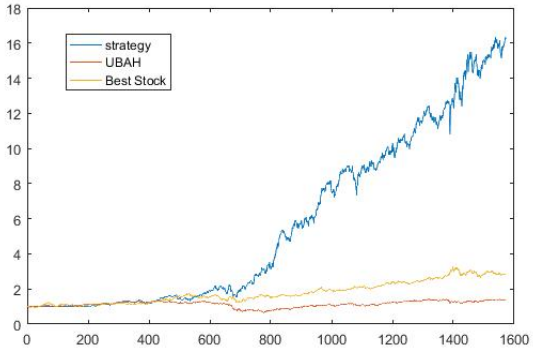

Figure 5: Log P&L when there are transaction costs and regularizer

2.3.2 Trading with Transaction Costs

The result became even worse when we included transaction costs. The strategy cannot take back the initial investment after all six years. Thus we revised our strategy. With all the parameters unchanged, we changed the threshold parameter from 0 to 0.6. And when there is less than 2 similar windows chosen, we would not change our position. And the result is shown in Figure 6.

From the figure, we can find out although the performace is improved by this new revised strategy. Our strategy does no much better than the benchmark(we only use one benchmark here, uniform buy and hold strategy). Thus, our strategy perform poorly when transaction costs included.

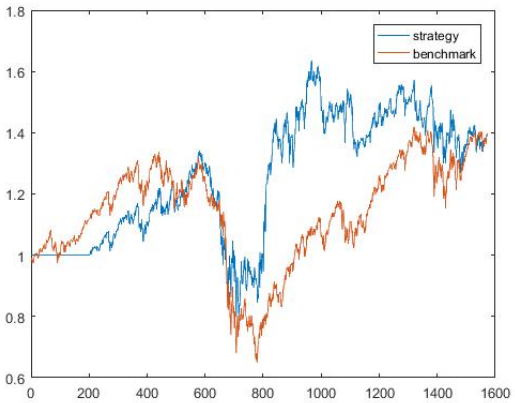

Figure 6: Log P&L when there are transaction costs and regularizer

Part IV

Conclusion

Online portfolio selection has been a hot topic recently. In our paper, we made revisions to one exsiting strategy, correlation-driven nonparametric learning (CORN) method and built up our correlation-driven nonparametric learning decay (CORN-D) method. Applying our model to both stock datasets and index datasets, we find our strategy performed well for indices because they are more possible to maintain the pattern. We also tested different parameters and different methods for our strategy and picked up the most efficient combination. Then we took the transaction costs into consideration and came up with methods like regularization to lower the effects. In the last, we applied our strategy to tradable ETF data with chosen parameters and methods and found out our strategy performed well when there are no transaction costs but performed poorly when transaction costs included.

Through our analysis, we also found out that our strategy performed well when the market is volatile. Thus it leads us to think whether it would be possible to combine algorithms suitable for different market conditions to achieve higher returns. We also find out transaction costs would badly hurt our algorithms and thus figuring out ways to control transaction costs would be another importent issue for our strategy.

References

[1] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[2] John L Kelly. A new interpretation of information rate. *bell system technical journal*, 35(4):917–926, 1956.

[3] Bin Li and Steven CH Hoi. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):35, 2014.

[4] Bin Li and Steven Chu Hong Hoi. *Online Portfolio Selection: Principles and Algorithms*. CRC Press, 2015.

[5] László Györfi, Gábor Lugosi, and Frederic Udina. Nonparametric kernel-based sequential investment strategies. *Mathematical Finance*, 16(2):337–357, 2006.

[6] László Györfi, Frederic Udina, and Harro Walk. Nonparametric nearest neighbor based empirical portfolio selection strategies. *Statistics & Decisions International mathematical journal for stochastic methods and models*, 26(2):145–157, 2008.

[7] Bin Li, Steven CH Hoi, and Vivekanand Gopalkrishnan. Corn: Correlation-driven nonparametric learning approach for portfolio selection. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):21, 2011.