POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Hyperparameter Tuning for Pairs Trading: an Online Learning Approach

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Nahuel Coliva

Advisor: Prof. Marcello Restelli

Co-advisors: Luca Sabbioni, Pierre Liotet, Lorenzo Bisi

Academic year: 2021-2022

1. Introduction

Pairs trading is a statistical arbitrage which tries to exploit market inefficiencies that lead to over/underpricing of assets. Specifically, two assets are selected if their movements are similar, with respect to some metric, during the *pair formation* period: in the following period, the *trading* one, the arbitrage waits for the two assets price difference to deviate from the expected trend. Once a profitable threshold is reached, the strategy prescribes to bet on the mean-reversion of the two assets spread, which is likely to happen due to their historical movement similarity, achieving returns.

A state-of-the-art implementation for pairs trading is [7], which relies on the application of Principal Component Analysis (PCA) and clustering in order to reduce the computational effort required to identify suitable subsets of assets. Then, the pair selection procedure would be run considering only pairs of assets that belonged to the same subset. In order to optimise the hyperparameters of their pipeline, we apply an online learning approach: in particular, we handle each parametrization as an asset (called *expert* for the rest of the document) and optimise our budget distribution over them. Thus, we are in the context of the *online portfolio optimisation* framework: among the available algorithms, we will apply Online Gradient Descent with Momentum (OGDM, [8]), not only because of its theoretical guarantees on both the regret bound and the computational complexity, but also because of its empirical results on dealing with transaction costs. The experiments are performed both with and without transaction costs, while also measuring the impact of the rebalancing interval. The remainder of the document is structured as follows: in Section 2 we recall some mathematical background, followed by Section 3, where we provide an in-depth description of [7] methodology and elaborate on which hyperparameter is worth the optimisation. In Section 4 we present the details of the optimisation framework and experiments performed, while in Section 5 an overview of the related works can be found. Section 6 reports the exploratory analysis of the dataset we are dealing with and then, in Section 7, the performance of the OGDM optimisation is shown: it achieves good results without considering transaction costs, while the pairs trading itself seems to struggle when costs are factored in. Finally, in Section 8 we draw the conclusions and present possible directions for future research.

The present work is based upon the results of the cooperation between Politecnico di Milano and MDOTM.

2. Preliminaries

General Statistics.

Definition 2.1. (Integration and Co-integration) *We will call the time series* $X_t = [x_1, \dots, x_T]$ integrated of order 1 *if the process* $X_t - X_{t-1}$ *is stationary and we will denote it as* $X_t \sim I(1)$. *Moreover, given another time series* $Y_t \sim I(1)$, *we will call the pair* co-integrated *if there exists a coefficient* β *such that their difference* $U_t = Y_t - \beta X_t$ *is stationary, that is,* $U_t \sim I(0)$.

Notice that, when two time series are co-integrated, not only their spread will have constant mean, but it will also be a mean-reverting process. Another interesting property of mean-reverting processes is the Hurst exponent, here reported in the generalized version (as suggested in [1]).

Definition 2.2. (Generalized Hurst Exponent) *Given a time series* $X_t = [x_1, \dots, x_T]$, *consider the statistic* $K_q(\tau)$ *defined as*

$$K_q(\tau) = \sum_{t=0}^{T-\tau} \frac{|x_{t+\tau} - x_t|^q}{(T - \tau + 1)}, \tag{1}$$

which can be interpreted as the q-order moment of the distribution of the increments with τ *lag. We will call* generalized Hurst exponent $H(q)$ *the one which approximate* $K_q(\tau) \approx c\tau^{qH(q)}$ *the best. In particular, we will refer to* $H(2) = H$ *as the* Hurst exponent, *since its estimate is comparable to the original one.*

Notice that this parameter gives insight on the evolution of the process: in fact, $H \in [0, 0.5)$ suggest a mean-reverting trend, while it is likely to be divergent when $H > 0.5$. Finally, values around 0.5 suggest a Brownian motion.

Definition 2.3. (Mean-Reversion Half-Life) *Given an Ornstein-Uhlenbeck process defined as*

$$dX_t = \alpha(\mu - X_t)dt + \sigma dW_t, \tag{2}$$

where $\alpha > 0$ *and* W_t *is the Wiener process, we will call the time needed (on average) to halve the distance from its historical mean as* mean-reversion half-life $t_{1/2}$. *Such a quantity can be obtained starting from the ordinary differential equation derived from the process equation* $\dot{x} = \alpha(\mu - x)$, *yielding* $t_{1/2} = \frac{log2}{\alpha}$.

Online Portfolio Optimisation. We will deal with *online portfolio optimisation (OPO)* framework: given a set of N tradable assets (stocks, in our case, to which we will also refer as tickers), we will call the sequence $r_t = [r_{1,t}, \dots, r_{N,t}]$ *price relatives* at time t, where $r_{1,t} = \frac{p_{i,t+1}}{p_{i,t}}$ and $p_{i,t}$ is the price of asset i at time step t. Then, the objective of the OPO framework is to allocate its resources as the vector $x_t = [x_{1,t}, \dots, x_{N,t}]$, trying to maximise the total wealth obtained, called *cumulative wealth* and defined as

$$W_T(x_{1:T}, r_{1:T}) = \prod_{t=1}^{T} \langle x_t, r_t \rangle, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. In order to face a simpler problem, we can rewrite the objective function as a convex loss function: following [8], the chosen one for our setting is

$$f_t(x_t) = -log(\langle x_t, r_t \rangle), \tag{4}$$

and our new objective will be its minimisation.

