Why are Tree-Based Methods Effective for Structural Data Modeling: An Exploration

Ran Wang
University of California, Riverside
rwang022@ucr.edu

ABSTRACT

Recently, the success of data science has extraordinarily and deeply affected the tons of areas from academy to industry and is changing the way how to explore, discover and solve problems. But one problem is still unsolved: even though we have more powerful algorithm and highly efficient computing resources, like deep neural network and GPU cluster, to implement the data mining task of structural data, ensemble methods, like random forest, are still the most popular and effective way to solve these tasks. In this paper, we study in the machine learning techniques, especially for decision tree model, for structural data. We explore the performance of several learning methods related to neural networks and decision tree, on the simulated global and local data respectively. Finally, we test our method on Google Merchandise Store customer data set with several other machine learning techniques as benchmarks.

KEYWORDS

Decision Tree, Neural Network, Local and Global Method, Structural Data, Data Clustering

ACM Reference Format:
Ran Wang. 2018. Why are Tree-Based Methods Effective for Structural Data Modeling: An Exploration. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Last ten years witnessed the dramatically development of data science. Thanks to extremely large dataset and high performance super computer clusters, many important researches can be done based on big data For example, the Higgs Boson Machine Learning Challenge, posted via Kaggle data science platform, was organized to promote collaboration between high energy physicists and data scientists and led a new way to solve large scale problem via data mining techniques. At the same time, the data mining techniques, especially the machine learning algorithm, also extraordinarily developed to a higher stage. The deep neural networks, like Convolution Neural Networks or Recurrent Neural Networks, has totally changed the old way to implement the tasks of object instance segmentation in video and real-time multiple languages translations. Self-driving cars, machine translation, recommendation system and Go playing, even hedge funding start to partially or totally depend on deep neural networks and related machine learning techniques.

Although the deep neural networks are so popular among many areas, several research areas related to structural data, like economics, still depends more on "simple" method, like spline, nonparametric kernel regression. Plus, tree-based methods, like random forest and boosting tree, are still the best way to implement the structural data, like predicting consumer's purchase or economic growth. Even in the Higgs Boson Machine Learning Challenge, the most successful methods totally depended on boosting decision tree method. Thus, two important questions are still needed to answer: why does decision tree method outperform other methods in structural data modeling? And why deep neural networks cannot be better than simple tree methods even neural network can extract very complex features from data during training?

In this project, we concentrate in these questions related to decision tree and neural networks. In the first part, we start by introducing many related papers about decision tree, neural networks and its variants. By analyzing several machine learning techniques related to decision tree and neural networks from the papers, we try to distill some possible reasons to the questions above. Next, we design some simulation studies and then test the performances of these learning algorithms to verify our hypothesis. At last, we implement a data science task about Google Merchandise Store customer data.

2 RELATED WORKS

In this section, we start our project by introducing some related researches about learning for structural data. To answer our questions above, we mainly focus on the two groups of methods: decision tree and deep neural networks. We explore many decision tree and neural networks papers related to the theories such that we can analyze the properties of the two groups of methods.

Gradient boosted machines and deep neural nets have dominated recent Kaggle competitions

Competition	Type	Winning ML Library/Algorithm
Liberty Mutual	Regression	**XGBoost**
Caterpillar Tubes	Regression	**Keras** + **XGBoost** + Reg. Forest
Diabetic Retinopathy	Image	SparseConvNet + RF
Avito	CTR	**XGBoost**
Taxi Trajectory 2	Geostats	Classic neural net
Grasp and Lift	EEG	**Keras** + **XGBoost** + other CNN
Otto Group	Classification	Stacked ensemble of 35 models
Facebook IV	Classification	sklearn GBM

Figure 1: Machine learning approaches won the most Kaggle competitions

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Conference'17, July 2017, Washington, DC, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

