Neural Oblivious Decision Ensembles for Drugs Classification Based on Biological Activity

Zekun Chen[1], Giuseppe Barbalinardo[2], Yiming Wu[3], Chenghan Sun[4]

[1]zkuchen@ucdavis.edu, [2]gbarbalinardo@ucdavis.edu
[3]scywu@ucdavis.edu, [4]chesun@ucdavis.edu

Abstract

We apply three state-of-the-art machine learning models in the study of the Mechanisms of Actions for drug discovery. Starting from information about how treatments are applied to different cells, we predict their responses to various drugs by comparing the predictions from a deep neural networks model, a gradient boosted decision trees model, XGBoost, and the novel deep learning architecture Neural Oblivious Decision Ensembles. By comparing their predictive power and the performances on the publicly available Mechanism of Action dataset, which is identified to contain highly correlated features, we highlight the advantages of Neural Oblivious Decision Ensembles model over the other two baselines. In this systematic study, such model achieves a logarithm loss of 0.01723, where the XGBoost and the deep neural network baseline models have log losses of 0.01850 and 0.0205 respectively. These metrics reasonably proves that Neural Oblivious Decision Ensembles model has the best performance for strongly correlated tabular data, consistent with recent results in the literature.

1 Introduction

In recent years, deep neural networks (DNN) have been constantly producing breakthrough results for several machine learning (ML) tasks, including computer vision, natural language processing, speech recognition and reinforcement learning [1]. On the other hand, comparing with widely used DNNs, tree model still has some advantages due to its simple structure and explainable decision process. Especially for tabular data, gradient boosted decision trees (GBDT) [2] models usually have better accuracy than deep networks, which is verified by many Kaggle competitions and real applications. This gap between DNN models and tree-based models yields interests of exploring the effectiveness of DNN on tabular heterogeneous data, which remains to be a debated topic [3, 4]. The features constructed from tabular data are often correlated, so a small subset of features are responsible for most of the predictive power. Moreover, there is usually a strong class imbalance in the labels. For these reasons shallow models based on decision trees are still considered the go-to choice to approach these kinds of tasks.

In this project we aim to facilitate new drug development, improving the Mechanism of Action (MoA) prediction algorithms starting from a tabular dataset of experimental features. The MoA is a property of a drug, which helps to identify a protein target associated with disease and develop a molecule to modulate that protein target. Improving algorithms to predict such property can advance drug development by classifying drugs based on their biological activity. We aim to provide predictive way to estimate MoA for a given set of drugs, by comparing three modern machine learning (ML) algorithms, DNNs, the gradient boosted decision trees algorithm XGBoost [5], and the recent Neural Oblivious Decision Ensembles (NODE) model [6].

1.1 Dataset

The dataset is provided by a collaboration between the Laboratory for Innovation and Science at Harvard, the NIH Lincs program and the ConnectivityMap project[1]. Given information about experimental samples, consisting of gene expressions levels, cell viability measurements, compounds (or control) used, duration and dosage of a treatment, our goal is to predict how each sample responds to a provided drugs, for example an analgesic, an antibiotic, or cannabinoids. The features dataset consists of a collection of 23814 recorded assays. Each of them is composed of the following features:

- A sample identification code.
- An indication if the sample have been treated with a compound or with a control perturbation. Control perturbations have no MoA.
- The duration of the treatment in hours, i.e. 24, 48, 72.
- An indication of the amount of the dose tested, i.e. low or high.
- 772 values between -10 and 10 indicating the gene expression. The gene expression is the process by which the information encoded in a gene is used to direct the assembly of a protein molecule. Naively, this can be translated as how "hard" a particular gene is working in the particular cell, or group of cell. All the cells in a organism have the same genes, and one cell differs from another only by these gene expression.
- 100 values for indicating the cell viability measured for each cell line or culture, with a number between -10 and 10. The cell viability is a measure of the proportion of live, healthy cells within a population, and these assays are used to determined the overall health of cells, i.e. the cell survival information following the treatment with the compound.

