Comparison of models for Ubiquant Market Prediction

Jiabao Li
Department of Life Science, HKUST
Hong Kong SAR, China
jligm@connect.ust.hk

Zhihan Zhu
Department of Chemical and Biological Engineering, HKUST
Hong Kong SAR, China
zzhuay@connect.ust.hk

Abstract

Ubiquant Market Prediction is a competition held in Kaggle. In this competition, the characterized market data and the transaction features are provided with the competitors and there are many models as well as pipelines which win high grade for the leader board. However, there is no benchmark for comparing different models. In this study, we use the Ubiquant Market dataset to implement a method to compare with different models such as Least Absolute Shrinkage and Selection Operator (LASSO), Gradient Boosting Decision Trees (GBDT), Deep neural network (DNN) and Long short-term memory (LSTM). For the implementation, the training dataset and the validation dataset are firstly defined and then the validation for the models are adopted independently. Through our methods, we found that GBDT model performs best among all the candidate models.

Contents

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1 Introduction

Ubiquant market prediction is a competition help on Kaggle platform, providing real historic prices from many investments, and competitors are asked to build a model for investments' return rates prediction. The transaction time, investment ID and 300 features are provided. Totally, there are 3141410 samples of transaction with investment in the Ubiquent dataset, which is also RAM-consuming and takes time to load the data. In this studies, we compared four models with optimized hyper-parameters, and selected the best one for the competition.

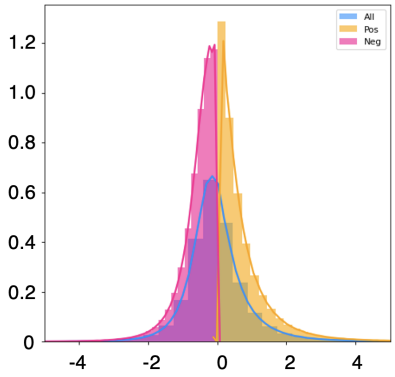

Figure 1: Distribution for the target value

There are many exploratory data analysis on the return rate. The target value is concerned in this study and the distribution for the target value by whether it is negative or positive is shown in the above figure. From the figure, the target value is normally distributed while the negative value and the positive value are right sided and left sided distributed with tails.

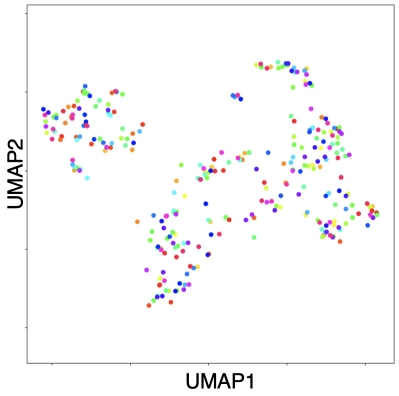

Figure 2: UMAP projection of 300 Ubiquant features

The nonlinear dimensionality reduction method is performed by Uniform Manifold Approximation and Projection (UMAP). The UMAP is performed with umap packages after the characterization of 300 features of 3141410 transaction data. The UMAP projection for 300 features of the Ubiquant dataset is shown in the figure and the scatters with different color represent 300 Ubiquant features named f_1 to f_300 in Ubiquant prediction dataset. From the scatter plot, there is no evident clusters of 300 Ubiquant features and some features are overlapped to each other.

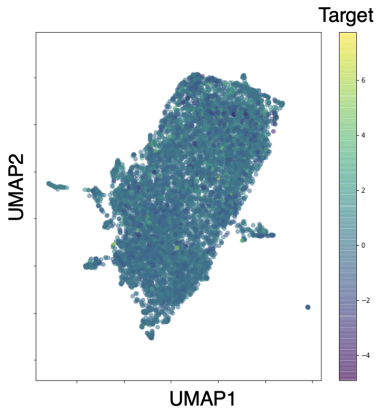

Figure 3: UMAP projection of 10000 Ubiquant data

The Ubiquant data are visualized with UMAP projection as well. Since there are millions of data, calculating the local distance is time-consuming. Ten thousand Ubiquant data is then collected for the UMAP projection and each of them is in the 300 dimension in feature space. The target value of each transaction is colored with gradient color. The scatter plot shows that the Ubiquant data with different target values are mixed together and no apparent clusters with target value preferences are found, indicating that this Ubiquant dataset may be difficult to be predicted and some models with advanced structure should be considered.