2.1 Decision Tree and Ensemble Methods

As one of the best choices in many structural data modeling tasks, decision tree, or much precisely, tree-based ensemble methods like random forest and boosting, are very important tool to handle these tasks. In many Kaggle tasks, the predicting result benchmark is mainly based on boosting tree method. Why does the tree-based model work so well in this area? This is the question we want to answer in this research. Basically, there are several papers discussing the connections among decision tree, random forest, KNN and kernel methods. Lin and Jeon in paper [2] showed that the random forest is an adaptive nearest neighborhood method, or more general, a local method. In their paper, they proposed a new KNN method called potential nearest neighbors (k-PNN). Based on PNN, we could analyze the properties of random forest. Then, the authors propose a new splitting scheme which could not only be combined into random forest but applied into other methods. Also, in the paper [3] by Alex Davies and Zoubin Ghahramani two authors considered a new adaptive kernel, random partition kernels, based on random forest. The authors designed two kernels, random forest kernel and fast cluster kernel, and then apply this kernel into other kernel machines like Gaussian Processes, SVM and Kernel-PCA, which largely boosted the performances for those learning algorithms. Both of the papers uncovered and discussed the relationship between decision tree and adaptive kernel methods. In another words, tree-based method a local method similar to kernel and related methods. Thus, a possible reason why do tree-based methods work so well in structural data is that structural data often have many clustering structures that can be capture by local method like decision tree.

2.2 Neural Network

As we discussion in the introduction part, deep neural networks had shown its power in nonlinear function approximating and feature extracting even though the related theories were not clear in several years ago. Recently, there are bunch of papers focusing on the theory behind deep learning. Chiyuan Zhang et al in their paper [4] discussed the generalization of deep learning based on the phenomenon that deep neural networks often generalize very well on testing dataset even though their always be over-parameterized. Zeyuan Zhu et al in the paper [5] explored the convergence theory for over-parameterized neural networks via SGD optimization and showed that the over-parameterized neural networks feature polynomial convergence rate if the networks contain ReLU as activation function and the dataset is non-degenerate, which is a very strong result for optimization problem of deep learning. In paper [6], Pierre Baldi and Roman Vershynin gave the capacity of neural networks which showed that the "power" of a neural network and strongly proved that the multi-layered structure of deep neural networks is a structure-based regularization leading to a better generalization result. To sum up, the success of neural networks is mainly based on two things: over-parametrized networks which can be easily optimized via SGD and multi-layer generalization which can control the over-fitting problem of networks.

Also, the theories about why do neural networks capture very complex structure of data were also be constructed. Article [7] discussed this topic in details. There are two main reasons: distributed representation and combination generalization. Distributed representation means that every neuron does not represent a specific definition by itself but represent a feature and then jointly represent a definition. That is, neural network is a global method, compare to local method like kernel methods and decision tree. Combination generalization is that the flexibility of deep neural network comes from combining previous layers of neurons by next layers of neurons. That could lead to a hierarchical feature extracting process, which is often reasonable for many tasks like visual recognition and language processing. In other words, compare to tree-based model above, deep neural networks are global model with multiple layers. This is the largest difference between tree and neural networks. In our study, we will mainly focus on this difference.

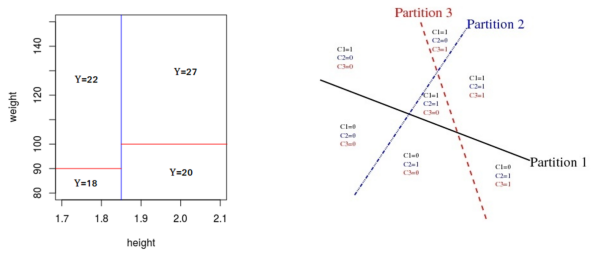

Figure 2: Decision Tree VS Neural Network

2.3 Soft Decision Tree

Based on previous discussion, we find that adaptive local method should work better when we face structural data in research. Now we can make a comparison between tree-based model and neural network model. Basically, neural network is not only an end to end trainable model but a global model like smooth spline. Tree-based model is, on one hand, trained by "growing" based on purity since it is a non-differentiable method. On the other hand, it is an adaptive local method. Thus, if we could combine the end to end training process to tree-based model, it could be expected that we can get a method not only captures local structures in structural data, but all the parameters could work together like the parameters in neural networks.