Out of these 23814 recorded samples, 1866 samples are identified as control perturbations and their corresponding prediction targets are not provided and thus the total number of trainable samples is 21948. The targets dataset consists of a list of 21948 trainable records, one for each sample, indicating the response of the sample to one of the 206 drugs tested. The response is stored as a value between 0 and 1, where 1 indicates a measurable experimental response and 0 indicates the absence of a response.

[1]The dataset is publicly available for download at kaggle.com/c/lish-moa/data

2 Methods

2.1 Models

2.1.1 Deep Neural Network

Our first approach consists of adopting a DNN as our primary baseline model. The architecture of the network is made of a sequence of 4 dense layers, 1 normalization layers, and 5 dropout layers acting as regularizers. In the dense layers, the RELU [7] activation function is used. In the normalization layer, Lee-Cun normalization [8, 9] function is used such that samples can be drawn from a truncated normal distribution centering on 0. Meanwhile, the Scaled Exponential Linear Units (SELU) activation function is used in the normalization layer. Due to the relatively complex dimensions in both number of samples and number of features (i.e. 21948×877), dropout layers well prevents overfitting. Each dropout layer is set alternatively with the dense and normalization layers. The dropout rate for each of the layer is set to be 0.6. The 206 dimensional output layer, each dimension corresponding to one of the 206 drugs tested, is built along with a sigmoid activation function. Since in each entry of the output predictions, a binary response will be generated, using sigmoid is appropriate because the probabilistic values computed from the sigmoid function can effectively differentiate two discrete classes. In terms of optimizer, an Adam optimizer with Nesterov momentum (NAdam)[10] is used. Lastly, the number of total trainable parameters for our DNN base line model is $522,206$.

2.1.2 XGBoost

Our second approach aims to build classification models for the MoA dataset with XGBoost, the leading GBDT package, so that it can be used as a baseline model to compare with NODE for classification tasks with tableted data [6]. Our architecture in this XGBoost model is an collection of 206 multivariate XGBoost models, one model for each of the 206 drugs tested. In each model, a minimum child weight of 30, an α learning rate of 0.05, a 0.65 column sample by tree, a 3.705 minimum loss reduction (γ), a maximum depth of 10 and 170 estimators are used. The subsample rate for our models is 0.9. Upon training XGBoost classifiers, feature importance scores can be computed. In this study, these feature importance scores are used by the following NODE model for top feature extractions and seek for potential model improvement.

2.1.3 Neural Oblivious Decision Ensembles (NODE)

Our third study aims to predict MoAs using the Neural Oblivious Decision Ensembles (NODE) architecture, which consists of differentiable oblivious decision trees (ODTs) that are trained and optimized end-to-end by back-propagation, made it an effective technique for studying tabular data. The essential building block of one NODE layer is a single ODT. ODT is a decision table of 2^d entries that splits the data along d splitting features and compares each feature to a learned threshold. Since the tree returns one of the 2^d possible responses, corresponding to the comparisons result, each ODT is completely determined by its splitting features $f \in \mathbb{R}^d$, splitting thresholds $b \in \mathbb{R}^d$ and a d-dimensional tensor of

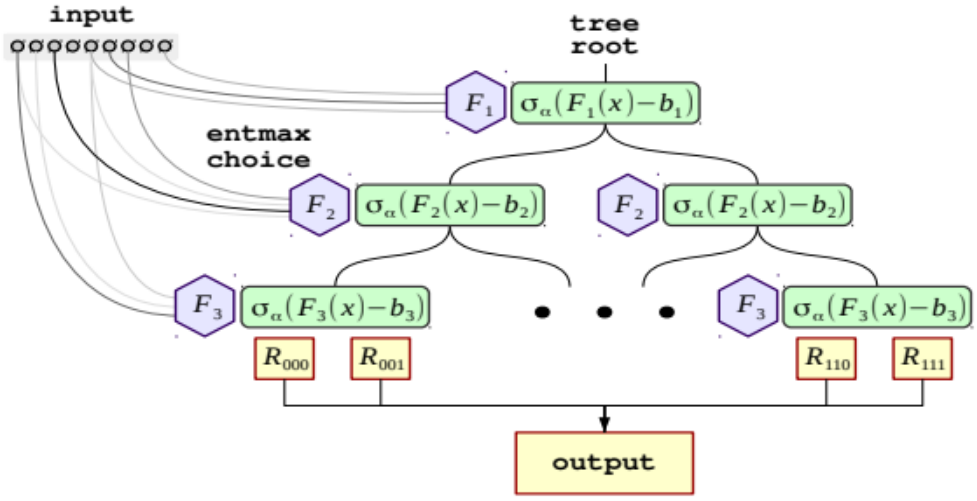

Figure 1: Schematic of single ODT architecture

responses $\mathbb{R}^d$. Denoting $\mathbb{1}(\cdot)$ as the Heaviside function, the tree output is defined as:

