Deep Prediction Of Investor Interest: a Supervised Clustering Approach

Baptiste Barreau, Laurent Carlier, Damien Challet

DEEP PREDICTION OF INVESTOR INTEREST: A SUPERVISED CLUSTERING APPROACH

Baptiste Barreau[1, 2], Laurent Carlier[2], and Damien Challet[1]

[1]Chair of Quantitative Finance, MICS Laboratory, CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvette, France
[2]BNP Paribas Corporate & Institutional Banking, Global Markets Data & Artificial Intelligence Lab, Paris, France
{baptiste.barreau, laurent.carlier}@bnpparibas.com
damien.challet@centralesupelec.fr}

ABSTRACT

We propose a novel deep learning architecture suitable for the prediction of investor interest for a given asset in a given timeframe. This architecture performs both investor clustering and modelling at the same time. We first verify its superior performance on a simulated scenario inspired by real data and then apply it to a large proprietary database from BNP Paribas Corporate and Institutional Banking.

***K*eywords** investor activity prediction, deep learning, neural networks, mixture of experts, clustering

1 Introduction

Predicting investor activity is a challenging problem in Finance. The basic problem can be stated as follows: given many thousands of assets and many thousands of investors, predict which investors will be interested in buying/selling which assets in the next (short) time period. What makes this problem difficult is the large heterogeneity of both investors and assets, compounded by the non-stationary nature of markets and investors and the limited time over which predictions are relevant.

Ad-hoc methods are surprisingly efficient at clustering investors according to their trades in a single asset [Tumminello et al., 2012]. In addition, clusters of investors determined for several assets separately have a substantial overlap [Baltakys et al., 2018], which shows that one may be able to cluster investors for more than a few assets at a time. The activity of a given cluster may systematically depend on the previous activity of some clusters, which can then be used to predict the investment flow of investors [Challet et al., 2018]. Here, we leverage deep learning to train a single neural network on all the investors and all the assets and give temporal predictions for each investor.

The heterogeneity of investors translates into a heterogeneity of investment strategies [Tumminello et al., 2012, Musciotto et al., 2018]: for the same set of information, e.g., financial and past activity indicators, investors can take totally different actions. Take for instance the case of an asset whose price has just decreased: some investors will buy it because they have positive long-term price increase expectations and thus are happy to be able to buy this asset at a discount; reversely, some other investors will interpret the recent price decrease as indicative of the future trend or risk and refrain from buying it.

Formally, in our setting, a strategy f is a mapping from current information x to expression of interest to buy and/or sell a given asset, encoded by a categorical variable y: $f : x \mapsto y$. We call here $\mathcal{D} = \{f_k : x \mapsto y\}_k$ the set of all the investment strategies that an investor may follow. Unsupervised clustering methods suggest that the number different strategies that describe investors' decisions is finite [Musciotto et al., 2018]. We therefore expect our dataset to have a finite number K of clusters of investors, each following a given investment strategy f_k. Consequently, we expect $\mathcal{D}$ to be such that $|\mathcal{D}| = K$, i.e. $\mathcal{D} = \{f_k : x \mapsto y\}_{k=1,\cdots,K}$. Alternatively, $\mathcal{D}$ can be thought of as the set of distinguishable strategies, which may be smaller than the total number of strategies and which may therefore be considered as an effective set of strategies. At any rate, a suitable algorithm to solve our problem therefore needs to be able to infer the set of investment strategies $\mathcal{D}$.

A simple experiment shows how investors differ. We first transform BNP Paribas CIB bonds' *Request for Quotation* (RFQ) database, along with market data and categorical information related to the investors and bonds, into a dataset of custom-made features describing the investors' interactions with all the bonds under consideration. This dataset is built so that each row can be mapped to a triplet *(Investor, Financial Product, Date)*. This structure allows us, for a given investor and at a given date, to provide probabilities of interest in buying and/or selling a given financial product in a given timeframe. When an investor shows an interest in a given financial product, i.e. a *positive event*, we extend the duration of this event to 5 business days, for two reasons: first because bonds are by essence illiquid financial products and second because this increases the proportion of positive events.

At each date, negative events are randomly sampled in the *(Investor, Financial Product)* pairs that were observed as positive events in the past and that are not positive at this date. Using this dataset, we conduct an experiment to illustrate the *non-universality* of investors, i.e. the fact that investors have distinct investment strategies. The methodology of this experiment is reminiscent of the one used in Sirignano and Cont [2018] to study the universality of equity limit order books.

Figure 1: Universality matrix of investors' strategies: the y-axis shows investors' sector used to train a gradient boosting model while the x-axis shows investors' sector on which predictions are made using the model indicated on y-axis. Scores are average precision, macro-averaged over classes and expressed in percentages.

We use a dataset constructed as described above with five months of bonds' RFQ data. We split this dataset into many subsets according to the investors' business sector, e.g. one of these subset contains investors coming from the Insurance sector only. We consider here only the sectors with a sufficient amount of data samples to train and test a model. The remaining sectors are grouped together under the *Others* flag. Note that this flag is mainly composed of Corporate sectors, such as car industry, media, technology, telecommunications…For each sector, some of the latest data is held out, and a gradient boosting model is trained on the remaining data. This model is then used for prediction on the held-out data of the model's underlying sector, and for all the other sectors as well. For comparison purposes, an aggregated model using all sectors at once is also trained and tested in the same way.