Basically, there are many research papers related to the "soft" version of decision tree. This first soft decision tree model is called Hierarchical Mixtures of Experts by Michael I. Jordan, Roberts A. Jacobs in the paper [8]. They discussed a version of differentiable decision tree model. Comparing to classic "hard" decision tree, the authors discussed the "soft" decision tree method, which is called hierarchical mixtures of experts (HME).

First, because of its hierarchical structure, this method is a local method same as hard decision tree. Also, since it is a differentiable decision tree, which contains a differentiable activation function in each node, we can train all the parameters via EM or back propagation method. Plus, it is a mixture of experts, which means in each leaf node, we could use not only a constant, but introduce more flexible methods, like linear regression even a simple neural network. The authors also check the performance of HME in different data sets. One possible drawback to soft decision tree method is that the

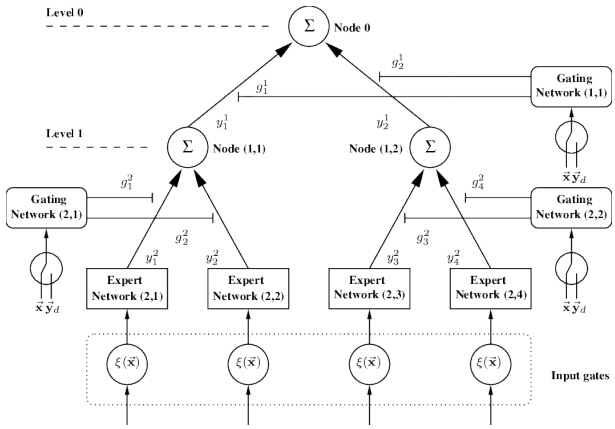

Figure 3: A Two Layer Hierarchical Mixtures of Experts

activation function could lead to a long-time training process, just like sigmoid function in deep neural networks. In our research, we mainly concentrate into this problem and improve soft decision tree with new activation functions. Another research paper about soft decision trees is made by Ozan Irsoy, Olcay Taner, Yildiz, Ethem Alpaydin in paper [9]. To train a hard decision tree, we find the optimal decision stump in each node via purity or related measures. Intuitively, this means we "grow" a tree. In the HME method we discussed above, we first designate the structure of a soft decision tree, then train this tree via EM. In this paper, the authors introduced a new way to grow a soft decision tree. In each node, they used gradient descent to find the optimal splitting line then compare the predicting outcome between two trees with and without the new splitting line to decide that this new node should be added or not. Thus, this method can adaptively learn the structure of soft decision tree but could have a slower training speed.

Basically, the soft decision tree method is not so popular as classic decision tree since the training process is not very fast than grow a decision tree or training a neural network. But in many recent researches, the soft decision tree show its power for learning hierarchical features via CNN features. in paper [12], Peter Kontschieder et al proposed deep neural decision forests that combine the CNN feature extractor and tree structure into one differential hierarchical CNN. In paper [13], Nicholas Frosst and Geoffrey Hinton explored the soft decision tree in distilling the knowledge or features extracted by a neural network based on its hierarchical structure. They found that the soft decision tree method can definitely learn hierarchical features via end to end training. Thus, to our research, a soft version of decision tree which can be trained via gradient based method, will definitely make the comparison between neural networks and decision tree more reasonable.

