Dimensionality Reduction and Clustering for Pairs Trading Selection

Jerry Loh[1], Lian Kah Seng[2]

National University of Singapore[1,2]
jloh@u.nus.edu[1], kahseng@u.nus.edu[2]

Abstract

In this project, we attempt to search for pairs of securities suitable for a mean-reversion pairs trading strategy using machine learning methods. In particular, we apply unsupervised dimensionality reduction and clustering techniques to historical daily price data for a set of 4088 ETFs. These methods include Principal Component Analysis (PCA), k-Means Clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Hierarchical Clustering.

Introduction

In financial markets, pairs trading is a trading strategy which involves two or more securities that behave similarly. Given two companies, A and B, with similar underlying characteristics (for example Coca-Cola and Pepsi), we might expect the stock prices of A and B to behave similarly in the financial market. As a result, a linear combination of the two stocks (ex. long A and short B) in a portfolio might produce a stationary, mean-reverting set of data, which can then be exploited by traders for profit. When stock A is relatively underpriced to stock B, traders might take a long position in A and a short position in B, expecting the difference in price to eventually converge to a normalised level.

However, pairs that exhibit such properties are rare in practice. A computational search for such pairs can speed up research greatly, given the large number of securities in the financial market. A naive, pairwise brute-force approach incurs a time complexity of $O\left(\frac{n(n-1)}{2}\right)$, where n is the number of securities tested. Such an approach is not only time inefficient, but the large number of statistical tests may also result in the multiple comparisons problem, leading to many Type I errors in our search.

Our approach is to apply dimensionality reduction (PCA) and clustering (KMeans, DBSCAN, Hierarchical Clustering) techniques on time series price data to search for groups of similar securities. This reduces the number of statistical tests required, and improves computational efficiency.

Related and Referenced Works

Sarmento and Horta's *A Machine Learning based Pairs Trading Investment Strategy* explore unsupervised machine learning methods for pairs trading selection. In particular, they apply OPTICS for clustering and selection. We attempt to explore other methods such as k-Means, DBSCAN and Hierarchical Clustering for the same application. Vidyamurthy's *Pairs Trading: Quantitative Methods and Analysis* was immensely useful for the data processing step. In particular, Vidyamurthy's explanations on factor models for logarithmic stock returns provided us a foundational understanding of pairs trading. Avellaneda and Lee's *Statistical Arbitrage in the U.S Equities Market* provided the motivation for the implementation of PCA for dimensionality reduction.

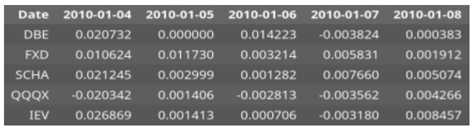
Datasets

The dataset used is from financial data vendor *kibot.com*, and contains .csv files of historical adjusted daily price data for 4088 ETFs. The chosen training and validation periods are from 2015/01/01 to 2020/01/01, and 2020/01/02 to 2022/01/02. The data cleaning process is detailed in the accompanying Jupyter Notebook.

Data Preparation

To prepare our data for use, we transformed the 4088 .csv files into pandas DataFrames indexed by datetime and with one column of daily logarithmic returns, where the log return at time t, r_t with price p_t is $r_t = ln\ \frac{p_t}{p_{t-1}}$.

These were then concatenated along columns into a single DataFrame. By taking the transpose of this DataFrame, each row of the DataFrame represented the log returns of a single ETF, while each day in the training period represented a feature column.



Date	2010-01-04	2010-01-05	2010-01-06	2010-01-07	2010-01-08
DBE	0.020732	0.000000	0.014223	-0.003824	0.000383
FXD	0.010624	0.011730	0.003214	0.005831	0.001912
SCHA	0.021245	0.002999	0.001282	0.007660	0.005074
QQQX	-0.020342	0.001406	-0.002813	-0.003562	0.004266
IEV	0.026869	0.001413	0.000706	-0.003180	0.008457

Fig. 1.1 Transformed DataFrame format

Feature Scaling

We applied standardisation as our feature scaling method to the dataset. The standardisation algorithm is as follows: $x' = \frac{x-\mu}{\sigma}$. We chose to fit the data to a standard Gaussian distribution because log returns are an unbounded, quantitative, continuous variable. Furthermore, the assumption of log-normal prices and hence Gaussian log returns is common in the field of quantitative finance.

Methodology and Implementation

As the time-series data is unlabelled, unsupervised learning methods must be used to identify the underlying patterns and structures within each ETF price series.

Our attempt at solving this problem was twofold: First, we applied dimensionality reduction techniques (PCA) to reduce the number of features in our DataFrame while maintaining as much signal as possible. Second, we applied a variety of clustering algorithms on the principal components to group similar ETFs with similar price behaviour together. In order to minimise the human decision factor, we applied optimisation methods to automate the selection of hyperparameters for clustering.