Because classes are unbalanced, we compute the average precision score of the obtained results, as advised by Davis and Goadrich [2006], macro-averaged over all the classes, which yields the *universality matrix* shown in Fig. 1. The y-axis labels the sector used for training, and the x-axis is the section on which the predictions are made.

We observe that some sectors are inherently difficult to predict, even when calibrated on their data only — this is the case for Asset Managers of Private Banks and Pension Funds. On the contrary, some sectors seem to be relatively easy to predict, e.g. Broker Dealers and, to some extent, Central Banks. Overall, we note that there is always some degree of variability of the scores obtained by a given model — no universal model gives good predictions for all the sectors of activity. Thus follows the *non-universality of clients*. In addition, it is worth noting that the aggregated model obtained better performance on some sectors than the models trained on these sectors' data only. As a consequence, a suitable grouping of sectors would improve predictions for some sectors. This observation is in agreement with the above K-investment strategies hypothesis.

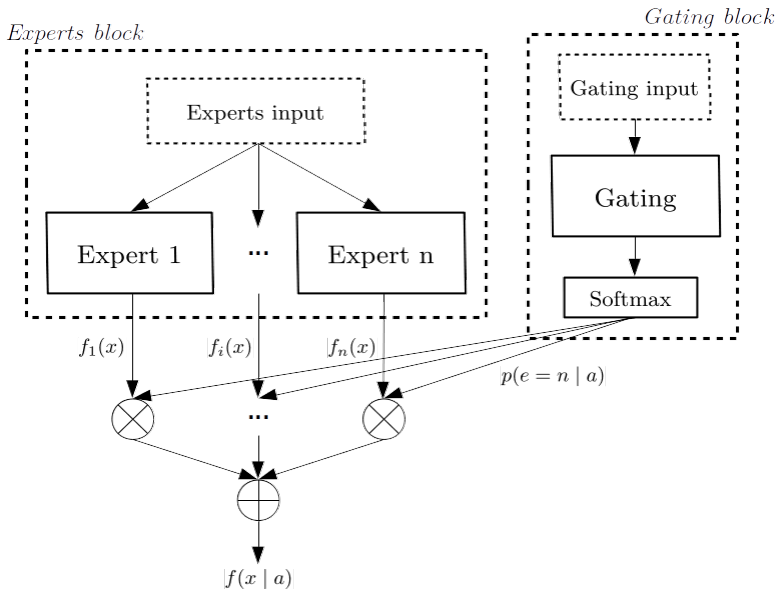

Figure 2: Global architecture of an ExNet

Following on these hypotheses, this work leverages deep learning both to uncover the structure of similarity between investors, namely the K clusters, or strategies, and to make relevant predictions using each inferred clusters. The advantage of deep learning lies in the fact that it allows to solve both of these tasks at once, and thereby unveils the structure of investors that most closely corresponds to their trading behaviour in a self-consistent way.

2 Related work

This work finds its roots in mixture-of-experts research, which began with Jacobs et al. [1991], from which we keep the basic elements which drive the structure presented in Section 3, and more particularly the gating and expert blocks. A rather exhaustive history of the research performed on this subject can be found in Yuksel et al. [2012].

The main inspiration for our work is Shazeer et al. [2017], which, although falling within the *conditional computation* framework, presented the first adaptation of mixture of experts for deep learning models. We built on this work to come up with a novel structure designed to solve the particular problem presented in Section 1. As far as we know, the approach we propose is new. We use an additional loss term to improve learning of the strategies, reminiscent of the one introduced in Liu and Yao [1999].

3 Experts Network

We introduce here a new algorithm, inspired by Shazeer et al. [2017], which we call the *Experts Network* (ExNet). The ExNet is purposely designed to be able to capture the hypotheses formulated in Section 1, i.e. to capture a finite, unknown number K of distinct investment strategies $\mathcal{D} = \{f_k : x \mapsto y\}_{k=1,\cdots,K}$.

3.1 Architecture of the network

The structure of an ExNet, illustrated in Fig. 2, comprises two main parts: a **gating block** and an **experts block**. Their purposes are the following:

- The **gating block** is an independent neural network whose role is to learn how to dispatch investors to n *experts* defined below. This block receives a distinct, categorical input, the *gating input*, corresponding to an encoding of the investors and such that the i-th row of the gating input corresponds to the investor indexing the i-th row of the experts input. Its output consists in a vector of size n which contains the probabilities that the input should be allocated to the n experts, computed by a softmax activation.

- The **experts block** is made of n independent sub-networks, called *experts*. Each expert receives as input the same data, the *experts input*, corresponding to the features used to solve the classification or regression task at hand, e.g. in our case the features outlined in Section 1 —for a given row, the intensity of the investor's interest in the financial asset considered, the total number of RFQ done by the investor, the price and the volatility of the asset... As investors are dispatched to the experts through the gating block, each expert will learn a mapping $f : x \mapsto y$ that most closely corresponds to the actions of its attributed investors. The role of an expert is therefore to retrieve a given f_k, corresponding to one of the K underlying clusters of investors which we hypothesized.