2.1. Pairs Trading

Pairs trading is a market neutral trading strategy composed of two-steps:

1. firstly, a pair of assets is selected, according to some criteria, so that they are strongly correlated during a window that is called *pair formation period*;
2. secondly, the pair is traded during the *pair trading period*, speculating on the fact that the price spread between the two assets will converge to its historical mean.

The technique is thus based on the assumption that the two assets in a selected pair are structurally related to each other: their difference in price will be stable over time and the profits will come from the exploitation of market inefficiencies.
Effectively, these inefficiencies come in the form of an overpriced asset and an underpriced asset: the strategy then prescribes to sell short the former and to buy the latter one. Thresholds are usually defined (statically or dynamically) in order to decide when to open and close the position (long-short or vice versa).

Although many metrics can be used to select related pairs, as reported in [6], we will focus on the cointegration relationship (see Definition 2.1), which has both econometric foundations (the spread is expected to reverse to the mean) and empirical good results.

3. Sarmento&Horta Pair Selection Pipeline

The main idea of the authors was to deploy an unsupervised learning algorithm in order to reduce the computational effort of an exhaustive pairs search: by grouping the assets in clusters, the possible generated pairs drop considerably, with a largely effective reduction in the computational costs. Unfortunately, this comes with an issue: clustering techniques are well-known to struggle when applied to high dimensional data (namely, our assets price series).

Principal Component Analysis (PCA). To overcome the aforementioned issue, the authors applied PCA to the normalized return series $r_{i,t}$, that is:

$$r_{i,t} = \frac{p_{i,t} - p_{i,t-1}}{p_{i,t-1}}. \tag{5}$$

In particular, the authors chose to keep the 5 most explicative features: thus, assuming that there are N assets with T prices each, the dataset got restricted from $N \times T$ to $N \times 5$ in their experiments.
Let us notice that the number of features was selected empirically in [7], quoting: "We adopt 5 dimensions since we find adequate to settle the ETFs' representation in a lower dimension provided that there is no evidence favouring higher dimensions".

Clustering: OPTICS. OPTICS is a clustering algorithm which was created to fix some of the weaknesses of DBSCAN: in particular, the latter searches the data space for areas where data points are more dense. However, having a fixed concept of density (defined as having at least *MinPts* points in an ε-radius area) it fails in dataset with different densities. To overcome this issue, OPTICS dynamically selects ε inside the feature space, so that sparser regions have a broader concept of core points (that is, the main data points of a cluster).

Once transformed with PCA, data are fed to OPTICS which, in turn, outputs the asset labels, including the possibility to mark as noise. Among the pairs within the same cluster, the candidates for trading are then selected according to a set of hand-crafted rules.

Selection Rules.

1. first of all, pairs should be cointegrated: the authors propose the Engle-Granger test, due to its simplicity;
2. since the mean-reversion property of the spread series is of crucial importance, pairs should also have a Hurst exponent $H < 0.5$;
3. mean-reversion half-life is also taken into consideration: since we would like to generate profit in a reasonable time lapse, pairs with mean-reversion half-life less than a day and more than a year are discarded;
4. finally, to assure that there are enough profit opportunities, the spread process of each pair should have crossed its mean at least 12 times each year in the pair formation period.

3.1. Optimising the Pipeline

As explained in the previous sections, the framework from [7] has several hyperparameters that could possibly be tweaked to achieve better performance. In particular, the list boils down to:

1. length of the pair formation period;
2. number of features selected by PCA;
3. MinPts OPTICS parameter;
4. p-value threshold for the Engle-Granger test;
5. Hurst exponent upper bound;
6. mean-reversion half-life bounds;
7. minimum number of average mean-crossing per year.

To understand whether the impact of such hyperparameters were worth the optimisation, we performed a grid search exploration, paying attention to the sensitivity of an objective function. In the following, the main steps are reported:

- for each parametrization x in the grid, the *feedback function* $f(\boldsymbol{x})$ is defined as the mean daily return of the selected pairs over a two year rolling window, that is

$$f(\boldsymbol{x}) = \frac{1}{720} \sum_{t=1}^{720} \frac{daily\_return_t}{n\_selected\_pairs_t}; \tag{6}$$

- in order to consider only the most promising parametrizations, the analysis is then reduced on

$$\bar{X} = \{\boldsymbol{x} \in X | f(\boldsymbol{x}) > \frac{f_{max} + f_{min}}{2}\}, \quad (7)$$

where f_{max} and f_{min} are the maximum and minimum, respectively, of function $f(\boldsymbol{x})$;

- as far as one dimension of $\boldsymbol{x}$, said h, is considered (i.e. a single hyperparameter), the goal is to assess how narrow is the range in which the best feedback values are found: hence, for each tuple of values (h_1, h_2) available in the grid, we define the *range feedback* $g(h_1, h_2)$ as

$$g(h_1, h_2) = \mathbb{E}[f(\boldsymbol{x}) | \boldsymbol{x} \in \bar{X}, \boldsymbol{x}^h \in [h_1, h_2]], \quad (8)$$

where $\boldsymbol{x}^h$ is the value of hyperparameter h in parametrization $\boldsymbol{x}$. $g(h_1, h_2)$ represents the average of function $f(\boldsymbol{x})$ among parametrizations that have $\boldsymbol{x}^h$ in the considered range.

