COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

IEOR Deep Learning

Final Project

Fund Clustering

Raina Cho
Yuanjun Ling
Thomas Luong

Supervised by
Prof. Ali Hirsa, Department of Industrial Engineering and Operations Research, Columbia University
&
Satyan Malhotra, Managing Partner, FlexStone Partners

August 2020

Table of Contents

Introduction

The purpose of this project is to use financial features to cluster mutual funds and label them into different categories. This is a continuation of the project in the Spring 2020 semester when the last team only took return data to do the clustering and did not come up with validation and labelling measures. Our job is to take into account more important and robust aspects of the research, like having statistical measures or other metrics to explain and validate the clusters, applying holding asset and performance indicator data when doing the clustering, and creating labels for the cluster results.

We apply here a two-layer clustering model using neural networks and a statistical labelling method. The idea is to, first, cluster funds with regards to their holding assets (equities, bonds and cash) with a combination of k-means clustering and hierarchical clustering, then secondly, do subclustering on each initial cluster using return and performance data with a convolutional autoencoder.

For cluster identification, we create labels using the holding asset data to capture the investment nature of the clusters and the level of active management, also adding a measure of performance and risk based on the relative rankings among all clusters.

In addition to cluster labels, since this research is essentially an unsupervised learning problem, we have also come up with different statistical methods to gauge the performance of the clustering. The elbow method, gap statistic and t-SNE method were used to quantitatively and qualitatively choose the optimal number of clusters in the k-means algorithm, the Davies-Bouldin index and the silhouette coefficient, along with box plots and other statistical plots, helped judge the quality of the clustering output.

Clustering

Having tested multiple clustering algorithms, it appeared that outliers would appear no matter what features were chosen as inputs and that funds with similar exposure could have vastly different performances. Therefore, the idea is to apply a first clustering layer to group funds using the holding asset data only and iteratively regroup mutual funds with a k-means clustering module. Then, we feed the clusters and their performance features to a second layer to cluster funds with similar features together.

1. First layer

Since the holding assets are most often stable through time, we are first clustering based on the holding asset data from 2010 to 2019 as our sole input for the clustering algorithm. The holding asset data is a fund identifier (around 11k funds rows) by holding asset type (cash, equity, bond, security - four columns in total) dataframe. Cell (*i*,*j*) represents the percentage of fund *i*'s total invested capital in asset *j*. Negative percentage of asset holding exists and suggests a short or borrowing in certain asset categories while a sum of 100 for all holding asset percentages will always be guaranteed for every fund. We temporarily set the clustering window as one year, e.g. we do clustering in 2019, 2018, …, separately and compare the cluster results to indicate the stability as a measure of validation.
Note: clustering window is one of the future works to look at.

The main issues we found about holding asset data-based clustering are as follows.

- The holding asset data is hard to cluster due to its continuity. Think of finding a plane to well classify one-dimensional data like 1,2,3,4,5,6,7, which is very hard to use quantitative methods solely.
- The continuity of the data makes it hard not only to separate the funds but also to define the best number of clusters *k*.
- There are many different outliers. For example, we could find many leveraged funds that short greatly in cash or security to have more than 100 percent holdings in equity or bonds.

To solve the issues above, we propose to:

- Add subjective qualitative standards to cut the mutual funds with continuous holding asset percentages. More precisely, we create a boxplot for each cluster group and check (1) The range of 75th percentile and 25th percentile exceeds 20% of the total or not, (2) The range between 'minimum' and 'maximum' exceeds 50% of the total or not, (3) The number of points either lower than 'minimum' or higher than 'maximum' exceeds 10 or not. These

standards are focused on reducing the in-cluster variances and improving the consistency in terms of the holding asset. If any of the above conditions are satisfied, we consider there are outliers in the current group and accordingly split the cluster group into two.

- Use hierarchical clustering to get the initial cluster centers to avoid the randomness and uncertainty of the k-means algorithm.