2 Methodology

We tested 4 models in this study, including Least Absolute Shrinkage and Selection Operator, Gradient Boosting Decision Trees, Deep Neural Network and Long Short Term Memory network, and for each model, the hyper-parameters were optimized by cross-validation.

2.1 Model Selection

To find the optimal model for this dataset, we randomly selected 25% of data as independent test dataset. The remained 75% of data were used for hyperparameter selection. 3-fold cross-validation was used to grid-search the combination of hyper-parameters with highest average Pearson correlation coefficient.

2.2 Least Absolute Shrinkage and Selection Operator

Least Absolute Shrinkage and Selection Operator (LASSO) is a regression model predicting by

$$\hat{y}_i = \sum_j \beta_j x_{ij}$$

that can achieve regularization during the training process, to reduce the chance of over-fitting, by adding a penalty part in the objective function L (1):

$$L = \sum_{i=1}^{n} \left(y_i - \beta_0 c - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \alpha \sum_{j=1}^{p} |\beta_j|$$

Here, $\alpha \sum_{j=1}^{p} |\beta_j|$ is the regularization part of the objective function L, and α controls the strength of regularization. We optimized α and c during the hyper-parameter tuning, with scopes of [0.001,0.01] and [0,1] accordingly. The training average AUC was shown in Figure 1.

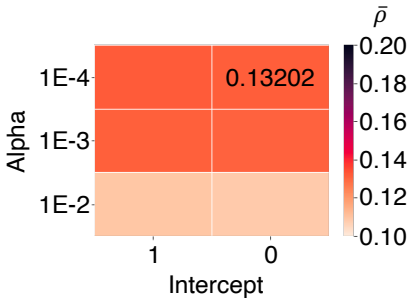

Figure 4: Average ρ of LASSO model in 3-fold cross-validation

2.3 Gradient Boosting Decision Trees

Gradient Boosting Decision Trees (GBDT) is one of the decision tree ensemble learning algorithms, iteratively generating weak decision trees and finally combining these weak decision trees to a better model:

$$\hat{y}_i = \sum_{t=1}^{T} f_t\left(x_i\right), f_t \in \mathcal{F}$$

For each iteration step t, a new decision tree $f_t\left(\mathbf{x}_i\right)$ is generated by minimizing the objective function $L^{(t)}$ (2):

$$L^{(t)} = \sum_{i=1}^{n} \left(y_i - \hat{y}_i^{(t-1)} - \alpha f_t\left(\mathbf{x}_i\right)\right)^2 + \Omega\left(f_t\right)$$

Here, $\hat{y}_i$ represents for the label of sample i and $\hat{y}_i^{(t-1)}$ represents for the ensembled prediction for sample i obtained at the iteration step $t-1$. Some hyper-parameters, including the learning rate α and the maximal tree depth D were optimized in this study. For learning rate α, we tested 0.01, 0.1 and 1. For maximal tree depth D, we tested 1, 10. The training average AUC was shown in Figure 2.

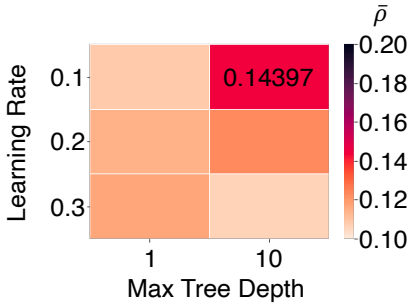

Figure 5: Average ρ of GBDT model in 3-fold cross-validation

2.4 Deep Neural Network Model

Deep neural network (DNN) (3) model is adopted in predicting the target value for the Ubiquant market data. The model pipeline is folked from Kaggle::graydolphin (4), which contains the input, output and model APIs by PyTorch Lightning. To find the robust results for the prediction, two learning rate and two model structure of DNN models are considered.