$$h(x) = R\Big(\mathbb{1}(f_1(x) - b_1), \cdots, \mathbb{1}(f_d(x) - b_d)\Big). \tag{1}$$

Given full description per ODT, a single NODE layer is composed of m differentiable ODTs of equal depth d, where all m trees get a common vector $x \in \mathbb{R}^n$, containing n numeric features, as shown in Figure 1. Based on equation (1), the current NODE layer approach clearly lacks of differentiability as other tree-based models, which is the key disadvantage compare to the DNNs. In such case, there is no chance to use chain rule to propagate errors, thus back-propagation is no longer possible. To provide NODE layers with full differentiability, the α-`entmax` transformation [11] was introduced to replace the splitting feature choice f_i and the comparison operator $\mathbb{1}(f_i(x) - b_i)$ by their continuous counterparts, as it is able to learn sparse choices, depending only on a few features, via standard gradient descent. The choice function is replaced by a weighted sum of features:

$$\hat{f}_i(x) = \sum_{j=1}^{n} x_j \cdot \texttt{entmax}_a(F_{ij}), \tag{2}$$

where the weights are given as `entmax` over the feature selection matrix $F \in \mathbb{R}^{d \times n}$. Similar approaches were performed by relaxing the Heaviside function $\mathbb{1}$ as a two-class `entmax`, and applying scaling function over features with different characteristic scales based on learnable parameters. The final prediction is shown as:

$$\hat{h}(x) = \sum_{(i_1, \cdots, i_d) \in \{0,1\}^d} R_{i_1 \cdots i_d} \cdot C_{i_1 \cdots i_d}, \tag{3}$$

which is computed as a weighted linear combination of response tensor entries R with weights from the entries of choice tensor C [6]. With m individual ODTs contained per NODE layer, the final output form is given by concatenation of m predictions, $[h^1(x)$,..., $h^m(x)$]. Noted