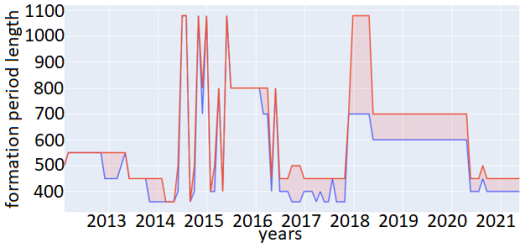
By plotting the range $[h_1, h_2]$ that attains the maximum over $g(\cdot, \cdot)$ and studying how it varies, we can get a deeper understanding of how much tuning the parameter may influence the final result: narrower intervals suggest high sensitivity (selecting the correct value is crucial to achieve almost-optimal performance), while, in turn, wider ones suggest low sensitivity (being in a neighbourhood of the optimal value is already enough).

The dataset is the S&P 500 daily data from 1st January 2010 to 12th June 2021 and in Section 6 you can find an in-depth analysis of the clusters identified during the pair formation step with the parametrization suggested in [7].

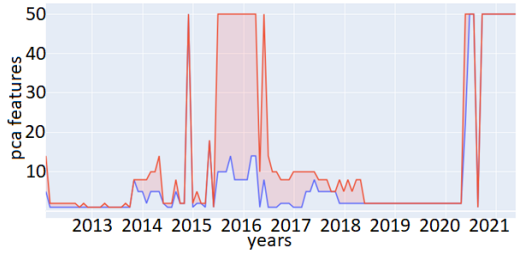
We found that we could divide the hyperparameters into two categories:

- the first one, including the hyperparameters whose plot is reported in Figure 1, is quite promising; in fact, optimal areas are usually narrow and, moreover, their included values change over the considered period;
- on the other hand, the other ones are characterized by wide areas, generally the whole set of tested values, suggesting that the exact value is of little importance to the overall final result; being them and their plot of little information, they are here omitted.

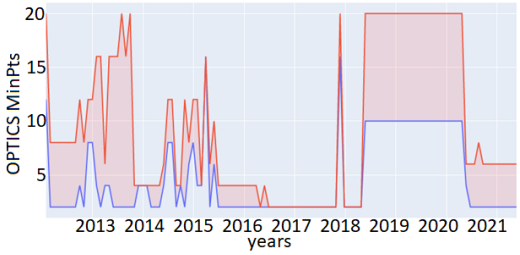
Therefore, the hyperparameters grid that will be considered during the rest of the document is the one presented in Table 1.



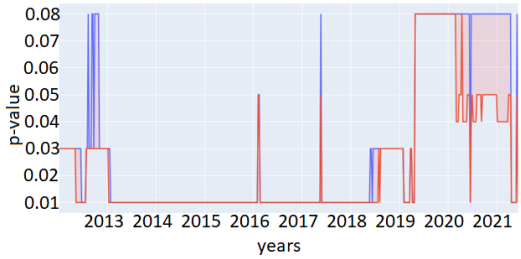
(a) Formation period length.



(b) PCA features.



(c) OPTICS MinPts.



(d) p-value threshold.

Figure 1: Hyperparameters optimal areas.

4. Online Optimisation Framework

In order to both have a good feedback frequency and keep the trading period of optimal length, we would like to be able to receive feedback as the positions are opened/closed: in this way, an expert learning algorithm could find the best allocation of the available budget to each one of

Hyperparameter name	Values
formation period length (in days)	360,400,500,600,700, 800,1080
PCA selected features	1,2,4,10,14,20
OPTICS MinPts	2,4,6,8,10,12,16,20
p-value threshold	0.03,0.05,0.08

Table 1: Final hyperparameters optimisation grid.

the considered parametrizations (i.e. the experts) of the underlying pairs trading strategy. However, such an idea comes with an obstacle: we have to keep track of the pairs selected by each expert, since they will be inherited by the same one in the next time step. Indeed, such a solution would bound together the performance of an expert to its choices in previous time steps, which is outside of the expert learning setting, where prediction and evaluation are somewhat independent.

Instead, the online portfolio optimisation framework looks more appropriate: here, we represent each expert as an asset, with its price variation due to the performance achieved by the trading strategy. Even though the presence of transactions costs still bounds an expert to its previous choices, the effect is here mitigated by computing the fees in an aggregated way: in fact, where moved money are only a matter of internal computation (i.e. a position is now managed by expert A instead of expert B), no fees are payed. Hence, we are improving with respect to the expert learning setting without introducing the complexity of a reinforcement learning approach.