- Create an iterative process that iteratively extracting outliers from the cluster groups, setting the outliers as new cluster centers and feeding them into k-means clustering again until no more outliers are stratified from the latest clustering results, in which way we do not have to worry about selecting the best number of clusters.

We can conclude the first layer process as:

1. Use the hierarchical clustering with a k (number of clusters) determined statistically through silhouette coefficient to get the initial cluster centroids.

2. Apply the k-means clustering model with the appointed initial cluster centroids and Euclidean distance as the dissimilarity matrix to get the cluster results.

3. Extract outliers from the results from step 2 based on some qualitative standards.

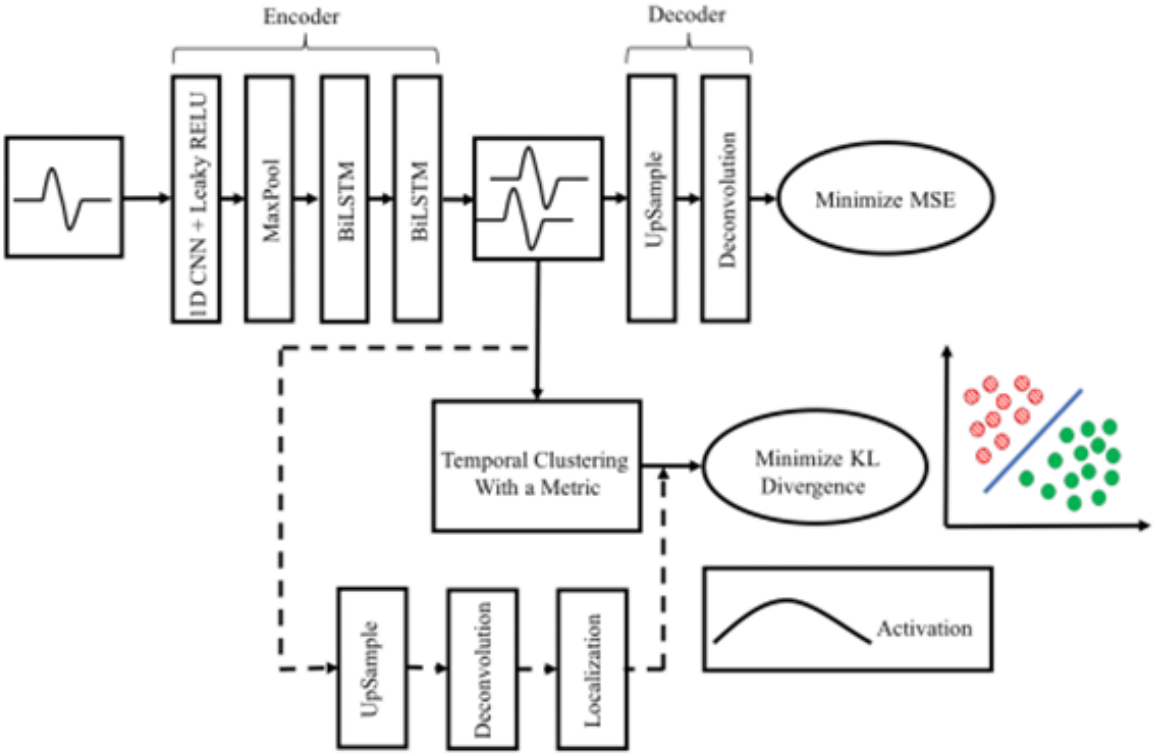
4. If any outlier is identified, add it to the existing initial cluster centroids list and go back to step 2. Else, we move forward to step 5.

5. Merge cluster groups that have very 'close' cluster centroids and investing styles because the iterative approach from step 2 to step 4 would exhaustively output 1000+ clusters. We define close with a Euclidean distance based threshold and similar investing styles as the same descendant holding asset order.

2. Second layer

After finishing the clustering based on holding assets, we do subclustering for each of the clusters derived from the first layer based on the daily return data respectively.