The outputs of these two blocks are combined through $f(x|a) = \sum_{i=1}^{n} p(i|a) f_i(x)$, where a denotes the investor related to data sample x and $p(i|a)$ is the probability that investor a is assigned to expert i. Our goal is that K experts learn to specialize to K clusters. As K is unknown, retrieving all clusters requires that $n \geq K$, i.e. n should be 'large enough'. We will show below that the network ability to retrieve the K clusters is not impacted by high values of n; using large n values therefore ensures that the $n \geq K$ condition is respected and only impacts computational efficiency. The described architecture corresponds in fact to a *meta*-architecture. The architecture of the experts is still to be chosen, and indeed any kind of neural network could be used. For the sake of simplicity and computational ease, we use here rather small feed-forward neural networks for the experts, all with the same architecture, but one could easily use experts of different architectures to represent a more heterogeneous space of strategies.

Both blocks are trained simultaneously using gradient descent and backpropagation, with a loss corresponding to the task at hand, be it a regression or classification task, and computed using the final output of the network only, $f(x|a)$. One of the most important features of this network lies in the fact that the two blocks do not receive the same input data. We saw previously that the gating block receives as input an encoding of the investors. As this input is not time-dependent, the gating block of the network can be used *a posteriori* to analyse how investors are dispatched to experts with a single pass of all investors' encodings through this block alone, thereby unveiling the underlying structure of investors interacting in the considered market.

For a given investor a, the gating block performs the following computation:

$$p(x|a) = Softmax\left(KeepTopL\left(W_{gating}x + \epsilon * \sqrt{Softplus(W_{noise}x)}\right)\right),$$

where x is a trainable d-dimensional embedding of the investor a, W_{noise} and W_{gating} are trainable weights of the network, $\epsilon \sim \mathcal{N}(0,1)$, and we define $Softmax(x)_k = e^{x_k} / \sum_i e^{x_i}$ and $Softplus(x) = \log(1 + \exp x)$. The $KeepTopL$ function introduced in Shazeer et al. [2017] is designed to restrict the scope of the gating process by transforming its input so that we only give weight to the top L values of the input. This function can be written:

$$KeepTopL(x, L)_i = \begin{cases} x_i & \text{if } x_i \text{ in top } L \text{ values of } x; \\ -\infty & \text{else.} \end{cases}$$

$KeepTopL$ adds sparsity to the network, L becoming a new hyperparameter of the network, with $L < n$ and usually chosen so that $L \ll n$. As we want experts to specialize to the investment strategy of a particular cluster of investors, the $KeepTopL$ function drives the network away from ensembling strategies, i.e. a setting where all investors are dispatched to all experts with equal probability. Setting $L = 1$ is however problematic, as a given investor will not be able to sufficiently try out new experts' configurations. L must therefore be tuned along the other hyperparameters of the network.

Noise is introduced in the gating procedure by the ϵ term and the W_{noise} weight matrix in order to enhance the exploration of experts' configurations by a given investor. Since W_{noise} is trainable and shared by all investors' embeddings, it allows for a given investor to set the confidence it has in its attribution to given experts, and change it if not appropriate by setting high levels of noise for irrelevant experts. Borrowing from the reinforcement learning vocabulary, we can say that L enhances the *exploitation* of relevant experts, whereas W_{noise} enhances the *exploration* of experts attributions.

3.2 Driving experts specialization

Without a suitable additional loss term, the network has a tendency to let a few experts learn the same investment strategy, which also leads to more ambiguous mapping from investors to experts. Thus, to help the network finding different investment strategies and to increase its discrimination power regarding investors, we add a *specialization loss* term, which involves cross-experts correlations, weighted accordingly to their relative attribution probabilities. It

is written as:

$$L_{\text{spec}} = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} w_{i,j}\overline{\rho}_{i,j}$$
$$\text{with } w_{i,j} = \frac{\overline{p}_i\overline{p}_j}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\overline{p}_i\overline{p}_j} \text{ if } i \neq j, 0 \text{ else,}$$
$$\text{and } \overline{p}_i = \frac{\sum_{a\in A} p_i^a}{\sum_{a\in A} \mathbb{1}_{p_i^a\neq 0}}.$$

Here, $i, j \in \{1, \cdots, n\}$, $\overline{\rho}_{i,j}$ is the batch-wise correlation between experts i and j outputs, averaged over the output dimension, and $\overline{p}_i$ is the batch-wise mean attribution probability to expert i, with p_i^a the attribution) probability of investor a to expert i, computed on the current batch of investors that counted this expert in their top L only. The intuition behind this weight is that we want to avoid correlation between experts that were confidently selected by investors, i.e. to make sure that the experts that matter do not replicate the same investment strategy. As the size of the investors clustering around a given expert should not matter in this weighing, we only account for the top-L probabilities for all the considered investors in these weights. In some experiments, it was found useful to rescale L_{spec} from $[-1; 1]$ to $[0; 1]$.

This additional loss term is reminiscent of Liu and Yao [1999]. As a matter of fact, in ensembles of machine learning models, negatively correlated models are expected to perform better than positively correlated ones. This can also be expected from the experts of an ExNet, as negatively correlated experts better span the space of investment strategies. As the number of very distinct strategies grow, we can expect to find strategies that more closely match the ones the investors use in the considered market, or the basis functions on which investment strategies can be decomposed.