4.1. Dealing with Transaction Costs: OGDM

In order to keep the transaction costs low, we chose Online Gradient Descent with Momentum (OGDM, [8]) as the optimisation algorithm, which provides also some interesting properties:

- it provides good guarantees on the regret, that is, the difference of $log(W_T(\cdot,\cdot))$, w.r.t. the best constantly rebalanced portfolio: it scales, both theoretically and empirically, as $\mathcal{O}(\sqrt{T})$;
- it is computationally efficient, scaling well both on the time horizon and on the size of the portfolio, with a per-iteration complexity of $\Theta(N)$ number of assets.

Moreover, the only assumption needed by the algorithm is the following basic one:

- there exist two finite constants ϵ_l, $\epsilon_u \in \mathcal{R}^+$ such that $r_{i,t} \in [\epsilon_l, \epsilon_u]$ with $0 < \epsilon_l \leq \epsilon_u < +\infty$, $\forall i \in [1, N]$, $\forall t \in [1, T]$.

The algorithm is described in the following:

Algorithm 1 OGDM(H,Λ)

input: learning rate sequence $H = \{\eta_1, \ldots, \eta_T\}$, momentum parameter sequence $\Lambda = \{\lambda_1, \ldots, \lambda_T\}$
Initialize $x_0 \leftarrow (0, \ldots, 0)$ and $x_1 \leftarrow (\frac{1}{N}, \ldots, \frac{1}{N})$
for $t \in 1, \ldots, T$ **do**
  Receive r_t from the market
  Store the obtained wealth $\langle x_t, r_t \rangle$
  Compute the new budget distribution: $x_{t+1} \leftarrow \prod_{\Delta_{N-1}} \left(x_t + \eta_t \frac{r_t}{\langle r_t, x_t \rangle} - \frac{\lambda_t}{2}(x_t - x_{t-1}) \right)$
  for $i \in 1, \ldots, N$ **do**
    Update expert i pairs portfolio $\mathcal{K}_{i,t+1}$
    Distribute budget $x_{i,t+1}$ evenly among the pairs belonging to $\mathcal{K}_{i,t+1}$
  end for
end for

Let us notice that $\prod_X(y) = \arg\inf_{x \in X} ||y - x||_2^2$ is the standard projection of the vector y onto X and hyperparameter sequences H, Λ will be constant and optimized through a validation period: see Section 4.3 for more details.
Indeed, the trickiest part of the algorithm is the computation of r_t, which is reported next:

$$r_{i,t} = 1 + \frac{1}{K_{i,t}} \sum_{k \in \mathcal{K}_{i,t}} g_{k,t}, \tag{9}$$

where $\mathcal{K}_{i,t}$ is the set of the $K_{i,t}$ tradable pairs for expert i at time step t and $g_{k,t}$ is the percentage gain on pair k at time step t; hence, $g_{k,t} = 0$ if the amount of money did not change once the position was closed or the pair was not opened in the first place.

Notice that we do not force any position to close before rebalancing, although we do compute $g_{i,k,t}$ with the actual available money (thus, in a certain sense, as we closed the position just before

rebalancing). Another thing to notice is that each expert i will distribute its budget $x_{i,t}$ uniformly over its pairs: this procedure is performed at rebalance time and may be needed even when $x_{i,t} = x_{i,t+1}$. Finally, in order to keep the computation straightforward, the update of the budget distribution provided by OGDM is synchronized with the update of the pairs in each expert portfolio.
It is also worth mentioning that the transaction computation will be slightly different from [8]; namely, here we perform a transaction only if:

- a position is opened/closed;
- the OGDM update involves moving money from or to an expert which has still open positions: in fact, upon receiving a budget increase, the amount received is immediately invested in open positions; similarly, it may happen that experts that see their budget decreased have to partially close open positions.

4.2. Pairs Portfolio Management and Trading Strategy

Pairs Portfolio Management. Each expert has its own portfolio of tradable pairs obtained *incrementally* from the various pair selection steps. Specifically, the common policy for inserting/removing pairs is given in the following:

- at each rebalancing step, the pairs selection algorithm is run, but only 5 pairs can be added to the expert portfolio: namely, the top 5 pairs in order of cointegration p-value significance (the lower, the better);
- we will call the number of pairs that are dropped in favor of new pairs *turnover* and will cap it at 5;
- the maximum portfolio size is 30 and they are dropped in order of returns: thus, in the case of a complete turnover, the dropped pairs would be those that have generated the lowest return.

Trading Strategy. The main steps of the trading strategy are:

1. for each pair, extract the spread time series of the pairs selection period, that is

$$S_t = X_t - \beta Y_t, \tag{10}$$

   where X_t and Y_t are the two assets series, while β is the cointegration coefficient;
2. compute its mean μ and its standard deviation σ;
3. during the trading period, monitor the value of the spread s_t: a position is opened (investing an equal amount of money on the two assets) when $|s_t - \mu| > 2\sigma$, while it is closed when $|s_t - \mu| < \frac{\sigma}{2}$.

Additionally, the strategy employs a stop-loss threshold: when a pair return goes below -7%, it is closed and dropped from the expert portfolio.