We reference the model from the paper '*Deep Temporal Clustering: Fully Unsupervised Learning Of Time-domain Features*' by Naveen Sai, M. et al, with slight changes. In short, the model is divided into two parts, the first part is an autoencoder trying to represent the return data in a latent space while the second part is the clustering model that assigns groups for compressed data derived from the autoencoder. The model architecture is as follow:



Model structure (sourced from the paper 'Deep Temporal Clustering: Fully Unsupervised Learning Of Time-domain Features')

First part - autoencoder:

The input data is encoded into a latent space by a temporal autoencoder (TAE) consisting of convolution and Bi-LSTM combined neural networks. The first level of the network architecture consists of a 1D convolution layer, which extracts key short-term features (waveforms), followed by the leaky rectifying linear units and max pooling layer. The first level thus casts the time series into a more compact representation while retaining most of the relevant information. This dimensionality reduction is crucial for further processing to avoid very long sequences. First-level activations are then fed to the second level (Bidirectional LSTM) to obtain the latent representation. We use Bi-LSTM to learn temporal changes in both time directions. This allows to collapse the input sequences in all dimensions except temporal, and to cast the input into a much smaller latent space. Learning in 1D CNN and Bi-LSTM is driven by the autoencoder loss, provided by the mean square error (MSE) of the input sequence reconstructed from the latent representation after Bi-LSTM.

Second part - temporal clustering:

In this part, the model takes the compressed data as input and generates the initial cluster centroids through hierarchical clustering with complete linkage to give a good starting point and mitigate the randomness in k-means initial centroid generating process. Note: this is a point to look at in future work because different starting points are helpful for deep network training. One important parameter when doing clustering is to define the dissimilarity matrix. There are mainly four options: Complexity Invariant Similarity (CID), Correlation (COR), Auto Correlation based Similarity (ACF) and Euclidean (EUCL). After running each of them, we found that the traditional Euclidean distance performs best and picked it as the dissimilarity matrix in our clustering algorithm.

After setting the initial centroids and defining the dissimilarity matrix, we move on to train the temporal clustering layer using an unsupervised algorithm that alternates between two steps for x epochs.

1. First, we compute the probability of assignment of input x_i to the cluster j. The closer the latent representation z_i of input x_i is to the centroid w_j , the higher the probability of x_i belonging to cluster j (see equation 1).

$$q_{ij} = \frac{\left(1 + \frac{siml(z_i, w_j)}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_{j=1}^{k}\left(1 + \frac{siml(z_i, w_j)}{\alpha}\right)^{-\frac{\alpha+1}{2}}}$$

Equation 1, where $\alpha = 1$

2. Second, we update the centroids by using a loss function, which maximizes the high confidence assignments using a target distribution p, (see equation 2). Finally, the clustering layer assigns the latent representation of sequences x_i, $i = 1, \ldots, n$, to clusters.

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j=1}^{k} q_{ij}^2 / f_j}$$

Equation 2, where $f_j = \sum_{i=1}^{n} q_{ij}$

By using the target distribution p_{ij}, we could compute the KL divergence loss through the equation below, where n and k are the number of samples and number of clusters respectively.

$$L = \sum_{i=1}^{n} \sum_{j=1}^{k} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

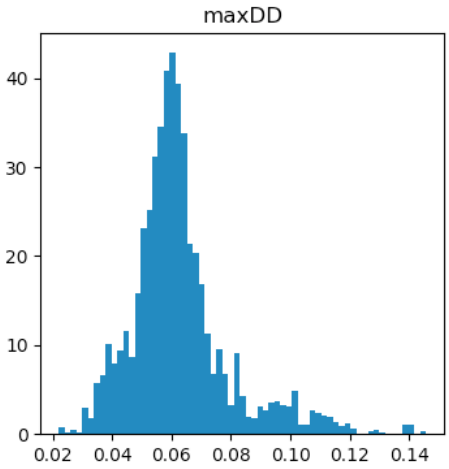
By training the loss to its minimum using the Adam optimizer, we can get the final subclustering result for the group we get from the first layer.