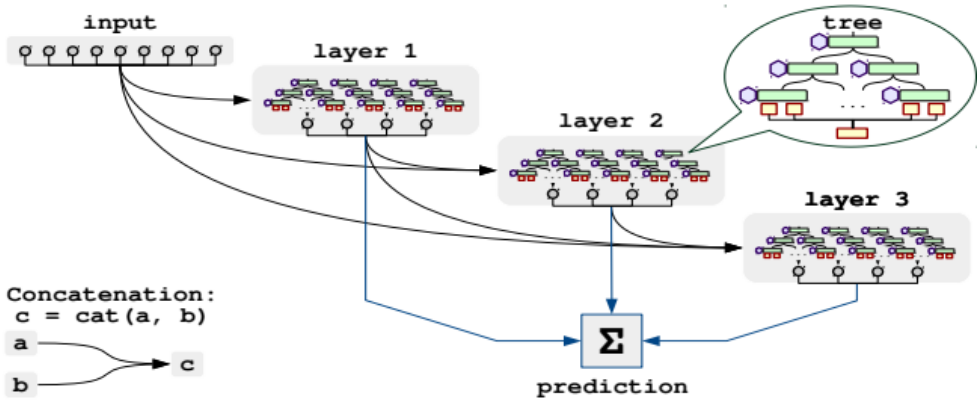

Figure 2: Schematic of NODE architecture

that for classification scenario, $\hat{h}(x)$ need to be clarified as multidimensional tree format with $\hat{h}(x) \in \mathbb{R}^{|C|}$, where $|C|$ is the number of classes. To put every building blocks together, inspired by DenseNet [12] model, the full architecture of NODE model which consisting by a sequence of densely connected NODE layers, where each layer uses a concatenation of all previous layers as its input, as shown in Figure 2. This architecture guarantees NODE preserving the capability of capturing complex dependencies over tabular datasets, as any single tree on ith layer can inherit from chains of up to $(i-1)$th layer outputs as input features. After training the architecture end-to-end via backpropagation, the final prediction is obtained by averaging the outputs of all trees from all the layers, with no constraints on the dimensionality of $\hat{h}(x)$, to be equal to the number of classes. In our particular set up, the total trainable parameters for our NODE model is $234,945$. Lists of hyper-parameters model summaries for all three models are shown in the supplemental files (`DNN_model_full_descriptions.txt`, `NODE_model_full_descriptions.txt` and `XGB_model_full_descriptions.txt`).

2.2 Model Development and Evaluation

As the first step to build up our classification models, the 21948 trainable samples are partitioned with a 90–10 train-test split, from which 19759 samples are used in training and 2189 samples are used in testing. Due to the heavy computation required to build up the XGBoost model, only a single run calculation is performed. While training the baseline DNN and NODE models, early stop and additive learning rate α monitors are used to prevent overfitting in the model training stage. The number of epochs used to train both models is set to be 15 and has tested to achieve convergence. Meanwhile, a stratified k-fold cross validation (CV) scheme is applied to train both DNN and NODE models. Specifically, 3 random states are given to the scheme, and for each random state, a 5-fold CV is performed. Thus, a total of 15 statistically independent runs are executed, and the results are shown as the average from these runs. To evaluate all 3 model performance, the logarithmic loss (log loss) function is utilized and it has the following form:

$$\ell = -\frac{1}{M}\sum_{m=1}^{M}\left(\frac{1}{N}\sum_{n=1}^{N}\left[y_{i,m}(\log \hat{y}_{i,m}) + (1-y_{i,m})\log(1-\hat{y}_{i,m})\right]\right), \tag{4}$$

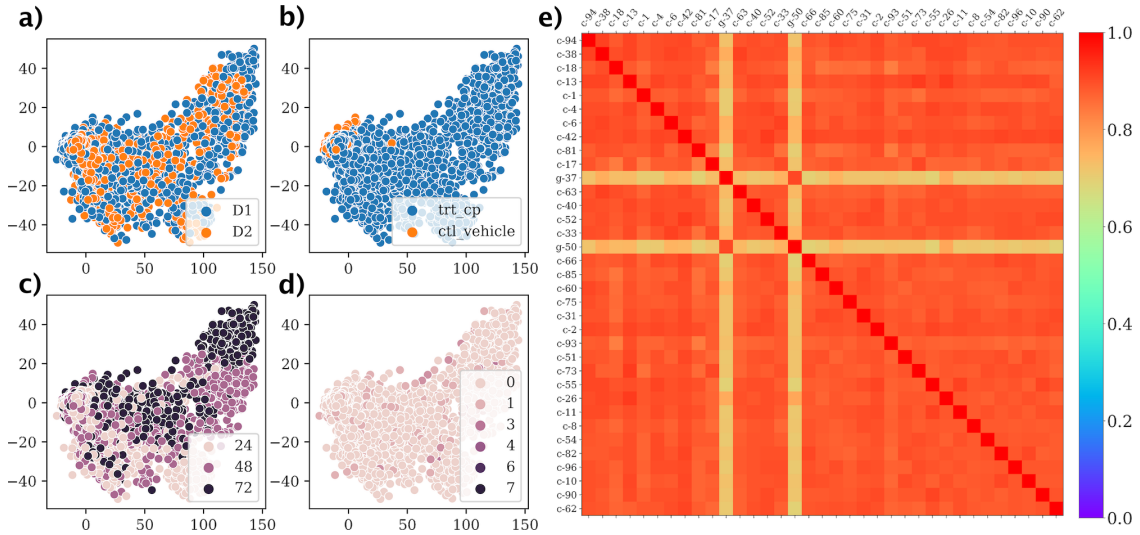

Figure 3: a)-d) illustrate the PCA performed on the training dataset, projected onto a two dimensional space. The colors represent the dose of the compound (a), which can be low, D1, or high, D2 , the type (b), the time (c) and the sum of the number of drugs which the sample responded to. The plot e) shows the correlation matrix between pairs of features with a correlation higher than 0.9.