2.4 Multi-Layered Trees

Recently, deep neural networks show its power in so many areas in terms of its multi layered structure. Similarly, decisions tree could also have multiple layered structure as neural network. In paper [11] Zhi-Hua Zhou and Ji Feng explored Deep Forest by the first time. In this paper, the authors proposed one possible deep learning model without neural networks, which is called gcForest. gcForest mainly contains two parts: the first part is Cascade Forest Structure, which is a multi-layered model with four random forests growing in each layer. In each layer, we train two forests with random feature and two forests with features from subsets of all features. That is, the transformation in each layer is not given by neurons but by forests. Also, the skipping connections, same as the connections from ResNet and DenseNet, are introduced into deep forest. The second part is Multi-Grained Scanning, which is a forest version of convolution layer in spatial or time-dependent way. In the simulation studies, the gcForest method not only performed better than other methods in so many tasks, but it is very robust to different hyperparameters.

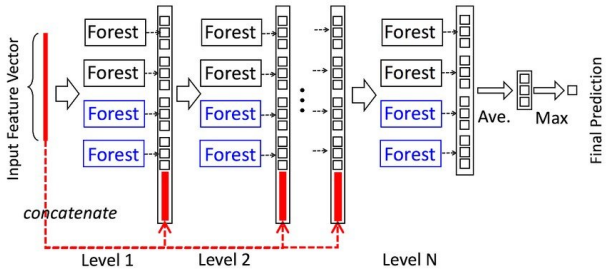

Figure 4: Cascade Forest Structure

3 METHODOLOGY

Based on the exploration of research papers in previous section. The core reason we can get is that structural data often have many clustering structures that can be capture by local method like decision tree but be relatively hard to global method like deep neural network if both of them share similar amount of parameters. Thus, in this section, we mainly focus on two methods: soft decision tree and multi-layered neural networks, and then give learning methods we select for our simulation study.

3.1 Self-Normalized Neural Networks

Since our task is focusing on structural data, it is not reasonable to assume the spatial relationships and time dependency between input variables, which means CNN and RNN could not be selected for implementing our tasks. Thus, a more reasonable choice is fully connected neural networks, FNN. Unfortunately, deep FNN usually suffers from the problem of gradient vanish, even though some techniques, like ReLU and Batch normalization, are introduced to solve this problem. That is the key reason we choose Self-Normalizing Neural Networks as our deep neural network benchmark model for global method.

Self-Normalizing Neural Network or SNN was introduced by Gunter Klambauer, Thomas Unterthiner, Andreas Mayr, Sepp Hochreiter in paper [10] in NIPS 2017. This neural network features a new activation function called Scaled Exponential Linear Unit (SELU), which normalize the output to a distribution with zero in mean and one in variance in each layer. The authors provided an extremely long proof by using fixed point theorem and others to show the properties of SNN: in a deep fully connected neural networks, the output of each layer will converges to a distribution of with 0 in

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}.$$

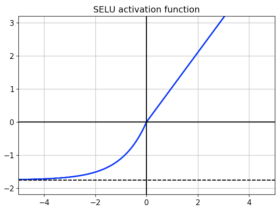

Figure 5: Scaled Exponential Linear Unit

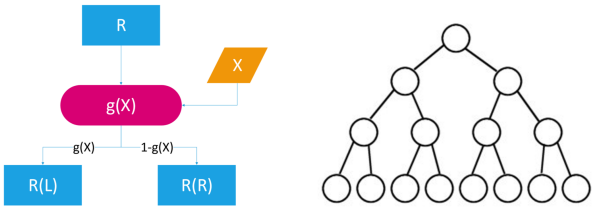

Figure 6: Hierarchical Mixture of Expert

mean and 1 in variance if the networks use SELU as its activation function. Also, they compared their Self-Normalizing Neural Networks with other methods in structural data modeling and found it worked very well. Therefore, in terms of their contribution, we can train a fully connected layer network with a lot of layers in a more efficient way.

3.2 Soft Decision Tree and HME

Basically, as we mentioned before, we select soft decision tree for its local structure and differentiable nodes, which make our experiment more reasonable. In our study, we select the hierarchical mixture of experts by Jordan and Jacob as our learning algorithm since HME cannot only be trained end to end, but, more importantly, features its local structure. Since it is not as popular as deep neural networks, we briefly introduce HME in the following.