Labeling

Because the clustering problem is an unsupervised learning task, we need to be able to identify the underlying meaning of the clusters and find a way to validate the results, both qualitatively and quantitatively. We have designed multiple tests and metrics to help us in this matter, as well as a method to label the clusters according to their performance and risk level.

1. Clustering validation

One of the biggest issues with unsupervised clustering is the validity of the result and its interpretation. Since we do not have target labels, we can not use accuracy or other similar metrics to judge the performance of the models. Intuitively, we could say that the further apart clusters are and the closer the components are to the centroids, the better. This is exactly what the Davies-Bouldin index and the silhouette coefficient measure. We found out that the Calinski-Harabasz index, based on the inter-cluster and intra-cluster variances, is faster to compute. These metrics provide a quantitative way to compare clustering results, but in practice, they may give very close scores to different models. In order to further discriminate the results, we also looked at the repartition of components across pairs of features such as total return vs max drawdown. Ideally, funds with similar performances should be grouped together, thus be adjacent to each other in these plots. We are also able to observe histograms or box plots for each cluster and each feature of interest to detect outliers or inconsistencies in the results. Having graphical methods adds an extra layer of inspection in the clustering output.



Example of max drawdown histogram for a particular cluster

Using all of these methods, we are able to effectively compare clustering models and explain why one model can perform better than the others with confidence.

2. Cluster labeling methods

Once we have reached the point where we are satisfied with the clustering results, the last step is to assess how the clusters differ from each other and label them. Based on the methods we used to perform our first and second layer clusterings, we expect the clusters to be separated with respect to both holding asset data and performance metrics. Below is a snapshot of the clustering labels we created for one of the clusters. Each column will be further explained below.

		Cluster Description	Single Asset Focus	Multi Asset Focus
Cluster	**Subcluster**			
4	12	80%> investment in Common Stock and Cash		Common Stock, Cash

		Shorted Asset	volatility	annual_return	max_dd	vol_median	return_median
Cluster	**Subcluster**						
4	12		High	High	High	0.007	0.19

		max_dd_median	Top Morningstar Category	Top Cluster Category	allocation_chg_std
Cluster	**Subcluster**				
4	12	0.105	Emerging Markets Funds	cash equity mixed group	3.264303

Active_management	Common Stock	Preferred Stock	Convertible Bonds	Corporate Bonds
Low	48.19	0.56	0.0	0.0

Muni Bonds	Gov Bonds	Other Securities	Cash	ABS	MBS	Other Equity	Other FI
0.0	0.0	0.0	41.08	0.0	0.0	10.17	0.0

At the individual mutual fund level, we have average percentages invested in different holding asset categories throughout each calendar year spanning 2010 to 2019. These holding asset categories include common stock, preferred stock, convertible bonds, corporate bonds, municipal bonds, government bonds, other securities, cash, ABS, MBS, other equity, and other fixed income. For each cluster, we calculated the median percentage invested in each asset type. These median investment percentage values are shown in the **'Common Stock', 'Preferred Stock', 'Convertible Bonds', 'Corporate Bonds', 'Muni Bonds', 'Gov Bonds', 'Other Securities', 'Cash', 'ABS', 'MBS', 'Other Equity',** and **'Other FI'** columns.

Cluster Description column

Our clusters fall into one of the following descriptions.

- **"Heavily invested…"** in one asset type: if a cluster has invested more than 75% in one type of asset.
- **"80%> investment in Asset A and Asset B"**: if the top two asset types make up more than 80% of the fund.
- **"Evenly diversified…"**: if no one asset type takes up the majority (no one asset exceeds 50% asset allocation), then these clusters are considered evenly diversified. The top assets that together add up to 80% of the fund are listed in the description column.
- **"Uneven multi-asset…"**: mix of multiple asset types but not evenly diversified across asset types. This category captures the residual clusters that do not belong to any of the above three categories. The top assets that together add up to 75% of the fund are listed in the description column. We first tried 80%, which resulted in certain clusters with too long of a description, so we dialed the cutoff down to 75%, which yielded cleaner cluster descriptions.
- **"Shorted…"**: if the average fund has a sizeable short position on a certain asset, we added this description at the end, specifying the shorted asset.