Dimensionality Reduction

Due to the sheer size of the dataset, one of our main concerns was time and space efficiency. In addition, the high number of features in our dataset would likely cause our model to overfit. Thus, we applied dimensionality reduction prior to any clustering methods on our data.

Principal Component Analysis (PCA)

PCA is an unsupervised learning technique that reduces the dimensionality of large data sets by simplifying data, reducing noise and transforming them into smaller *principal components*, while best retaining information in the original dataset.

PCA creates new *principal components (PCs)*, which are uncorrelated, linear combinations of features in the original dataset that maximise variance. Firstly, for a $N \times M$ dataset with M features, a $M \times M$ covariance matrix is formed, where the $(i,j) - entry$ of the covariance matrix is the covariance between the i^{th} and j^{th} features. Next, the covariance matrix is diagonalised to obtain its eigenvalues in descending order. The k^{th} principal component can be obtained from the eigenvector associated with the k^{th} largest eigenvalue. Choosing the n largest eigenvectors, a $M \times n$ matrix can be formed, which is post-multiplied to the original dataset to obtain a $N \times n$ PCA matrix with n PCs .

The choice of the number of PCs can be optimised by finding the elbow point of the graph of percentage explained variance against the number of PCs chosen. From the figure below, we chose the number of PCs to be 11 from the elbow point.

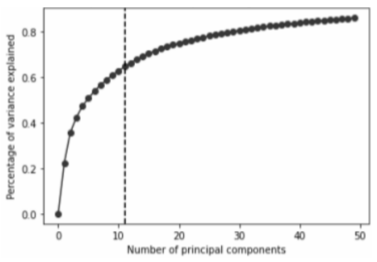


Fig. 1.2 Percent variance explained against no. of PCs

The scatterplot below of PC2 against PC1 showcases the application of PCA on our training set and the resulting transformed data points in $\mathbb{R}^2$.

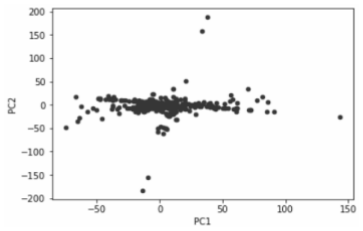


Fig. 1.3 Scatterplot of PC2 against PC1

k-Means Clustering

k-Means Clustering is an unsupervised learning technique that, for a chosen value of *k*, identifies k clusters of objects based on the objects' proximity to the centre of the *k* groups. The centre is determined as the arithmetic average of each cluster's n-dimensional vector of feature variables.

The execution of k-Means can be described as follows: The algorithm is initialised with a chosen *k*-value, and *k* points are initialised as our centroids. In the standard k-Means algorithm, the choice of initial centroids is randomised. However, poor initialisation of centroids can lead to a suboptimal solution. Hence, our initialisation was done via the k-means++ algorithm, which picks each centroid sequentially by maximising the squared euclidean distance from already existing centroids. The next two steps are then repeated until convergence: Firstly, all data points are assigned to their closest cluster centre. For a dataset with p features, the distance between the *i*-th data point, represented by a feature vector $(x1_i, x2_i \ldots xp_i)$, and a centroid D_c, $c \in 1,2,\ldots,k,$ in the vector space is calculated with a distance metric (we use the euclidean distance). This is given by $\sqrt{\sum_{j=1}^{p}\left(xj_i - xj_{D_c}\right)^2}$. Next, once all the data points have been assigned, in each cluster, a new centroid coordinate is generated by taking the arithmetic mean of all cluster points. For a new cluster with z data points, the new centroid is given by the vector

$\left(\frac{1}{z}\sum_{i=1}^{z} x1_i, \frac{1}{z}\sum_{i=1}^{z} x2_i, \dots, \frac{1}{z}\sum_{i=1}^{z} xp_i\right)$. The steps are then repeated till convergence, which is when cluster assignments stop changing.

We optimised the value of k using the Within Sum of Squares (WSS) method. For each centroid, the sum of squared distances from each of the points within the cluster is calculated. These values are then summed up together and plotted onto a graph. For a data set with M points and p feature variables, the WSS is calculated via $\sum_{i=1}^{M}\sum_{j=1}^{p}\left(xj_i - xj_{D_c}\right)^2$. In the case of our data set, the optimal k-value was 45. As seen in Fig 1.4, the "elbow" of the curve was at k = 45. Fig. 1.5 showcases the application of k-Means on our training set.