4.3. Experimental Framework

We will now summarise the main features of the framework adopted, along with some useful details:

1. about the generation of the experts: due to computational issues, not all the possible combinations of hyperparameters have been considered; nonetheless, 428/1008 experts already give an idea of the range of optimisation;
2. for each expert, the pair selection step is the one from [7] with the expert own parametrization, while the trading strategy is common to all of them, as it is described in the previous Section;
3. since the time interval between OGDM updates influences not only the frequency of the OGDM feedback but the whole problem too, we decided to test different values for it; specifically, the *rebalancing intervals*, in days, were selected in the set $\{7, 14, 21, 42, 63\}$, which roughly translate to a week, three weeks, a month, two months, three months respectively (due to the absence of Saturdays and Sundays in the dataset).

One of the baselines that will be used to assess the performance of OGDM is the one of *independent experts*: it is obtained by splitting the initial budget to each expert as in OGDM, but here they trade without the possibility to move money from one to another. Given this definition, the experiments run are of two types:

1. for each rebalancing interval, we perform a validation-test split $(25\% - 75\%)$ in order to extract the best values for H and Λ among $[0, 0.05, 0.1, 0.5, 1, 5, 10, 20]$: then, the performance of OGDM with the resulting parameters is compared with the average performance of independent experts and the

performance of the two most similar available experts to the parametrization of [7]; notice that here we do not account for transaction costs;
2. we repeat the experiments to assess the impact of considering them: in particular, we kept the optimal values for H and Λ found previously and run the experiments on the whole dataset; this time, the benchmark is just the average performance of independent experts, transaction costs included.

For both of them, we considered the same dataset as in 3.1, but from 3rd January 2000 to 31st December 2021.

5. Related Works

The documents available in literature can be roughly divided into two categories: those applying machine learning to pairs selection and those applying it to the trading phase. Starting from the former, we have the already cited [5], where machine learning models try to predict whether an asset will perform better or worse than the market: then, assets are ranked and pairs are generated by associating the k-th asset with the (n-k)-th (thus, the expected best with the worst and so on). Instead, the concept of cointegration is expanded in [2] with the introduction of Partial Co-Integration (PCI), a similar relationship which takes also into consideration a random walk component in the spread series.

The machine learning applications to the trading phase include trying to predict the asset trends (as already reported in, e.g., [5]), dynamically decide the opening/closing/stop-loss thresholds, as in [4], or directly the actions to be performed, as in [9].

6. Exploratory Analysis

Before running the experiments described in Section 4.3, some more exploratory analysis have been conducted. Specifically, we considered the same 1st January 2010 - 12th June 2021 period as in Section 3.1 and extracted some statistics about the clusters and pairs selected by the [7] pipeline.

6.1. General Aspects

For what general statistics are concerned, the average number of clusters in each pairs selection period is around 100, while their average size is stable at around 2.7 elements per cluster. Moreover, the number of noise points in each period (i.e. the points that were not identified as belonging to any cluster) was almost half of the dataset, namely around 200 elements per period. We can further explore the size of the clusters by plotting the distribution of the sizes, an example of which can be seen in Figure 2: the most prominent column is the one for size 2, representing almost half of all the clusters, while the distribution steadily decreases as the size increases.

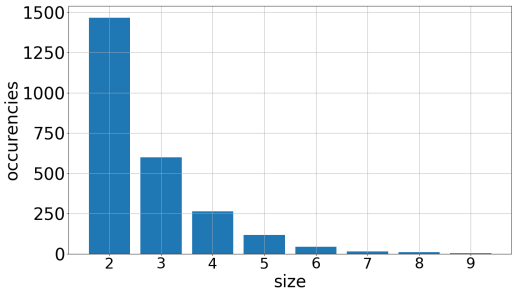
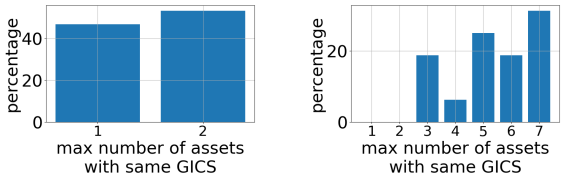


Figure 2: Cluster size frequencies across periods: the majority of the clusters has 2 or 3 elements, exponentially decreasing as the size increases.



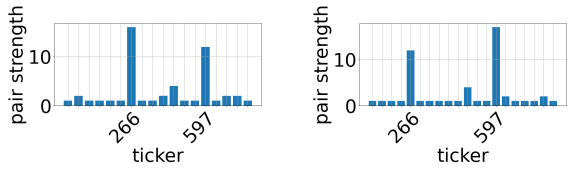
(a) Size 2. (b) Size 7.

Figure 3: Two examples of percentage of clusters w.r.t. maximum number of elements with same GICS code.

6.2. Consistency Analysis

In order to assess the consistency of the clusters, the first and most trivial analysis is related to the GICS classification: it is an industry taxonomy to classify companies on the basis of their sector, that is, same GICS code means same industry group. Hence, in Figure 3 we can see two meaningful examples of the percentage of clusters w.r.t. their maximum number of elements with the same GICS code inside them.

What we find out is that bigger clusters tend to be more consistent with the GICS classification, having the highest bars in the right part of the



(a) Pair strengths from asset #266 point of view. (b) Pair strengths from asset #597 point of view.

Figure 4: Example of symmetric pair.

graph, so more elements with a common GICS code. On the other hand, smaller clusters have a more uniform-like distribution.