Suppose we have a CART decision tree. For each node, there are only two corresponding sub nodes. To a hard version of decision tree, let $g(x)$ be the decision stump function. Thus we have the rule for splitting sample points in each node as:

$$R(X) = \begin{cases} R_{Left}(x) & if\ g(x) > 0 \\ R_{Right}(x) & if\ g(x) < 0 \end{cases}$$

In another words, every sample point will be only separated to one side, left or right. Compare to that, a very straight forwards variant is considering a soft version decision stump for each node:

$$R(x) = g(x) \times R_{Left}(x) + (1 - g(x)) \times R_{Right}(x)$$

where $g(x)$ is a sigmoid gating function $g(x) = \frac{1}{1+e^{-h}}$ and $h = h(x)$. The convex combination between two outputs $R_{Left}(x)$ and $R_{Right}(x)$. Instead of separating each data point to one of the two sides of decision stump like hard decision tree node, soft decision tree node calculate a weighted average value. Intuitively, hard decision tree learns a perceptron in each node, but soft decision tree do logistic regression in each node. Thus, soft decision tree is very similar to recursive neural networks that are neural networks with hierarchical structure.

From figure 5, we can see that, compare to neural network, HME features its hierarchical structure. In each node of the structure on the right, there is a logistic regression structure on the left. Then, in the leaf nodes, not only we can give different constant to each one of them, but choose linear regression even a neural networks. The models in the leaf nodes are called experts.

To calculate the prediction of HME, we first calculate the predictions of all the experts. Then, calculate $g(x)$ in all the intermediate nodes. Finally, according the hierarchical structure, we calculate the weighted averaging values from bottom to top. To train the HME, we can use EM method discussed by Jordan and Jacob, but it is more efficient and simple to choose gradient descent method since HME is differentiable. In our study, we use gradient method to optimize HME.

4 SIMULATION STUDY

In this section, we start to verify the conclusion we discussed before: structural data often have many clustering structures that can be capture by local method like decision tree but hard by neural networks. In the simulation experiment, we design three data generating process. In the first one, we consider a simple nonlinear function corresponding to non-clustering data since all the data distribute evenly. To the second one, we design a data set with several clusters like a Gaussian Mixture Model. The last simulation study combines the first one and second one to give a mixture data distribution.

To the machine learning algorithms, as we mentioned before, we select two methods: Self Normalizing Neural Networks and Hierarchical Mixture of Experts. To SNN, we consider three hidden layers inside. The structure of SNN is:

$$\text{Input} \longrightarrow \text{Hidden1(30)} \longrightarrow \text{Hidden2(30)} \longrightarrow \text{Hidden3(20)} \longrightarrow \text{Output}$$

To HME, we consider a soft decision tree with 16 leaf nodes. Also, each leaf node contains a linear regression as expert. Additionally, we consider a neural networks with one hidden layer as benchmark:

$$\text{Input} \longrightarrow \text{Hidden1(100)} \longrightarrow \text{Output}$$

To make our experiment reasonable, we use the same optimizer, Adam, to optimize three models above. Also, we do not consider regularization term but run gradient descent for 10000 epoch to each model. Finally, we implement our experiment via PyTorch.

4.1 Non-Clustering Data

DGP 1:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + u, u \sim N(0, 1)$$

From Table1 we can see that, to non-clustering data, both NN and SNN cannot only converge much faster than HME and have a lower loss compare to HME. This result means neural networks should perform better than soft decision tree in non-clustering dataset.