where M represents the number of scored MoA targets, N is number of predicted targets in the test set. Noticably, for log loss, a 0.001 loss difference is statically meaningful. With our best knowledge, the best Kaggle model performance, when making predictions on the full testing sets, is currently at 0.01799. To further optimize our NODE model, a grid search of hyper-parameters is done. In particular, parameters $n_{\texttt{layer}}$, $n_{\texttt{trees}}$, $n_{\texttt{units}}$ are explored. The detail summary of results for the grid search of NODE model and the model performance comparison between the DNN, XGBoost and NODE will be shown in the result section.

3 Results

3.1 Exploratory Data Analysis

In order to analyze the dataset we performed an exploratory data analysis (EDA) to better qualitatively identify any structure or pattern in the dataset. Figure 3a)-3d) shows the principal component analysis (PCA) projection of the training dataset on a two-dimensional space. We also analyzed the correlation among features in the training dataset. Figure 3e shows a double-entry plot highlighting the couple of features with a correlation value > 0.9. The figure shows how 33 cell viability features and 2 gene expression features satisfy this constrain.

3.2 Grid Search of NODE Parameters

To optimize our NODE model, a grid search of varying $n_{\texttt{layer}}$ & $n_{\texttt{trees}}$, is performed:

Table 1: Grid search for NODE model over parameters $n_{\texttt{layer}}$ & $n_{\texttt{trees}}$, with $n_{\texttt{units}} = 32$

$N_{\texttt{Run}}$	Parameters	Log loss
1	$n_{\texttt{layer}} = 3$, $n_{\texttt{trees}} = 3$	**0.01723**
2	$n_{\texttt{layer}} = 3$, $n_{\texttt{trees}} = 4$	0.01737
3	$n_{\texttt{layer}} = 3$, $n_{\texttt{trees}} = 5$	0.01745
4	$n_{\texttt{layer}} = 4$, $n_{\texttt{trees}} = 3$	0.01743
5	$n_{\texttt{layer}} = 4$, $n_{\texttt{trees}} = 4$	0.01747
6	$n_{\texttt{layer}} = 4$, $n_{\texttt{trees}} = 5$	0.01753
7	$n_{\texttt{layer}} = 5$, $n_{\texttt{trees}} = 3$	0.01757
8	$n_{\texttt{layer}} = 5$, $n_{\texttt{trees}} = 4$	0.01754
9	$n_{\texttt{layer}} = 5$, $n_{\texttt{trees}} = 5$	0.01765

After obtaining the optimal $n_{\texttt{layer}}$ & $n_{\texttt{trees}}$, a separate grid search with respect to the parameter $n_{\texttt{units}}$, which represents number of neurons, is done and the result is summarized:

Table 2: Grid search for NODE model over $n_{\texttt{units}}$, with $n_{\texttt{layer}} = 3$ & $n_{\texttt{trees}} = 3$

parameters	$n_{\texttt{units}} = 32$	$n_{\texttt{units}} = 64$	$n_{\texttt{units}} = 128$
Log loss	**0.01723**	0.01744	0.01749

3.3 Feature Importance from XGBoost and Model Comparison

Since feature importance can be obtained after training a XGBoost classifier, a comprehensive analysis is done by investigating the cumulative sum of feature importance, by sorted feature index. With a threshold of 80% total contributions of feature importance, 805 features are selected and used to explore the model development for NODE with only selected top features. Figure 4 illustrates the cumulative feature importance scores and the averaged log loss as a function of training epochs. After finish training all the models, the summary of their model performance when making predictions on the same testing sets can be tabulated below:

Table 3: Summary of model performance, when making predictions on the same test set.

