NEURAL OBLIVIOUS DECISION ENSEMBLES FOR DEEP LEARNING ON TABULAR DATA

Xinxing Cheng, Jiajie Chen, Shengyi Yang
Department of Electronics and Computer Science
University of Southampton
{xc5u19, jc8n18, sy1g19}@soton.ac.uk

ABSTRACT

With the development of AI International Conference on Representation Learning (ICLR) produces a large number of papers every year. Thus, ICLR runs a challenge encouraging researchers to reproduce the work presented in papers submitted to the conference. There is a paper published as a conference paper at ICLR 2020, the title is 'NEURAL OBLIVIOUS DECISION ENSEMBLES FOR DEEP LEARNING ON TABULAR DATA'. Our team improved the proposed NODE framework and implemented a significant amount of analysis and experiment. We find NODE architecture does not outperform the CatBoost and XGBoost on most of tasks and believe NODE still have challenges to become a universal framework for machine learning to tabular data.

1 INTRODUCTION

It widely believed that the deep networks have significant achievement in various fields, including computer vision, NLPs. However, there are no DNNs application to the tabular data. In this report, the NODE architecture is proposed by Sergei Popov will be experimented to evaluate whether the NODE has a great performance on the tabular data. This report consists of three sections. Section one is the experimentation and analysis of the proposed NODE model related to what we will reproduce. Section two will give the information about the experimental methodology such as baselines, implementation details, data set and analysis. The final section is a conclusion that the performance of NODE architecture compared by Xgboost and Catboost. The finding is that the NODE framework still has a big gap to become a universal framework.

2 EXPERIMENTATION AND ANALYSIS

2.1 TARGET QUESTIONS

Is the datasets provided by the author suitable for reproduction? The author's implementation is inefficient in memory and may require a lot of GPU memory to converge(Sergei Popov). However, the specific requirements of the GPU are not clearly given.

Whether additional datasets should be tested? The conclusion in (Sergei Popov) is mainly based on six data sets, and the accidental nature of the datasets will have a huge impact on the conclusion. It may be difficult to fully prove the correctness of the author's views using only six datasets.

Is the NODE framework better than XGboost and CatBoost on most tasks? (Sergei Popov) compared the NODE framework with many popular frameworks and mainly adopted three methods: CatBoost, XGBoost and FCNN.

Can the NODE framework be improved? The author's code is not concise, a large number of for loops increase the model training time, and the data visualisation function may need to be optimised.

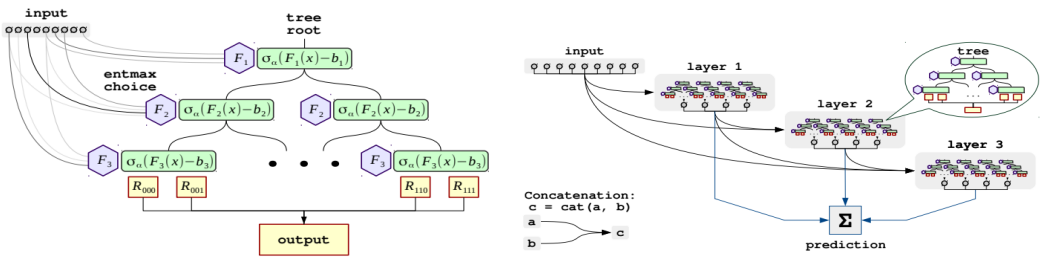
https://github.com/COMP6248-Reproducability-Challenge/NODE_FOR_DL_ON_TABULAR_DATA

Can NODE become a universal framework? The number of experiments may not be sufficient, the maintainability of the framework may be poor, and NODE may face many difficulties to become a universal framework.

2.2 EXPERIMENTAL METHODOLOGY

- **Framework optimization**: Accelerate training, smooth data visualization, and simplify code writing.
- **Extensive testing**: Multiple tests on XGBoost, CatBoots and NODE using existing and additional datasets.

2.3 IMPLEMENTATION DETAILS



(a) ODT (b) NODE architecture

Figure 1: The structure of NODE model. (Sergei Popov)

Neural oblivious decision ensembles (NODE) is the key part of the model author proposed. As show in figure 1 (a), the vector $x \in R^n$ with features n are split and passed into m differentiable ODTs with depth d. the ODT which is a decision table is responsible for splitting the data features, and learning the threshold by comparing with each feature. The α-entmax function is used to learn sparse choices.