	NN(1)	SNN(3)	HME(16)
MSE	1.00000	1.00000	1.08906
Epoch	3200	1600	6200

Table 1: Testing Set MSE Loss on Non-Clustering Data

4.2 Clustering Data

DGP 2:

$$y = exp(-x_1^2 - x_2^2) + exp(-(x_1 - c_1)^2 - (x_2 - c_2)^2)$$

$$+exp(-(x_1-d_1)^2-(x_2-d_2)^2)+exp(-(x_1-e_1)^2-(x_2-e_2)^2)+u, u \sim N(0,1)$$

	NN(1)	SNN(3)	HME(16)
MSE	1.030738	1.058751	1.002711
Epoch	2800	2000	6600

Table 2: Testing Set MSE Loss on Clustering Data

From Table2 we can see that, to clustering data, HME converges to the lowest MSE loss value. SNN and NN cannot perform better than HME. Surprisingly, SNN is worst than NN even though SNN has multiple layer structure but less neurons in each layer. That is also a proof that to clustering data, complex shallow model is better than deep model in some cases.

4.3 Mixture Data

DGP 3:

$$y = DGP1 + DGP2 + u, u \sim N(0,1)$$

	NN(1)	SNN(3)	HME(16)
MSE	1.246045	1.083705	1.003877
Epoch	10000	3200	8800

Table 3: Testing Set MSE Loss on Mixture Data

Lastly, from Table 3 we can see that, to mixture data set, HME has the best performance but worse than in the clustering case. SNN performs similar in the clustering data. But one layer NN converges very slowly and its performance is the worst compare to previous results.

On more thing we need to mention is that the training time of SNN and NN is much shorter than HME in three cases, but HME converges in a more robust way than SNN and NN. To sum up, based on simulation results, our conclusion mentioned before is verified.

5 REAL DATA EXPERIMENT

Based on the discussions and experiments in previous sections, we can see that local method like decision tree is more suitable to clustering data than global method like neural network. In this section, we start to implement a real data experiment to check the conclusion in practice.

The data we use is from Google Analytics Customer Revenue Prediction Challenge on Kaggle. The main object of this challenge is predicting how much GStore customers will spend. In this competition, you're challenged to analyze a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset to predict revenue per customer.

The dataset contains about 58 attributions including such as click time, location, income and so on and 1 target variable, the annual revenue for customer. This is a very huge data set containing 900 thousand data points in training set and 400 thousand in the testing set. Since our object is testing our conclusion about local and global methods but not winning the final prize, we would like to choose part of the big data set (about 30000 for training set and 10000 for testing set) to implement. Similarly, we use the same architectures of NN, SNN and HME in the real data experiment so that we can compare their performances.

5.1 Data Pre-processing and Visualization

Before go to the model training step, we need to concentrate in dataset for pre-processing. By checking the dataset, we find that in 59 attributes, many of them are irrelevant to spending by consumers, like ID. Also, we need to delete many variables only containing one outcome. Finally, we just do some simple feature engineering to make attributes more informative. We do not consider it too much since feature engineering is not the point we want to study in this research. After pre-processing, we have 37 attributes.

To target variable, annual spending, we visualized it into a histogram:

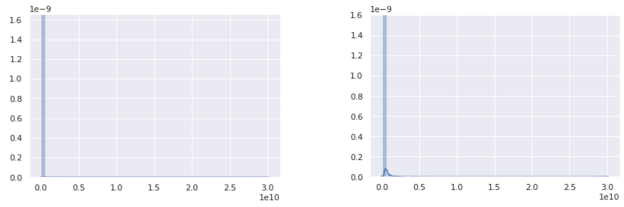

Figure 7: Distributions Before (left) and After (right) Delete NAs

In figure 7, obviously, we find most of them are NA and the distribution of data is right skewed too much even we delete the NAs. Thus, we take log to transform the original data:

In figure 8, after taking log, we can see that the distribution of data is more formal and closed to normal distribution which is better for training models in the followings.