Method	DNN base line	XGBoost	NODE (80% top features)	NODE (all features)
Log loss	0.0205	0.01850	0.01748	**0.01723**

4 Discussion

From a visual exploration in Figure 3 , it does not emerge any visible structure of the samples by coloring in terms of the dosage or the duration of the treatment apply in a specific assay (see Figure 3a and Figure 3c). From a visual inspection of the PCA colored by type, Figure 3b, it emerges that the compound used as a control, i.e. not responsible for

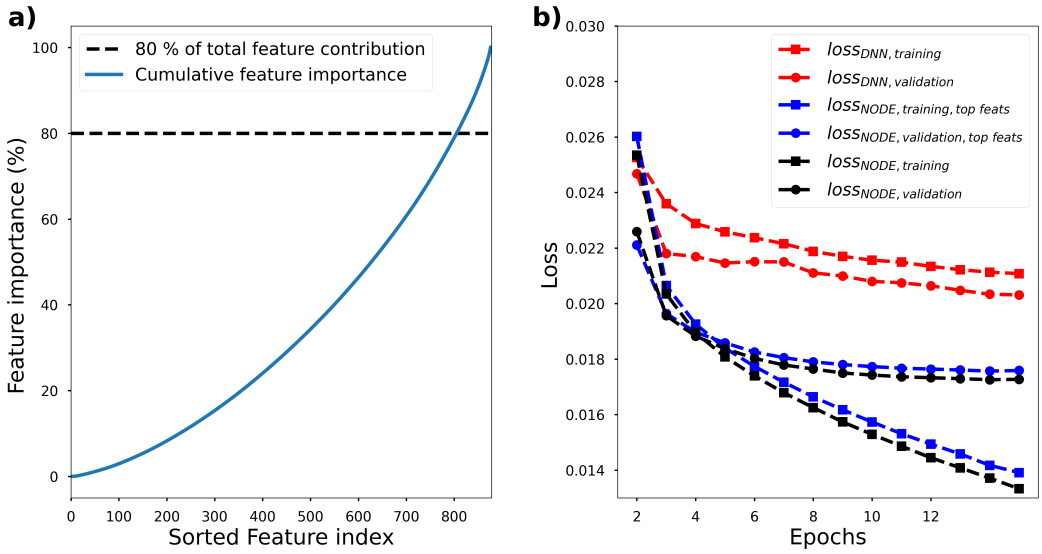

Figure 4: a). Cumulative feature importance scores coming from XGBoost model. b). Model comparison between DNN base line and NODE models with log loss as a unction of number of epochs. Log loss is averaged out of 15 individual runs. To ensure a sensible trace of progression of log loss, the sudden drop between $1^{\mathtt{st}}$ and $2^{\mathtt{nd}}$ epoch has been truncated.

any MoA, form a cluster in the upper-right corner of the plot, hinting that such assays have common characteristics. Finally, Figure 3d, highlight how in a few assays there is a response triggered in several number of drugs, while the majority of the others, have no response or only a single drug response. The initial EDA, as whole, suggests that in this tabular dataset there is an imbalance in the targets, and a small subset of features seem to be responsible for most of the predictive power. Moreover, several features are shown to be highly correlated. Because of this characteristics, this problem is well suited to be tackled not only by using DNN, but also by using decision-trees based model, i.e. XGBoost and NODE, which are expected to perform better on this kind of tabular data.

After the initial EDA, we analyze the performances in the three selected models. From the results shown in Table 3, it can be seen that the DNN baseline has the lowest performance, while the NODE model outperforms the XGBoost model. Diving into the details of the NODE model optimization, we observe that the log losses coming from different NODE architectures are all below the losses from XGBoost and DNN baseline models, proving that NODE model holds the best performance rigorously.

Referring to Table 1 and Table 2, we can see that the optimal NODE architecture occurs at $n_{\mathtt{layer}} = 3, n_{\mathtt{trees}} = 3$ & $n_{\mathtt{units}} = 32$. Meanwhile, from the trend shown in both Tables, a simpler NODE architecture tends to have a better performance. Variations of the log loss are minor during the grid search, and the results are obtained by an an average over 15 independent runs. As an innovative attempt to optimize our NODE model, we extract top features which sums up to 80 % feature importance form the XGBoost model and train a NODE model with only 805 features used (Figure 4a). From Figure 4b, we can observe

that log losses of both training and validation sets for NODE and DNN base line models are reasonably converged. When building the NODE model with top features, the log loss slightly upraises to 0.01748. This trend indicates that the top features can retain reasonably model performance while further reducing the dimension of the model.