Since OPTICS identified small clusters, the difference between a cluster and the pairs generated from it is narrow, at least most of the time. Therefore, we thought about looking for tickers (i.e. stock identification codes) that were in the same cluster most of the times. That is, if clustering actually helps and the relationships found are not spurious, it should consistently group together similar tickers. Thus, some definitions will be useful going forward:

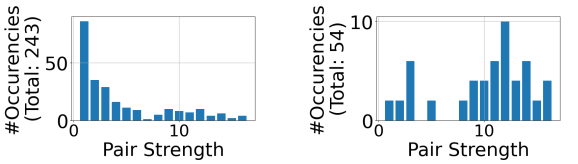
- pair strength: number of times two assets were clustered together;
- cluster time: number of times an asset belong to a cluster (with at least two elements);
- consistent pair: (x,y) is a consistent pair from x's point of view if

$$PairStrength(x,y) > \frac{PairStrength(x,x)}{2}, \tag{11}$$

  where x and y are two assets; this means that we consider x and y consistent w.r.t. x if at least half of the time x belongs to a cluster, also y is in the same cluster: thus it is not a symmetric relationship;
- symmetric (consistent) pair: when both (x,y) and (y,x) are consistent pairs.

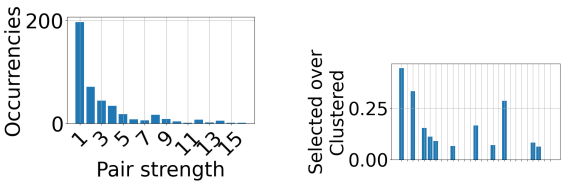
For example, assets #266 and #597 are in a symmetric pair, since their pair strength is greater than both their $0.5 \cdot ClusterTime$, as can be seen from Figure 4.

Let us acknowledge that such a measure of consistency is subject to outliers: the most trivial example is when an asset is clustered just a single time. In order to take care of such possible source of noise, we took a look at the pair strength distribution for consistent pairs: in Figure 5a we



(a) Consistent pairs strength distribution. (b) Symmetric consistent pairs strength distribution.

Figure 5: Consistent vs Symmetric consistent pairs strength distributions.



(a) Selected pairs strength distribution. (b) Selection percentage of symmetric pairs.

Figure 6: Symmetric pairs are not a remarkable portion of the selected ones.

can see that these “outliers" are actually a third of the total. Moreover, we can notice that the distribution is decreasing until 8, but then it starts a singular behaviour. Plotting the same graph but for symmetric pairs, which can be seen in Figure 5b, we understand that a lot of them are in the $[8, 16]$ interval: thus, they generate the unusual growth seen in the previous graph. Notice that we were exactly looking for these pairs: assets that are not only frequently clustered, but also often with the same neighbour (at least one). Therefore, the next question arises naturally: are symmetric consistent pairs (which should be the most similar pairs from a structural point of view, as suggested by the clustering) also the selected pairs? Unfortunately, they were not, as can be seen at the end of Section 6.3.

6.3. Selected pairs Analysis

In Figure 6a the pairs strength frequencies for selected pairs is shown: it is a similar distribution to the one shown in Figure 5a, although we miss the anomalous region after 8, since symmetric pairs are almost never selected. Therefore, it seems that clustering helps in reducing the computational effort required to search for promising pairs, indeed, but the cointegration test, along with the other rules, are still crucial to obtain robust mean-reverting spread series.

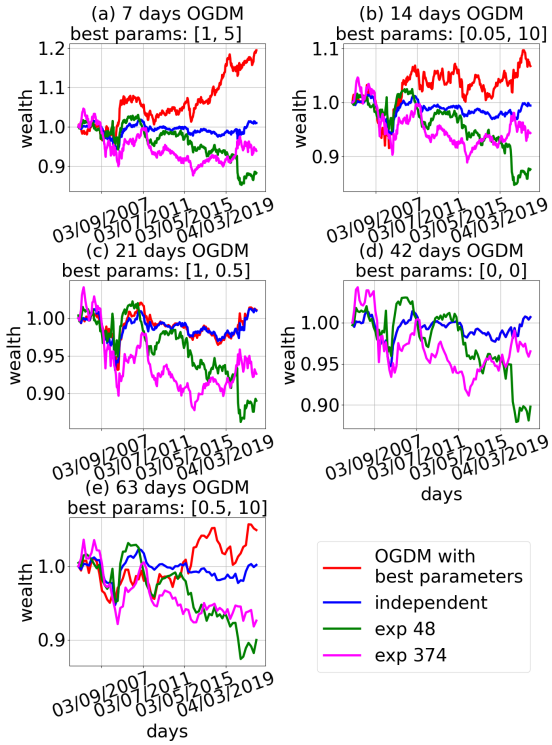


Figure 7: Total wealth obtained in the test set by OGDM with best parameters, divided by rebalancing interval and compared w.r.t. the independent experts and the two most similar experts to the suggested parameters of [7].