The structure of neural network resembles the fully connected layer to organise a multi-layer architecture with k NODE layers. The figure 1 (b) indicated that the output of each layer is concatenated and passed into the next layer as the input, and each layer incorporates with the certain the number of trees. The prediction simply sum up and average the all output from all decision trees. The structure is both suited to the deep and shallow rules which is capable of learning complicated dependencies.

To evaluate the performance of the NODE on tabular datasets, the Catboost (Prokhorenkova et al. (2018)) and XGBoost (Chen & Guestrin (2016)) have been used as the baselines. New datasets (Shelter, Adult) were added for extensive testing. According to the regimes in paper, the datasets were tested by the baselines with the default hyperparameters and the baselines with hyperparameters tuned by the Hyperopt library. Each dataset was tested over ten runs with different random states (Bergstra et al. (2013)) using XGBoost, CatBoost and NODE.

Our team used three google colab pro accounts to complete the experimental test. We found that the yahoo data set will generate errors when normalising, so the yahoo data set is not Normalised. The total number of trees is 2048, 2-layer NODE architecture is used for classification data sets, and 8-layer NODE architecture is used for regression data sets.

2.4 ANALYSIS

By analysing the NODE framework, the style of the code is redundant which has lots of unused functions. In term of the structure of ODT and DensBlock, we just go through it and figure out whether it is reasonable for the approach author proposed. Then re-implement the content and tiny them up to make them simpler, so there are no much changes on those parts. However, there are some rewrite and improvements for the code which is listed in the below.

`https://github.com/COMP6248-Reproducability-Challenge/NODE_FOR_DL_ON_TABULAR_DATA`

Data set preprocessing. The function of downloading and processing are adopted to minimise the redundant code. Additionally, we defined a customer data set class to transform the processed dataset to tensor loader which is easier to pass it for torchbearer trial function.

Torchbearer for trial. The other change is using the torchbearer framework to trial our model replacing the custom evaluation. The callback function should be defined to early stop and save the best model before the training data over-fitting, and record the loss per each training step and each validating epoch.

Tensorboard for visualisation. The author plot the loss value for each training step and the valid loss during the process of gradient descent, which has significant influences on the running speed. Our method provided two way of presenting the running loss (accuracy) for training and validating data. The first one is to plot the recorded history after running, the history is also be written into a file for later inspection. The second way is to write the running log to tensorboard and the dynamic visualisation can be viewed in the localhost server.

File store. Besides the model and loss history is stored into files, the torch data loader and processed data set also be saved, which speeds up for later testing with different parameters.

There are two reasons for adding new datasets. On the one hand, the Epsilon dataset has 2000 features and a size of 11.3G. After testing, it takes more than 10 hours to run on tesla p100-pcie-16gb once, and the training sample number of the Higgs dataset is 10.5M, which also needs to be occupied lots of GPU resources. On the other hand, the new data set can be used to prove whether NODE has superior performance on other data sets.

It is mentioned that all experimental data were calculated on the GPU. For the regression task, the number of trees of XGBoost was set to 2048, the objective of CatBoost was set to RMSE. For the classification task, the XGBoost keep the default hyperparameters while the Catboost use the cross-entropy loss for the binary classification and 'MultiClass' loss function for multi-label classification.

The Hyperopt library should be used to fine-tune the hyperparameters of the baselines. However, it is found that the search space in the paper has issues. For the Catboost, the hyperparameters 'learning rate' provided in the paper is the log-uniform distribution of $[e^{-5}, 1]$. In experiments, it was found that the original learning rate was too large which could decrease the performance of the model. The learning rate is adjusted to the uniform distribution between [0.05,0.3], which could give a better results on datasets. For the XGBoost, the hyperparameters 'min child weight' in the paper is the log-uniform distribution between $[e^{-16}, e^{-5}]$, which leads to the out-of-range issue. Therefore, it is adjusted it to log-uniform distribution between $[e^{-16}, e^{-4}]$ so that the model could be tuned correctly.