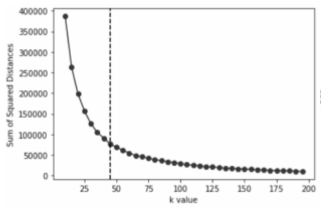


Fig. 1.4 Graph of WSS against k-value

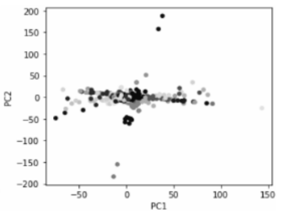


Fig 1.5 k-Means Clustering results for k = 45

One of the limitations of k-Means is that it is unable to handle outliers and noisy data. To combat this, the next clustering technique, DBSCAN, covers this exact flaw.

DBSCAN

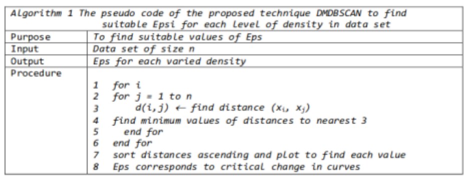
DBSCAN is an unsupervised learning technique that clusters based on a density metric. In the case of DBSCAN, not all data points will belong to a cluster at the end of the process. Data points belonging to no cluster will be labelled as outliers and noise. This is good for our problem, since not every ETF should be paired up.

In order to discuss how DBSCAN works, certain key terms and underlying fundamental concepts have to be defined. First, the ε-neighbourhood of a point, q, is the set of all points at most ε distance away from p in the vector space. This is defined as $N_\varepsilon(q) = \{p \in X | d(q,p) \leq \varepsilon\}$, where $d(q,p)$ is the euclidean distance between points q and p, and X is the set of all points. Second, a point q is considered a core point if $card\left(N_\varepsilon(q)\right) \geq minPts$, where $card\left(N_\varepsilon(q)\right)$ represents the number of points within the ε-neighbourhood of q, and $minPts$ is a hyperparameter to be chosen. Third, a point p is *directly density-reachable* from a point q if and only if $p \in N_\varepsilon(q)$ and q is a core point. Fourth, a point p is *density-reachable* from a point q if and only if there is a chain of objects $p_1, \dots, p_n$, where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i. Finally, a point p is *density-connected* to a point q if both points are density-reachable from a common core point.

All points within a DBSCAN cluster must be mutually *density-connected*. This means that the non-core points form the edges of the cluster, since they cannot be used to reach more points. Since points can only be considered core points if the ε-neighbourhood is large enough, this algorithm takes into account not just distance but also cluster density.

The simplified execution of DBSCAN is as follows. First, we pick an arbitrary point in the dataset and apply the following algorithm until all points have been visited: For the ε-neighbourhood of a point, check if it is a core point, and if the neighbourhood has $\geq minPts$ points. If so, all points in this neighbourhood are part of the same cluster. Otherwise, the point is assigned as noise. This cluster is then recursively expanded by repeating this step for each neighbouring core point.

The ε-value is a hyperparameter to be defined. To optimise the selection process of the ε-value, an algorithm (Fig. 1.6) derived from an existing paper was used on our training set.



Algorithm 1 The pseudo code of the proposed technique DMDBSCAN to find suitable Epsi for each level of density in data set	
Purpose	To find suitable values of Eps
Input	Data set of size n
Output	Eps for each varied density
Procedure	see below

```
1  for i
2  for j = 1 to n
3       d(i,j) ← find distance (x_i, x_j)
4  find minimum values of distances to nearest 3
5    end for
6  end for
7  sort distances ascending and plot to find each value
8  Eps corresponds to critical change in curves
```

Fig. 1.6 Pseudocode of Algorithm (Elbata 2012)

This algorithm plots out the sorted euclidean distance between each data point and its nearest neighbour. Then, using the elbow of the graph (Fig. 1.7), the optimal ε value 12.5 is chosen. The resulting DBSCAN clustering can be seen in Fig. 1.8.

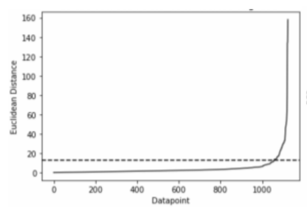


Fig. 1.7 Graph of Euclidean Dist. Against Data Points

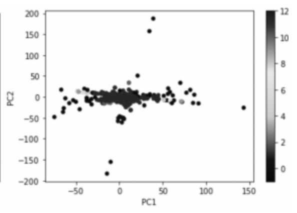


Fig. 1.8 DBSCAN results

Hierarchical Clustering

Hierarchical Clustering is an unsupervised learning technique that builds a hierarchy of clusters. In our chosen approach, which was Agglomerative Hierarchical Clustering, each data point starts off as a single cluster, and pairs of clusters nearest to each other are merged together iteratively until all the data is in a single cluster.

This generates a dendrogram showing how the clusters merge from bottom-up. The dendrogram also displays the euclidean distances between each cluster. The advantage of Agglomerative Clustering is that the number of clusters does not need to be specified. However, this comes at the cost of not being able to be optimised, as well as worse time complexity.