As supported by recent literates [3, 6], XGBoost and NODE tend to outperform DNN when dealing with correlated tabular data. Our studies suggest the same trend. As explained through the *manifold hypothesis*, in order to enhance the classification performance, DNNs introduce non-linearity and operate manifold transformations until separations of different discrete classes are achieved. However, for correlated tabulated data, the hyperspaces containing these classes are very likely to be regular or approximated by hyperlanes. Thus, the complex non-linear hyperspace constructed by DNN will doom to have a poor performance when classifying these correlated classes. On the other hand, as shown in the EDA results, the tabular data are correlated so a representative sets of features can contribute the most to the classifications. Decision trees based methods are tailed for such classification task, as they select and combine features very well in this case.

Regarding the performance comparison between XGBoost and NODE, since the NODE architecture generalizes CatBoost [13] to perform gradient boosting on oblivious decision tables, inference becomes very efficient, and the method is quite resistant to overfitting. Moreover, the NODE architecture makes the splitting feature choice and decision tree routing differentiable, which resembles "deep" GBDT and allows constructing multi-layer architectures as well as training end-to-end via mini-batch SGD. From the perspective of inference runtime, as introduced in Method section 2.1.3, the elementary ingredient of a NODE layer are ODTs. One can compute d independent binary splits in parallel and return the appropriate table entry, which makes the inference in ODTs very efficient. Noticeably, different implementations of XGBoost classifer can also lead to different performances. In our study, unlike the classification chain implementations, we implemented XGBoost model as an collection of 206 multivariate XGBoost models. This implementation increases the computational time exponentially. Therefore only single run calculation is done with XGBoost. From the perspective of statically convergence, NODE has a slightly better performance than XGBoost.

5 Conclusion

Although deep neural networks has achieved remarkable advancements in various fields during the past decade, there is no sufficient evidence that deep learning machinery allows constructing methods that outperform gradient boosting decision trees (GBDT), which are often the top choice for tabular problems. In this work, we performed an accurate comparison of DNN, XGBoost, and NODE on the MoA heterogeneous tabular dataset to demonstrate the advantages of the differentiable deep GBDTs architecture over existing competitors with fine-tuned hyperparameters. In particular, the NODE model has achieved a log loss of 0.01723 with all features, 0.01747 with 80% top features, where XGBoost and DNN baseline models have log losses of 0.01850 and 0.0205 accordingly. We suggest future explorations focus on obtaining hierarchical distributed representations [14] learned by tree ensembles on modeling such discrete or tabular datasets, as well as incorporating the NODE layer into

complex pipelines trained via back-propagation, such as multi-modal models.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[2] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[3] Zhi-Hua Zhou and Ji Feng. *Deep Forest*. 2020. arXiv: 1702.08835.

[4] Yongxin Yang, Irene Garcia Morillo, and Timothy M. Hospedales. *Deep Neural Decision Trees*. 2018. arXiv: 1806.06988.

[5] 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*. KDD '16. Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322.

[6] Sergei Popov, Stanislav Morozov, and Artem Babenko. *Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data*. 2019. arXiv: 1909.06312.

[7] Abien Fred Agarap. "Deep learning using rectified linear units (relu)". In: (2018). arXiv: 1803.08375.

[8] Günter Klambauer et al. "Self-normalizing neural networks". In: *Advances in neural information processing systems*. 2017, pp. 971–980.

[9] Yann A LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[10] Timothy Dozat. "Incorporating nesterov momentum into adam". In: (2016).

[11] Ben Peters, Vlad Niculae, and André FT Martins. "Sparse sequence-to-sequence models". In: (2019). arXiv: 1905.05702.

[12] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[13] Liudmila Prokhorenkova et al. "CatBoost: unbiased boosting with categorical features". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018, pp. 6638–6648. URL: https://proceedings.neurips.cc/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

[14] Ji Feng, Yang Yu, and Zhi-Hua Zhou. "Multi-layered gradient boosting decision trees". In: *Advances in neural information processing systems*. 2018, pp. 3551–3561.

Author contributions

All the authors equally contributed on the model development: Exploratory Data Analysis, DNN baseline, XGBoost model, and NODE model, as well as to the report writing.