2.5 DATA SET

	Train	Test	Features	Task	Metric	Description
AdultCensus	26K	6.5K	14	Classification	Error	Kaggle Adult Census Income
ShelterAnimal	26.7K	11.5K	5	Classification	Error	Kaggle Shelter Animal Outcome
YearPrediction	463K	51.6K	90	Regression	MSE	Million Song Dataset
Microsoft	723K	241K	136	Regression	MSE	MSLR-WEB10K
Yahoo	544K	165K	699	Regression	MSE	Yahoo LETOR dataset
Click	800K	200K	11	Classification	Error	2012 KDD Cup

Table 1: The datasets used in our experiments.

3 REFLECTION AND DISCUSSION

The experiment results of the paper and our implementation have been displayed in the above tables, which were computed over ten runs with different random states.

For the baselines with default parameters, it is found that our implementation performs better than results in paper, which could be resulted from we define the loss function and use the GPU to accelerate calculations. For the baselines with tuned parameters, it is found that our implementation

`https://github.com/COMP6248-Reproducability-Challenge/NODE_FOR_DL_ON_TABULAR_DATA`

	YearPred	Microsoft	Yahoo	Click
Default parameters				
CatBoost	80.68±0.04	0.5587±2e-4	0.5781±3e-4	0.3438±1e-3
XGBoost	81.11	0.5637	0.5756	0.3461
NODE	**77.43±0.09**	**0.5584±3e-4**	**0.5666±5e-4**	**0.3309±3e-4**
Tuned Parameters				
CatBoost	79.67±0.12	0.5565±2e-4	0.5632±3e-4	0.3401±2e-3
XGBoost	78.53±0.09	**0.5544±1e-4**	**0.5420±4e-4**	0.3334±2e-3
NODE	**76.21±0.12**	0.5570±2e-4	0.5692±2e-4	**0.3312±2e-3**

Table 2: Experiment results in the paper

	Adult	YearPred	Shelter	Microsoft	Yahoo	Click
Default Hyperparameters						
CatBoost	**0.1267±1e-3**	80.3748±0.1	0.4676	0.5595±6e-4	0.5702±5e-4	**0.3318±1e-3**
XGBoost	0.1336	80.6921	0.3591	0.5629	0.5749	0.3353
NODE	0.1345±2e-3	**77.3798±3e-1**	**0.3247±2e-3**	**0.5625±3e-3**	**0.5666±6e-4**	0.3319±1e-3
Tuned Hyperparameters						
CatBoost	**0.1237±8e-4**	79.72±0.1	0.3248±0.02	0.5569±4e-4	0.5670±4e-4	0.3311±1e-3
XGBoost	0.1283±3e-3	79.01±0.4	0.2578±0.02	**0.5602±1e-3**	**0.5537±7e-3**	**0.3305±1e-3**
NODE	0.1336±3e-3	**76.41±0.7**	**0.2578±9e-5**	0.5609±9e-4	0.5754±9e-4	0.3316±8e-4

Table 3: Our Experiment Results

performs worse than results in the paper, which could be resulted from that the hyperparameter space used for tuning in our experiments were smaller as mentioned in implementation details.

Node only performs better than XGBoost and CatBoost with default parameters in YearPred, Shelter Microsoft and Yahoo domain. With Turned parameters, the NODE framework does not show particular advantages. On the contrary, XGBoost has an advantage in Turned paramter and win in YearPred, Microsoft and Click domain. We also find XGBoost can achive better outcomes than authors', especially in Click domain. Therefore, different data sets have a great influence on the success of the NODE framework. The author's six data sets are too accidental to be used as evidence that NODE is superior to XGBoost and CatBoost.

4 Conclusion

To sum up, the NODE architecture could outperform the baselines in the most of tested datasets. However, the time and the GPU memory consumption would be a huge problem for the NODE architecture. The performance also vary for different data set, which do not beat the Xgboost or Catboost in some situation. For now, it seems that NODE only suits the tiny datasets. Therefore, in order to become practical for most tabular datasets, the NODE architecture still needs to be optimised in many aspects. The NODE framework cannot be competent a deep learning framework.

References

James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, pp. 13–20. Citeseer, 2013.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pp. 6638–6648, 2018.

Artem Babenko Sergei Popov, Stanislav Morozov. Neural oblivious decision ensembles for deep learning on tabular data.

`https://github.com/COMP6248-Reproducability-Challenge/NODE_FOR_DL_ON_TABULAR_DATA`