3.3 Further discussion on gating

Up to this point, we only discuss gating input related to investors. However, as seen above, being able to retrieve the structure of interactions *a posteriori* only requires to use categorical data as input to the gating part of the network. We can therefore perform gating on whatever is thought to be suitable —for instance, it is reasonable to think that bonds investors have different investment strategies depending on the bonds' grades, or depending on the sector of activity of the bonds' issuers. Higher-level details about investors could also be considered, for instance because investment strategies may depend on factors such as the sector of activity of the investor, i.e. whether it is a hedge fund, a central bank or an asset manager, or the region of the investor. The investor dimension could even be totally forgotten, and the gating performed on asset related categories only.

Gating allows one to retrieve the underlying structure of interactions of a given category, or set of categories. One can therefore purposely set categories to study how they relate in the problem one wants to study. This may however impact performance of the model, as chosen categories do not necessarily have distinct decision rules.

Note also that the initialization of weights in the gating network has a major impact on the future performance of the algorithm. To find relevant clusters, i.e. clusters that are composed of unequivocally attributed categories and that correspond to the original clusters expected in the dataset, categories need to be able to explore many different clusters' configurations before the exploitation of one relevant configuration. To allow for this exploration, the gating block must be initialized so that all the expert weights are fairly evenly initially distributed. In our implementation, we therefore use a random normal initialization scheme for the d-dimensional embeddings of the categories.

3.4 Limitations of the approach

Our approach allows us to treat well a known, fixed base of investors. However, it cannot easily deal with new investors, or, at a higher level, new categories as seen in Section 3.3, as embeddings for these new types of element would need to be trained from scratch. To cope with such situations, we therefore recommend to use sets of fixed categories to describe the evolving ones. For instance, instead of performing gating on investors directly, one can use investors' categories such as sector, region,..., that are already present in the dataset and on which we can train embeddings. Doing so improves the robustness of our approach to unseen categories. Note that this is reminiscent of one of the classic problems of recommender systems, known in the literature as the *cold start* problem.

4 Experiments

Before testing the ExNet architecture on real data, we first check its ability to recover a known strategy set, to attribute correctly traders to strategies, and finally to classify the interest of traders on simulated data. Then, we show how our methodology compares with other algorithms on BNP Paribas data.

4.1 Simulated data

4.1.1 Generating the dataset

Taking a cue from BNP Paribas CIB bonds' RFQ database, we define three clusters of investors, each having a distinct investment strategy, which we label as 'active', 'inactive' and 'common'. Each cluster contains a different proportion of investors, and each trader within a cluster has the same activity frequency: the 'active' cluster accounts for roughly 70% of the dataset samples, while containing roughly 10% of the total number of investors. The 'inactive' (a shorthand for 'not-very-active') cluster accounts for roughly 10% of the samples, while containing roughly 50% of the total number of investors. The 'common' cluster accounts for the remaining number of samples and investors. In all the clusters, we assume that investors are equally active.

We model the state of investors as a binary classification task, with a set of p features, denoted by $X \in \mathbb{R}^p$, and a binary output Y representing the fact that a client is interested or not in the considered asset. Investor a belonging to cluster c follows the decision rule given by $Y_a = (\sigma(w_a^T X) > U) \in \{0, 1\}$, where $w_a = w_c + b_a \in \mathbb{R}^p$, $w_c \in \mathbb{R}^p$ being the cluster weights and $b_a \sim \mathcal{N}(0, \alpha) \in \mathbb{R}$ an investor-specific bias, $X_i \sim \mathcal{N}(0, 1)$ for $i = 1, \cdots, p$, U is distributed according to the uniform distribution on $[0, 1]$, and σ is the logistic function.

The experiment is conducted using a generated dataset of $100,000$ samples, 500 investors and $p = 5$ features. This dataset is split into train/validation/test sets, corresponding to $70/20/10\%$ of the whole dataset. α is set to 0.5, and the cluster weights are taken as follows:

- Active cluster: $w_{active} = (5, 5, 0, 0, -5)$
- Inactive cluster: $w_{inactive} = (-5, 0, 5, 0, 5)$
- Common cluster: $w_{common} = (-5, 0, 5, 5, 0)$

These weights are chosen so that the correlation between the inactive and common clusters is positive, but both are negatively correlated with the the active cluster. In this way, we build a structure of clusters, core decision rules and correlation patterns that is sufficiently challenging to demonstrate the usefulness of our approach.

4.1.2 Results

We examine performance of our proposed algorithm, ExNet, against a benchmark algorithm, LightGBM [Ke et al., 2017]. LightGBM is a tree-gradient boosting algorithm, and one of the most widespread implementations of gradient boosting as measured by the percentage of Kaggle winning solutions using it. This algorithm is fed with both the experts input of the ExNet and an encoding of the considered investors, used as a categorical feature in the LightGBM algorithm. The LightGBM experiments will be conducted keeping in mind that boosting algorithms should use *weak learners*. For comparison purposes, experiments are also performed on an LightGBM model fed with experts input and an encoding of the investors' underlying clusters, i.e. whether the investor belongs to the active, inactive or common cluster, called *LGBM-Cluster*. We use a maximum depth of 20, and perform early stopping on the classification error with a patience of 10.