The implementation is as follows: For a set of N data points, each data point is initialised as its own cluster. Applying a greedy approach, the two closest data points are merged to form a single cluster, resulting in $N-1$ clusters. This is repeated until only one cluster containing all N points is left. This process is visualised in the dendrogram. For this project, we chose $\sqrt{N} = 33$ clusters as our cut off.

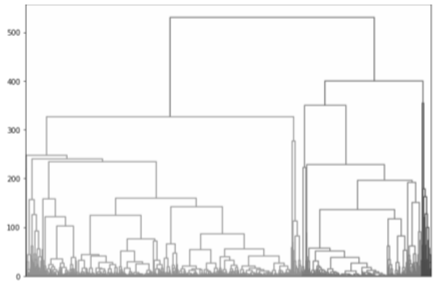
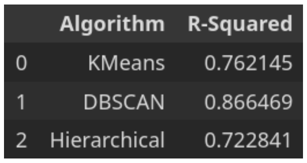


Fig. 1.9 Dendrogram of cluster results

Evaluation

In this section, we intend to highlight some results from the application of our proposed techniques. This will be done mainly through the evaluation of the composition of the clusters found. Since the usefulness of a cluster of ETFs is dependent on the suitability of its members in a pairs trading strategy, we can gauge the performance of a clustering algorithm by calculating the average pairwise R-Squared correlation for ETFs in each cluster.



	Algorithm	R-Squared
0	KMeans	0.762145
1	DBSCAN	0.866469
2	Hierarchical	0.722841

Fig. 1.10 Results of evaluation

From the test, DBSCAN seemed to fare better than the other two algorithms. Plotting some of the cluster groups from the DBSCAN results in the validation period of 2020/01/02 to 2022/01/02, we observe that each cluster of ETFs exhibit very similar price movements and behaviours. These would make good selections of pairs for a pairs trading strategy, which was our initial objective.

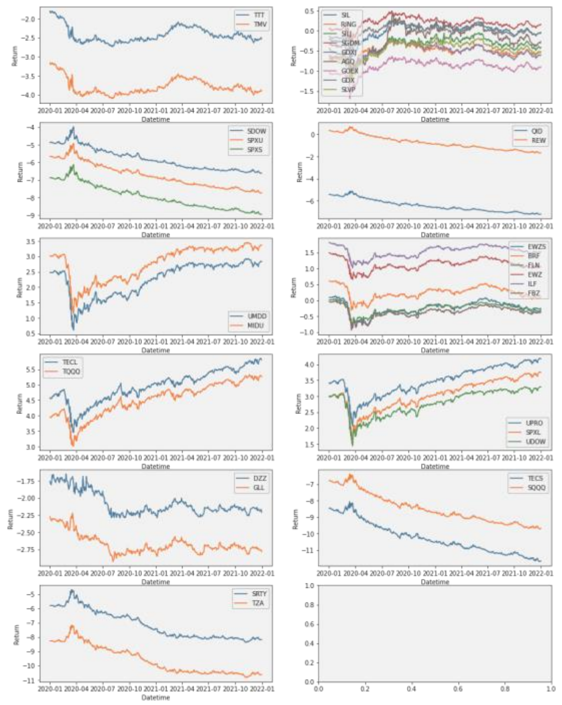


Fig. 1.11 DBSCAN clustering results

The selection of hyperparameters was done algorithmically, with the use of elbow methods for k-Means and DBSCAN, and the dendrogram for Hierarchical Clustering. This allowed us to optimise the algorithms for the best performance while avoiding serious overfitting. In addition, we also observed that increasing the number of principal components to keep at the dimensionality reduction stage improved the pairwise correlation metric mentioned above, leading to better performance. However, a choice of too many principal components leads to overfitting clusters of ETFs that are too similar and have too little variability, which is unsuitable for practical trading purposes.

Conclusion

The use of unsupervised dimensionality reduction and clustering techniques was successful in finding pairs of ETFs suitable for a pairs trading strategy. In particular, PCA + DBSCAN appeared to be the most effective solution out of the methods tried in this study.

However, one possible area of improvement is the evaluation criteria used for comparing the three clustering algorithms. Instead of using the R-Squared correlation metric, tests for cointegration could be more suited to check for stationary series. Also, other methods to optimise hyperparameters for clustering algorithms could be explored in the future.

Citations

Avellaneda, M., & Lee, J.-H. (2008). Statistical arbitrage in the U.S. Equities Market. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.1153505

Nadia, R.; and Imas S. S. 2016. *Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra.* IOP Publishing Ltd.

Sarmento Moraes Simão, & Horta, N. (2020). *A Machine Learning Based Pairs Trading Investment Strategy*. Springer.

Vidyamurthy, G. (2011). *Pairs trading: Quantitative methods and analysis*. John Wiley & Sons, Inc.