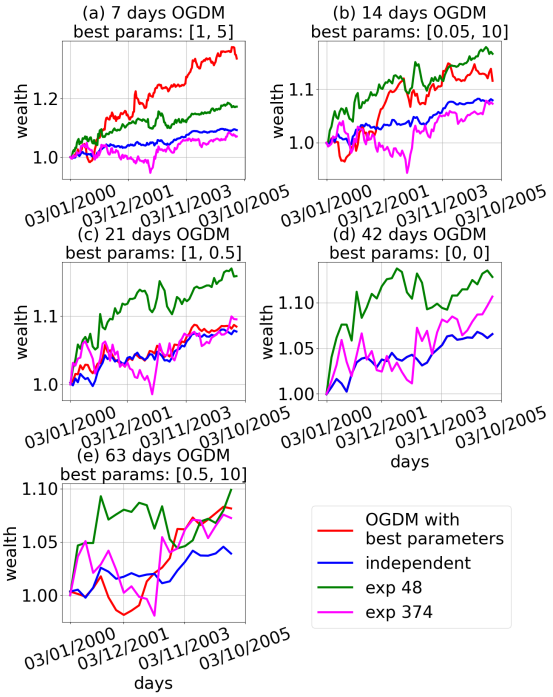


Figure 8: Total wealth in the validation set.

7. Experimental Results

7.1. Transaction Costs-Free

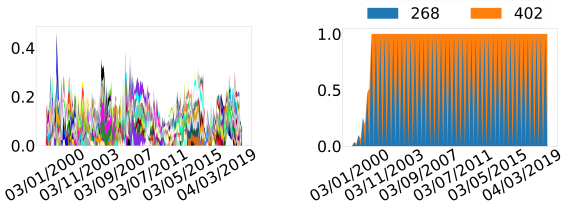
In Figure 7 we show the cumulative wealth obtained by OGDM on the test set, with the parameters selected on the validation dataset, along with two baselines: the first one is the sample mean obtained from the *independent* experts from the grid, while the second one are the two closest available experts to the trading scenario proposed in [7]. For what concerns the latter, the suggested parametrization of [p-value,PCA features,MinPts,formation period] would be $[0.05, 4, 2, 520]$: in order to provide a fair comparison with respect to OGDM, we will consider expert 48 with $[0.03, 4, 2, 500]$ and expert 374 with $[0.05, 4, 4, 400]$ as approximations, since they are the most similar ones inside the set of experts considered for OGDM application. Let us observe that the algorithm was not initialized again once the test started, in order to stick to a realistic scenario.

Notably, as depicted in Figure 7, performance varies significantly on the rebalancing interval basis: in fact, while good optimization was achievable for 7-14-63 days, 21 and 42 yielded almost no improvement over the independent approach. One of the reasons is that, along with the rebalancing interval, the underlying problem changes too: indeed, allowing pairs to be traded for a longer period of time may allow for greater gains/losses, while diminishing the frequency of the OGDM updates may result in missing trading opportunities. These two facts are in a trade-off, thus exploring the rebalancing interval impact was one of the purposes of this research.

Even though [7] parameters performed poorly during the test set, they would have been a strong choice during the validation period: as we can see from Figure 8, expert 48 yields the best results, except for (a) where it is still the best baseline. This fact remarks the importance of a dynamic strategy: the best pairs trading parametrization changes over time, as already reported in Section 3.1, thus trying to adapt as information is available yields better long-term gains.

7.2. Budget Distribution Analysis

Another interesting quantity that can be extracted form OGDM results is the budget distribution, that is, how money is distributed among experts during the considered time span. In order



(a) 21 days interval. (b) 63 days interval.

Figure 9: Budget distribution for the transaction costs-free experiments (top 10 heaviest experts). Notice how (b) oscillates between two experts after the first time steps.

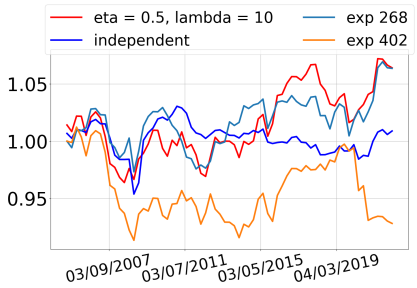


Figure 10: Total test wealth obtained by OGDM with best parameters and 63 days of rebalancing interval, compared with the average of independent experts and the main experts involved.

to provide a useful visualization, since plotting the budget line of each expert would be unfeasible, we decided to show only the 10 experts with the largest weights (approximately 2.5% of the total) at each time step. In Figure 9, we show the main two kinds of behaviour:

- the expected, chaotic distribution shown in Figure 9a, where we chose not to report the legend due to its dimension: in fact, a lot of different experts enter and exit the top 10, suggesting that the algorithm is rapidly adapting to the environment feedback; however, the overall distribution is still quite uniform, thus the gains are similar to the independent baseline;
- the other rebalancing intervals tell a different story: for example, in Figure 9b we report the results for 63, where OGDM is able to converge to a specific set of well-performing experts. However, the budget allocation degenerates to an oscillatory behaviour, resulting in moving the whole budget alternatively from an expert to another one.

In order to better understand such a swinging behaviour, in Figure 10 we report the cumulative wealth of OGDM alongside the main experts where the budget was allocated. Then, we can understand that the budget was generally split among two experts: one that performed way better than the average and one that is subject to less variance but is actually below average. Hence, we are likely facing some kind of overfitting: in fact, the three configurations share a high value of the momentum parameter, resulting in weighing the first updates way more than the following ones. Despite this undesired phenomenon, the performance on unseen data were good, with all three parametrizations performing quite better than both the average and the [7] suggested parameters.