ExNets are trained using the *cross-entropy* loss, since the problem we want to solve is a classification one. The network is optimized using Nadam [Dozat, 2016], a variation of the Adam optimizer [Kingma and Ba, 2014] using Nesterov's Accelerated Gradient [Nesterov, 1983] and reintroduced in the deep learning framework by Sutskever et al. [2013]. We tried to vary the number of experts: we call *ExNet-n* an ExNet algorithm with n experts. Experts considered in this experiment do not have hidden layers, and inputs are batch-normalized [Ioffe and Szegedy, 2015]. Investors are embedded with $d = 64$. The network is trained using early stopping with a patience of 10, and a batch size of 512. All ExNet experiments are carried out with a learning rate set to e^{-3}, which was found to lead to satisfactory solutions in all the hyperparameter settings that we tested. The weight attributed to the specialization loss is $\lambda_{\text{spec}} = e^{-2}$, and we use *KeepTopL* with $L = 4$. For comparison purposes, experiments are also performed on a multi-layer perceptron model, fed with the experts inputs, concatenated with a trainable embedding of the investors — this model therefore differs from ExNet-1 in that ExNet-1 does not use an embedding of the investor to perform its predictions. Hyperparameters for this model, called here *Embed-MLP*, were found using a random grid of size 500 on the architecture, embedding

size and regularization parameters of the network. Results for this model correspond to the model which achieved best validation accuracy over all grid elements.

Algorithm	Train Acc.	Test Acc.	Active Acc.	Common Acc.	Inactive Acc.
LGBM	96.31	92.16	92.72	89.77	89.13
LGBM-Cluster	94.39	92.37	92.50	91.78	91.82
Embed-MLP	93.76	92.16	92.44	90.91	87.99
ExNet-1	74.91	74.59	80.46	48.50	37.58
ExNet-2	90.78	90.90	91.71	87.32	83.02
ExNet-3	92.76	92.74	92.86	92.22	93.06
ExNet-10	**92.79**	**92.80**	**92.88**	**92.43**	**93.17**
ExNet-100	92.78	92.71	92.78	92.38	92.86
Perfect model	93.68	93.72	93.86	93.09	94.93

Table 1: Experimental results on simulated data: accuracy of predictions, expressed in percentage, on train and test sets, and on subsets of the test set corresponding to the original clusters to be retrieved.

Table 1 contains the results for all the tested implementations. As the binary classification considered here is balanced, we use the accuracy as evaluation metric. This table reports results on train and test splits of the dataset, and a view of the test results on the three different clusters generated. As the generation process provides us with the probabilities of positive events, it is also possible to compute metrics for a model that would output these probabilities, denoted here as *perfect model*, which sets the mark of what good predictive performance is in this experiment. We see here that the LGBM implementation fails to completely retrieve the different clusters. LGBM focused on the active cluster and mixed the two others, leading to poorer predictions for both of these clusters. In comparison, LGBM-Cluster performed significantly better on the common and inactive clusters. Regarding ExNets, we see that ExNet-1 successfully captured the largest cluster, ignoring the two others. ExNet-2 behaved as the LGBM experiment, retrieving the largest cluster and mixing the remaining two. ExNet-3 perfectly retrieved the three clusters, as expected. Even better, the same holds for ExNet-10 and ExNet-100: this is because the ExNet algorithm, thanks to the additional specialization loss, is not sensitive to the number of experts, even if $n \gg K$, as long as there are enough of them. Thus, when $n \geq K$, the ExNet is able to retrieve the K initial clusters and to predict the interests of these clusters satisfactorily.

4.1.3 Further analysis of specialization

The previous results show that as long as $n \geq K$, the ExNet algorithm is able to capture the investment strategies corresponding to the underlying investor clusters efficiently. One still needs to check that the attribution to experts is working well, for example that the investors are mapped to a single, unique expert. To this end, we retrieved from the gating block the attribution probabilities to the n experts of all the investors *a posteriori*.

Figure 3 reports the attribution probabilities to each experts for all the investors, each expert corresponding to a different color. The first block of indices correspond to 'active' investors, the second block to 'inactive' ones and the third block to 'common' ones. We see in this figure that the ExNet-100 algorithm retrieved the original clusters particularly well, with the exception of an inactive investor that was effectively classified as a common one. One also notes that some inactive investors were attributed to a lower extent to the expert corresponding to the common group. This behaviour was expected from the high correlation between core decision rule of the inactive and common clusters, and from the fact that inactive investors all have a rather small weight in the dataset, thereby facilitating attribution issues. Nonetheless, overall mean attribution probability to the main expert is $\geq 98\%$ for all three clusters. ExNet-100 therefore solved the original problem entirely, retrieving original clusters, mapping them to one given expert and with good predictive performance for all clusters, as shown by the results obtained for ExNet-100 in Table 1.

An alternative way to visualize this result is to investigate how the investors interact in a two-dimensional plot showing their attributions to experts. We use here the UMAP algorithm [McInnes et al., 2018], which is particularly relevant as it seeks to preserve the topological structure of the embeddings' data manifold in a lower-dimensional space, thus keeping vectors that are close in the original space close in the embedding space, and making inter-cluster distances meaningful in the two-dimensional plot (contrarily to algorithms such as t-SNE [Maaten and Hinton, 2008]). The two-dimensional map given by UMAP is therefore a helpful tool for understanding how investors relate to each other. Figure 4 plots the UMAP; the active investors are shown in red, the inactive ones in green and the common ones in blue. One immediately sees well-defined monochromatic clusters. Whereas there is only one red ('active') cluster, there are two mildly separated green ('inactive') clusters, and two more widely separated blue ('common') clusters:

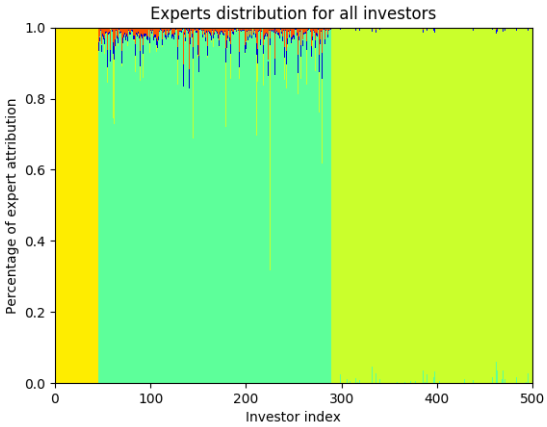

Figure 3: Distribution over experts of all investors for the ExNet-100 algorithm. Each column shows the attribution probabilities of a given investor, where colors represent experts. Investors 0-46 belong to the 'active' cluster, 47-290 to the 'inactive' cluster, and the remaining ones to the 'common' cluster.

these separations encode the fact that some common and inactive investors were marginally attributed to the expert corresponding to the other cluster.

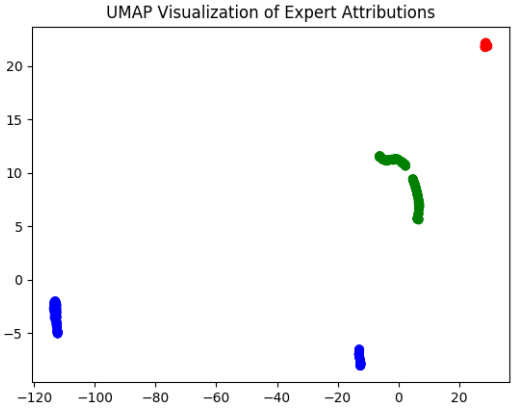

Figure 4: Visualization of investors' attribution to experts for ExNet-100 with the UMAP algorithm. Outputs of the gating block can be interpreted as *embeddings* of the investors. The active investors are shown in red, the inactive ones in green and the common ones in blue.

4.1.4 Influence of L and specialization loss

To study the influence of L and how the network behavior depends on the relative importance of specialization loss, we repeat the above experiment using an ExNet-100 twenty times for each combination of $L \in \{1, 2, 4, 8, 16, 32\}$ and $\lambda_{\text{spec}} \in \{0., e^{-2}\}$. This allows us to compute the confidence intervals of all the metrics.

ExNets with $L = 1$ cannot solve the task at hand, whatever the intensity of the specialization loss: because the learning process can only probe one expert at a time, it cannot explore and compare the relevance of experts, and thus the ExNet cannot find the relevant investor structure. For same reason, too small an L produces the same result.

Figure 5 shows the results for $L \in \{2, 4, 8, 16, 32\}$. The left plot shows the predictive performance of all the experiments, with, from left to right, overall performance on the train set, overall performance on the test set, and performance on the test set for all the original clusters. The right plot shows the attribution probabilities to the *main expert* for all the original clusters, i.e. the expert which, for a given investor, received the largest attribution probability. While ExNets without specialization loss obtained better average accuracies on both train and test sets, they were not

significantly better on test sets than those with specialization loss. It is also interesting to note that the best mean accuracy for the 'inactive' cluster was found for $L = 32$ with specialization loss. Increasing L, i.e. the *receptive field* of an investor over the different experts, helps achieving better performances on small-sized clusters —the 'inactive' cluster is the one with the fewer samples overall and with the fewer samples per investor. Concerning attributions, the right plot shows that for all the parameter combinations, specialization loss yields significantly better results for all the original clusters. As a matter of fact, the main expert attribution probability can be seen as a *replication index*. The smaller this probability, the more experts learnt the investment strategy of a same cluster, since the overall performances are satisfactory for all the setups considered in this experiment. Without specialization loss, ExNets replicated the active cluster particularly well. As this cluster and the corresponding investors weigh significantly more than all others, it might be more tempting for the network to try and capture these investors with more granularity unless suitably penalized; in other words, the specialization loss term helps enforcing diversity.

We can conclude from this experiment that the specialization loss helps the network to find the core behaviour of the original clusters and to map these clusters to one expert only, without significant loss of predictive power. Moreover, it appears that increasing L, i.e. investors' receptive field, also helps obtaining better predictive and attribution performances, a useful guideline to find suitable sets of hyperparameters for real data.

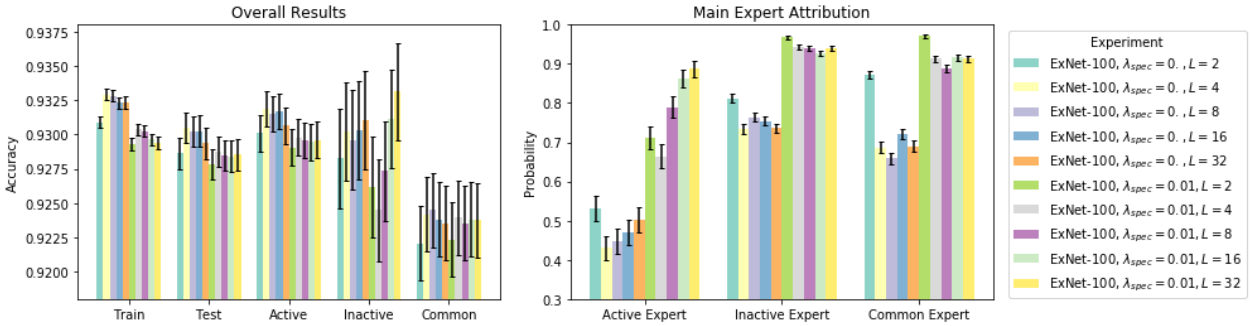

Figure 5: Synthetic data: prediction and attribution performances of ExNet-100 for different configurations of specialization loss weights and values of L.