Single Asset Focus column: if the cluster falls under the "heavily invested in one asset" type from above, the said asset is shown under the Single Asset Focus column.

Multi Asset Focus column: this column lists the multiple asset concentrations of the remaining cluster categories, which includes "80%> investment in…", "Evenly diversified…", and "Uneven multi-asset…" clusters. As mentioned above, these clusters have more than one type of asset focus, so we listed those assets in a separate column for easier understanding.

Risk and Return Profile of Each Cluster

To further describe the risk and return profile of the clusters, we first calculated each cluster's median annual volatility, return, and max drawdown values. These values are shown in the **vol_median, return_median, max_dd_median** columns. We took the different distributions of these metrics and for each metric, marked the top 33 percentile as 'High', the bottom 33 percentile as 'Low', and the remaining values in the middle as 'Mid.' These are shown in the **volatility, annual_return,** and **max_dd** columns. We listed both the median numerical values as well as the High/Mid/Low classifications.

Level of Active Management

We also looked to gauge how actively these funds were managed over time. To do so, we looked at how much the asset allocation percentages have changed over the years (2010 - 2019) by calculating the standard deviation for each fund. This metric is shown in the **allocation_chg_std** column. Applying the same logic we used for the risk and return profile of the clusters, we categorized them as High/Mid/Low in the **Active_management** column.

Top Morningstar and Cluster Category

The Morningstar database already included a couple categories for each mutual fund. These are labelled Morningstar Category and Cluster Category in the original dataset. Since we created clusters of our own, as a sanity check, we are showing the most frequent categorizations for each cluster. These are shown in the **Top Morningstar Category** and **Top Cluster Category** columns. For example, if we categorized a certain cluster as predominantly common equity and preferred equity focused cluster, but the Morningstar categories mention fixed income focus, then we would have reasons to look further into why that discrepancy exists.

Conclusion

The whole project is focused on clustering mutual funds into different cluster groups based on data like the holding assets, daily returns, historical performance, etc. We introduced a two layered clustering model where in the first layer we clustered the mutual funds according to the holding asset data which will give us a pretty stable cluster label across different time periods and in the second layer we did subclustering on the mutual funds for each of cluster results in the first layer according to their daily return data by extracting features through autoencoder techniques. After clustering, we proceeded to work on validating the cluster results, by either statistical methods that evaluate cohesion and dispersion of clusters in distance, or visualization methods that display the boxplots, distribution, etc. to tell the performance of the cluster results. Finally, we added labels for each of the cluster groups we got for better knowledge of the traits in each of the cluster groups.

There are also some important indications from this project. The first one is that using a purely quantitative method when doing the clustering would finally give a bad result and adding qualitative standards is essential. Secondly, if clustering on the performance features, the cluster results would be very unstable because the return performance indicators are inherently unstable, and the past is not a suggestion for the future. Lastly, adding as many more features would not help in this project because it would finally output hundreds of clusters, a number too large that loses its practical sense.

Future work

1. Further look into the clustering window. We currently define the clustering window as one year empirically. Improvement could be developing a more robust method to define the clustering window that well catches the sensitive change in holding asset percentages of each fund while ensuring the clusters' stability over time.
2. In the second layer clustering, we first train the autoencoder to get the best representation of original return data in a latent space and then train the clustering to assign cluster groups to the compressed data, while the original paper we reference on train the autoencoder and clustering at the same time and achieve a pretty good result. We tried to follow the simultaneous training method as the paper but realized that the result is no better than the separate training, which we finally adopted in our model. If having more time, trying to reconstruct the model through simultaneous training could be a way to improve the model power.

3. Finally, the parameter tuning is another aspect worth looking at, especially when defining the initial cluster centroids and the dissimilarity matrix used in the clustering algorithm, which we did a little bit but more could be done, besides on the ordinary hyper parameter tuning on epochs, filters, layers, etc.