In the data pre-processing part, the investment id and all the features are applied in the pipeline. The investment id is embedded into the 11 dimension in the first part.

$$X_{N\times 11} = Embedding(X_{N\times 1, investment})$$

where N denotes the total number of the transactions in Ubiquant market dataset. Then, 300 features of investment trades are concatenated into 311 features.

$$X_{N\times 311} = [X_{N\times 11} \quad X_{N\times 300, features}]$$

The 311 features in the Kaggle::graydolphin pipeline will be the input of our DNN models. The linear layer with the activate function of SiLU are used in the model. For DNN model 1, there are seven linear layers sequentially and around 360,000 parameters included. DNN model 2 is simpler than DNN model 1 with five linear layers and about 96,400 parameters. The SiLU activation function is newly used in the Kaggle::graydolphin pipeline.

$$silu(x) = x \cdot \sigma(x)$$

and

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Compared with ReLU activation function, SiLU function have non-zero value when the inputs are less than zero. SiLU is simmilar to the ELU activation function as for the activated value range. Finally, the output of predicted target value will be generated by a linear layer. In addition, we use a self-defined loss function in the training process.

$$L(\hat{y}, y) = 1 - cor(\hat{y}, y)$$

where

$$cor(x, y) = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}}$$

The optimizer is set as Adam with learning rate of 0.001 and 0.005. Since the learning rate will influence the training speed and consequences of final optimized models (such as the model weights and the way how model parameter change), the learning rate is considered and compared in the models. Totally, four models could be acquired in DNN models: complex DNN model with learning rate of 0.001, complex DNN model with learning rate of 0.005, simple DNN model with learning rate of 0.001 and simple DNN model with learning rate of 0.005.

In the training process, the training dataset and the validation dataset are split from the original dataset of Ubiquant market prediction by Zhihan. The validation dataset is to compated with different model individually as benchmark and the training dataset is to train the model in three-fold cross validation. The pearson correlation of the predicted target value and the target value.

Table 1: Results for DNN models

Model	Model Complex	Learning Rate	Pearson Cor	Name
DNN	complex	0.001	0.15191829	V4
DNN	complex	0.005	0.1302046	V42
DNN	simple	0.005	0.13668779	V43
DNN	simple	0.001	0.14695018	V44

In the training process, the validation loss decreased gradually and the Pearson correlation coefficient for the predicted value and the target value increased. The table for the model and the learning rate could be shown as follows:

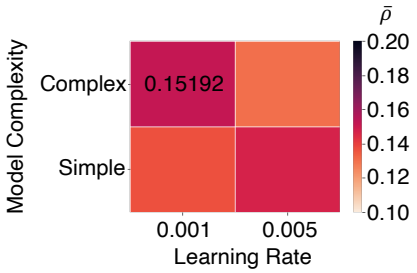

Figure 6: Average ρ of DNN model in 3-fold cross-validation

2.5 Long Short-Term Memory model

The Long short-term memory (LSTM) model use the recurrent neural network structure and is good at dealing with time series data (5). In this study, we try to introduce the LSTM layers into the baseline models from Kaggle::graydolphin (4), which is the same baseline as DNN models. The LSTM layer is implemented by the pytorch module with nn.LSTM, and the function for calculating the hiddenstate is the same with the LSTM model(6).

In LSTM layers, the hidden state will be the input for the next network layers such as a follow-up LSTM layer or a linear layer. The input sequences are generated by the linear layers after the feature embedding. The LSTM models are included the LSTM with high dimensional hidden state as the complex model and the low dimensional hidden state as the simple model. Other parameters for the LSTM layer structure set as defaults. The loss function is the same with DNN model, which is as follows:

$$L(\hat{y}, y) = 1 - cor(\hat{y}, y)$$

The optimizer is Adam optimizer in the training process. The learning rate of 0.001 and the 0.005 are included in the models. In all, there are four LSTM models: the complex structure with 256 hidden states and the simple structure with 128 hidden states. Both of them are trained with learning rate of 0.001 and 0.005.