4.2 BNPP data

The previous experiment proved the ability of the network to retrieve structure of investors polarizing around a finite set of fixed investment strategies. We will now examine how our algorithm performs on BNPP CIB data, using as benchmark the LightGBM algorithm.

4.2.1 Specifying the dataset

The investor interest classification problem *per se* is highly imbalanced. Indeed, most of the time, a given investor does not display any interest at all in any of the financial assets of a given market, leading to an over-representation of the "no interest" class. If we take for instance the problem of the bonds' RFQ database described in Section 1, this class accounts for 97% of samples, while "buy" and "sell" classes account each for 1.25% and "buy & sell" class accounts for only 0.5%. In such a setting, approaches such as the *focal loss* [Lin et al., 2017], an adaptive re-weighting of cross-entropy, lead to better results than the classic cross-entropy loss. This problem remains however a main challenge for predicting what can then be understood as rare events. For the sake of argument and to prove the usefulness of our approach, we therefore introduce here a slight variation on this problem so as to focus on the challenge that we wish to solve.

This experiment focuses on the equity options market. For a given investor, at a given date, for a given stock which we assume may be of interest to the investor and for a given maturity range, we aim to predict the *payoff structure*, along with the direction, in which the investor will be interested in the following business week, i.e. whether the investor is interested into buying or selling a call option, a put option, a straddle... in the next 5 business days. In this particular experiment, we consider 6 different payoff structures, leading to a 12-class classification problem, where classes are well balanced. Three tenor buckets are used: short, medium and long maturities, defined by observed quantiles of actual maturities. We consider here a perimeter of around 160 investors, interacting on about 450 assets. Features are formed as exposed in Section 1, using equivalent data sources. Algorithms are trained from Jan. 2016 to Aug. 2018,

validated from Sep. 2018 to Dec. 2018 and tested from Jan. 2019 to May 2019. The validation set is used as an early stopping set for both LightGBM and ExNet algorithms.

4.2.2 Results

Table 2 shows results of this experiment for the LightGBM and ExNet algorithms. Hyperparameters for both algorithms were found with Bayesian optimization [Frazier, 2018], where we model the validation cross-entropy as a function of the hyperparameters of the model with a Gaussian process, using the `scikit-optimize` library [Head et al., 2018]. Table 2 consequently shows LightGBM and ExNet results for models that achieved the best validation cross-entropy performance in 100 calls of the optimization procedure. The best LGBM had a maximum depth of 7, a subsampling ratio of 0.72, a maximal number of leaves of 2754 and a learning rate of 0.09. The best ExNet had $n = 32$ experts, $L = 8$, $\lambda_{\text{spec}} = e^{-2}$ with the specialization loss remapped to $[0, 1]$, investors' embeddings of dimension $d = 64$, experts with an hidden layer of size 16, a learning rate of $4e^{-3}$ and a batch size of 1024. Both algorithms used an early stopping patience of 20.

Algorithm	Train	Test
LGBM	63.92	33.31
ExNet	51.04	35.04

Table 2: Experimental results on BNPP data: average precision scores macro-averaged over the twelve classes considered for the LightGBM and ExNet algorithms, expressed in percentage.

We see here that LGBM seems to overfit the training set, since the ExNet obtained a better performance on the test set while having lower performance on the train set. Moreover, Fig. 6, which shows the distribution over experts for all investors and where a color corresponds to a given expert, exposes the investors' behavioral patterns captured by the ExNet. Of the 32 experts used in the network, 21 only are used by investors as their main expert, i.e. the expert to which they were attributed with maximal probability. The rescaled training specialization loss of 0.51, i.e. slight positive correlation, ensures that experts are not *replicas* of each other. A few behaviours emerge from this graph: we see many groups of investors polarizing around a given main expert. Noisy attributions appear as well, forming a group of 'indecisive', 'volatile', or hard-to-classify investors. These investors had on average 6 times less samples in the training dataset, making their embeddings harder to train, which is reminiscent of the above experiments on simulated data, where inactive investors had noisier attributions.

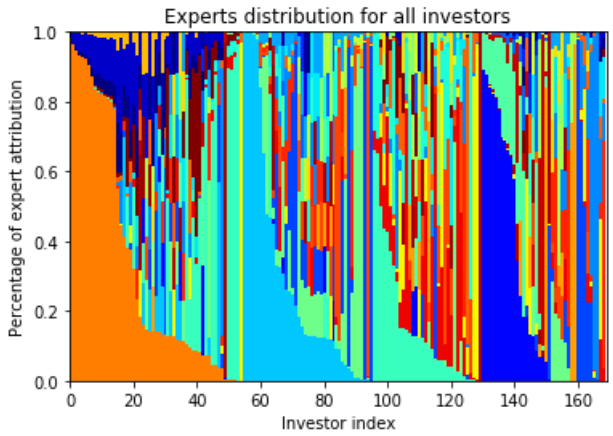

Figure 6: Distribution over experts for all considered investors in the real-world experiment. Each column shows the attribution probabilities for a given investor, where colors represent experts.