7.3. Experiments with Transaction Costs

In the following tests, parameter γ took the values $[0.001, 0.005, 0.01]$, which approximately translate to low, medium and high transaction costs environments.

The cumulative wealths from the experiments with transaction costs are reported in Figure 11: as we can see, both the naive independent implementation and the OGDM implementation would suffer in a real world scenario. Due to computational issues, it was not possible to obtain the curves of the experts similar to [7]: still, we can safely assume that introducing transaction costs would not improve the performance with respect to the independent average.

Despite the difficulties when dealing with transaction costs, let us remark the good performance of OGDM in Figure 11e: despite the additional price of the optimisation (i.e. more transactions) there is a clear improvement in the profit when transaction costs are low.

8. Conclusions

We provided an extension to the work of [7] by analyzing which hyperparameters could be suitable for optimisation in a daily stock market context, namely, the S&P 500 one. Furthermore, we analysed the clusters obtained by the default [7] parameters, discovering that selected pairs were not consistent with the clustering. Then, we actually performed the optimisation of the most

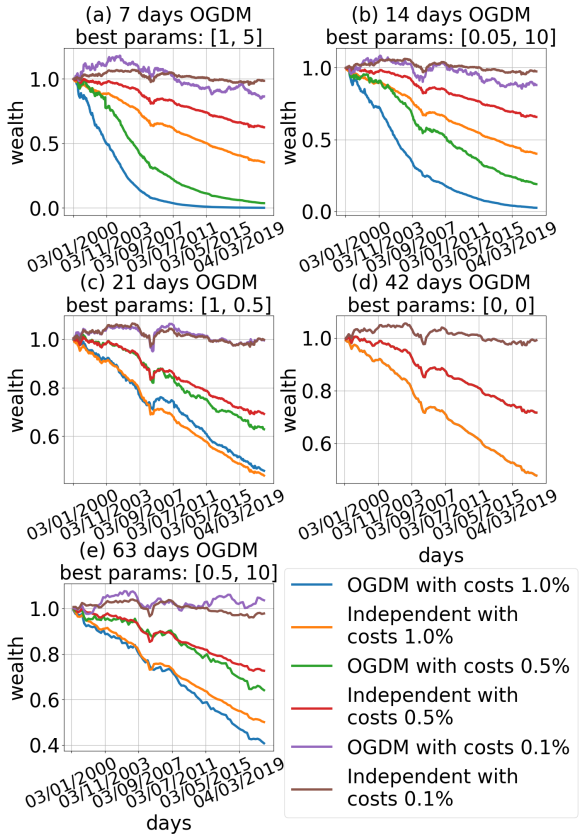


Figure 11: Total wealth obtained on the whole dataset by OGDM with best parameters, divided by rebalancing interval, for different costs of transaction and compared w.r.t. the average of independent experts.

suitable parameters through the OPO framework and the OGDM algorithm: the suggested parameters from [7] showed to be suboptimal for each one of the rebalancing intervals tested, while OGDM managed to outperform both the suggested parameters and the average independent expert when transaction costs were not involved.

Moving on to the experiments that included transaction costs, we found that the whole strategy suffers: despite OGDM had the worst performance lines, possibly due to its increased transaction rate, the results obtained with independent experts are far from generating profit too.

Many different directions could extend the presented work: starting from the expert generation, some metric of similarity could be employed to apply OGDM on a restricted selection of parametrization, possibly discarding similar ones in terms of both performance and hyperparameters used. The underlying data play a leading role too: allowing pairs to be more complex than two assets (e.g. an asset vs the other components of its cluster) may identify more stable relationships. A similar idea was already proposed in [3], thus applying our framework on top of their procedure may yield good results.

References

[1] Barunik, J. and Kristoufek, L. (2010). On hurst exponent estimation under heavy-tailed distributions. *Physica A: Statistical Mechanics and its Applications*, 389(18):3844–3855.

[2] Clegg, M. and Krauss, C. (2018). Pairs trading with partial cointegration. *Quantitative Finance*, 18(1):121–138.

[3] Galenko, A., Popova, E., and Popova, I. (2012). Trading in the presence of cointegration. *The Journal of Alternative Investments*, 15(1):85–97.

[4] Kim, T. and Kim, H. Y. (2019). Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. *Complexity*, 2019.

[5] Krauss, C., Do, X. A., and Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, 259(2):689–702.

[6] Krauss, C. et al. (2015). Statistical arbitrage pairs trading strategies: Review and outlook. *FAU Discussion Papers in Economics*, (09/2015).

[7] Sarmento, S. M. and Horta, N. (2020). Enhancing a pairs trading strategy with the application of machine learning. *Expert Systems with Applications*, 158:113490.

[8] Vittori, E., de Luca, M. B., Trovò, F., and Restelli, M. (2020). Dealing with transaction costs in portfolio optimization: online gradient descent with momentum. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8.

[9] Wang, C., Sandås, P., and Beling, P. (2021). Improving pairs trading strategies via reinforcement learning. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–7. IEEE.