Table 2: Results for LSTM models

Model	Model Complex	Learning Rate	Pearson Cor	Name
LSTM	complex	0.001	0.14113208	V5
LSTM	complex	0.005	0.11254942	V52
LSTM	simple	0.001	0.14182154	V53
LSTM	simple	0.005	0.12764723	V54

The three fold cross validation in the training process is adopted, which is split from the Ubiquant original training dataset. For the independent validation dataset, the remaining dataset is to validate the LSTM models and the Pearson correlation coefficients are calculated for each model. The results are displayed as follows:

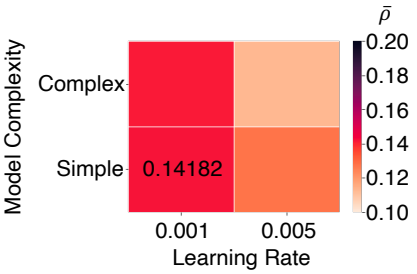

Figure 7: Average ρ of LSTM model in 3-fold cross-validation

3 Results

3.1 Conclusions

We applied four hyperparameter-optimized models in the independent dataset, which contained 25% of the data provided by the competition organizer. The Pearson correlation coefficient ρ was computed for all models, and shown in Table 3:

Table 3: Model performance

Model	ρ in validation dataset	ρ in Kaggle
LASSO	0.13063	0.1076
GBDT	0.15513	0.1193
DNN	0.15191	0.1348
LSTM	0.14182	NA

GBDT and DNN showed much better performance than LASSO and LSTM model, which is consistant with the results in the cross-validation. Thus, the results from GBDT and DNN were submitted as the final prediction in the competition.

3.2 Case Study

Case study for LSTMv53 model is to compare the real target value and the predicted target value. The validation dataset is applied in LSTMv53 model to make the prediction of the targets. In the distribution plot shown in the following figure, each plot represents one transaction and the points will be scattered according to the predicted target value and real target value. The distribution for the predicted target value and the real target value are displayed as well. The linear regression for the target value and the predictions is performed and visualized as well.

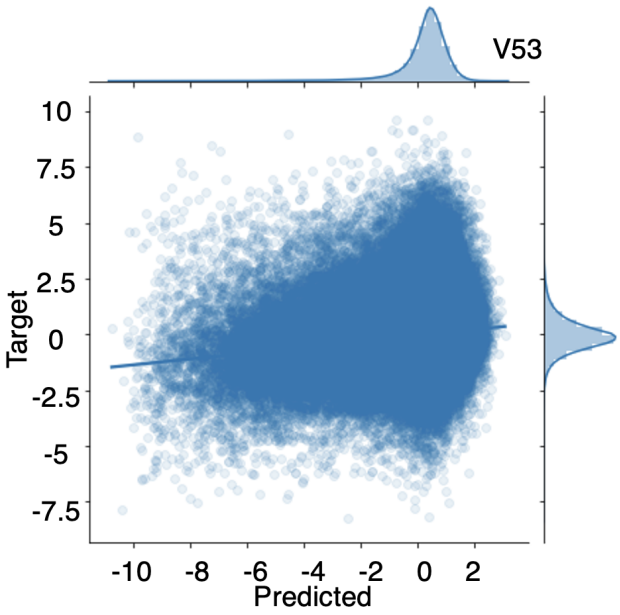

Figure 8: Prediction vs target in LSTMv53 model

From the above figure, the distribution for the target value and predicted value is approximately normal distribution by eyes. There are many data points which are not related to the target data and lead to the low Pearson correlation coefficient of the models.

4 Discussion

In this competition, the metric for the models from all participants shows that the Pearson correlation coefficient are pretty low, which is less than 0.2 among all the models. By our method, the best optimized model is selected. From the case study, it shows that many transactions are not well predicted and it may due to the limited features. Even though there are a huge number of transactions, the features are not enough, which is also discussed in Kaggle forum. In the future, some related public data may be included to improve the performance.

Leaderboard and scores

For the simple models, the best score is by DNN models. And generally the best score is 0.1554 with other pipeline and rank in 386/2885 (13.4%) in public dataset.

Contribution

Jiabao Li trained and optimized the DNN and LSTM models. Zhihan Zhu trained and optimized the LASSO and GBDT models.

References

[1] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288.

[2] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[3] Chong, E., Han, C., Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems with Applications, 83, 187-205.

[4] Leonweninger. (2022, March 3). Clean pytorch lightning dnn [lb 0.144]. Kaggle. Retrieved April 20, 2022, from https://www.kaggle.com/code/leonweninger/clean-pytorch-lightning-dnn-lb-0-144?scriptVersionId=89198555

[5] Wei, D. (2019, October). Prediction of stock price based on LSTM neural network. In 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM) (pp. 544-547). IEEE.

[6] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.