Figure 7 shows how investors' embeddings organize themselves on a two-dimensional map obtained using the UMAP algorithm. In this plot, each point corresponds to an investor, whose color is determined by the expert to which she is mainly attributed, with colors matching the ones in Fig. 6. We see in this figure that many clusters appear along with a group of more indecisive investors, as expected from the distribution over experts. Moreover, most of these clusters have a direct interpretation in terms of investors' characteristics: investors from the same sector of activity (asset managers, hedge funds, banks,...) of a given region (North America, Western Europe, Japan, Asia,...) tend to appear in the same clusters, which matches the expectations of financial intuitions. Therefore, the clusters appearing

in these plots can be used to obtain a better understanding of BNP Paribas CIB business, and how BNP Paribas CIB clients' are related to each other.

Note however that these maps (and ExNets) are built from measures of simultaneous distance, hence, do not exploit lead-lag relationships. How ExNets could be adapted to a temporal setting to retrieve lead-lag relationships will be worthy of future investigations.

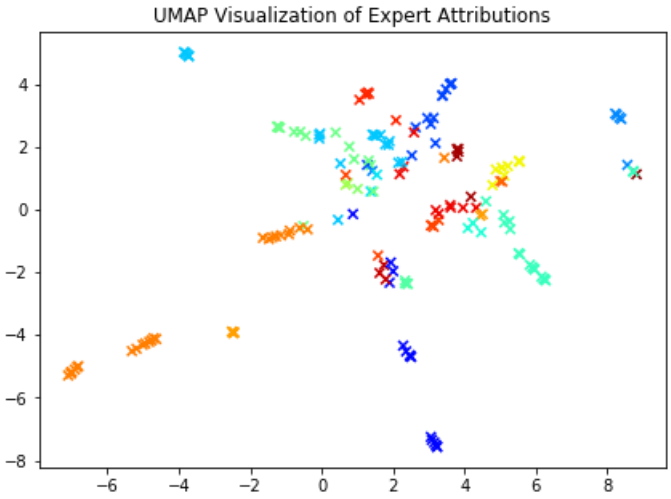

Figure 7: Visualization of investors' attributon to experts for an ExNet trained on real-world data. Colors correspond to the main expert to which investors were attributed.

5 Conclusion

We introduced a novel algorithm, ExNet, based on the financial intuition that in a given market, investors may act differently when exposed to the same signals, and cluster around a finite number of investment strategies. This algorithm is able to perform both prediction, be it regression or classification, and clustering at the same time. The fact that these operations are trained simultaneously leads to a clustering that most closely serves the prediction task, and a prediction that is improved by the clustering. Moreover, one can use this clustering *a posteriori*, independently, to gain knowledge as to how individual agents behave and interact with each other. To help the clustering process, we introduced an additional loss term that penalizes correlation between the inferred investment strategies. Thanks to an experiment with simulated data, we proved the usefulness of our approach, and we discussed how the ExNet algorithm performs on real-world data from BNP Paribas CIB. Further research on the subject will include how such architectures could be adapted to retrieve lead-lag relationships in a given market.

Finally, the ExNet architecture introduced in this article can be applied wherever one expects agents to use a finite number of decision patterns, e.g. in e-shopping or movie opinion databases [Bennett et al., 2007].

6 Acknowledgements

This work was conducted under the French CIFRE PhD Programme, in collaboration between the MICS Laboratory at CentraleSupélec and BNP Paribas CIB Global Markets. We thank Sarah Lemler, Frédéric Abergel and Julien Dinh for helpful discussions and feedback on early drafts of this work.

References

K. Baltakys, J. Kanniainen, and F. Emmert-Streib. Multilayer aggregation with statistical validation: Application to investor networks. *Scientific reports*, 8(1):8198, 2018.

J. Bennett, S. Lanning, et al. The Netflix Prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

D. Challet, R. Chicheportiche, M. Lallouache, and S. Kassibrakis. Statistically validated lead-lag networks and inventory prediction in the foreign exchange market. *Advances in Complex Systems*, 21(08):1850019, 2018.

J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

T. Dozat. Incorporating Nesterov momentum into Adam. 2016.

P. I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. Vinícius, cmmalone, C. Schröder, nel215, N. Campos, T. Young, S. Cereda, T. Fan, rene rex, K. K. Shi, J. Schwabedal, carlosdanielcsantos, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. Estève, L. Besson, M. Cherti, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller, and A. Fabisch. scikit-optimize/scikit-optimize: v0.5.2, Mar. 2018. URL `https://doi.org/10.5281/zenodo.1207017`.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

Y. Liu and X. Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):716–725, 1999.

L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

F. Musciotto, L. Marotta, J. Piilo, and R. N. Mantegna. Long-term ecology of investors in a financial market. *Palgrave Communications*, 4(1):92, 2018.

Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

J. Sirignano and R. Cont. Universal features of price formation in financial markets: Perspectives from deep learning. 2018.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

M. Tumminello, F. Lillo, J. Piilo, and R. N. Mantegna. Identification of clusters of investors from their real trading activity in a financial market. *New Journal of Physics*, 14(1):013041, 2012.

S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.