5.2 Results

The detailed results are in the table 4. From the real data experiment, we find several interesting things:

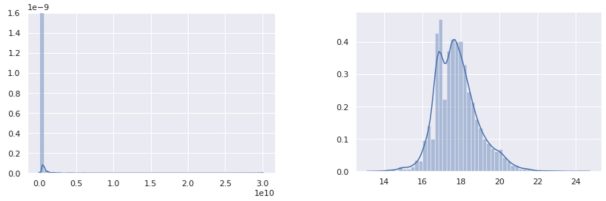

Figure 8: Distributions Before (left) and After (right) Taking Log

	NN(1)	SNN(3)	HME(16)
MSE	1.023264	1.062535	1.126098
Epoch	3200	1000	8800

Table 4: Testing Set MSE Loss on Google Merchandise Store customer data

- First, both one layer neural networks and SNN perform better than soft decision tree method HME. Surprisingly, simple neural networks perform even better than multiple layer neural networks. Thus, we can see that in this real data set, it contains the components for clustering data. And there are many patterns can be handled by simple shallow model and multi-layer trick is not so effective in this case. That is the key point we can get from this experiment.
- Second, both one layer neural networks and SNN converge much faster than HME. SNN converges in just 1000 epoch. Compare to neural networks, HME converges very slowly. Thus, we can see that multi layer trick could boost the convergence in real data experiment.
- Third, we implement neural networks and HME in CUDA. We find that neural network could be speed up significantly, but to HME, the training time before or after using CUDA is exactly at the same level.

To sum up, the real data set contains some components of clustering and also some simple pattern can be catched by shallow model. Also, by using CUDA, neural networks can be trained very efficiently. Conversely, HME suffers from a slow training speed, even though it has similar performance comparing to neural networks, which is a possible reason that HME is not so popular.

6 CONCLUSION AND FUTURE WORKS

In our exploration study, we first discuss the effectiveness of decision tree based model in structural data modeling and also the performance of deep neural networks in many tasks. Then, we propose questions why are deep neural networks that feature the power of approximating not widely used in structural data modeling and why does decision tree perform so well in structural data analysis. Based on a lot of research papers about theories behind deep neural networks and decision tree, we propose a conclusion that the key reason for those questions is the data clustering in structural data in most cases that can be captured easily by local method like decision tree but hard by global method like neural networks. To verify the conclusion, we select two learning algorithms, Self Normalizing Neural Networks for global method and Hierarchical Mixture of Experts for local method, and then analyze their features. In the simulation study part, we implement three different data generating processes including non-clustering data, clustering data and mixture data. We find that global method neural networks are suitable to non-cluster data and local method HME can capture features in clustering data. That is consistent to our conclusion. Finally, in real data experiment, we test the performance of SNN and HME. We find that the real data set contains patterns of clustering data. But compare to HMC, neural networks can be trained very fast.

In the future, based on the main conclusion verified in our research, we can extend the local method, like HME, to a get a better performance on structural data set. On one hand, we can use ensemble methods, like boosting, to improve the performance of HME. Also, like the multi-layered structure in Deep Forest, we can also consider deep structure based on HME. On the other hand, how to speed up the training process is another open question to HME.

REFERENCES

[1] Claire Adam-Bourdarios, Glen Cowan, Cecile Germain, Isabelle Guyon, Balazs Kegl, David Rousseau. 2015. *The Higgs boson machine learning challenge*. JMLR: Workshop and Conference Proceedings 42:19-55.

[2] Yi Lin, Yongho Jeon. 2006. *Random Forests and Adaptive Nearest Neighbors*. Journal of the American Statistical Association Vol. 101, No. 474 (Jun., 2006), pp. 578-590.

[3] Alex Davies, Zoubin Ghahramani. 2014. *The Random Forest Kernel and Creating Other Kernels for Big Data from Random Partitions*. arXiv:1402.4293 [stat.ML].

[4] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals. 2017. *Understanding Deep Learning Requires Rethinking Generalization*. 5th International Conference on Learning Representations.

[5] Zeyuan Allen-Zhu, Yuanzhi Li, Yingyu Liang. 2018. A Convergence Theory for Deep Learning via Over-Parameterization. arXiv:1811.03962 [cs.LG].

[6] Pierre Baldi, Roman Vershynin. 2018. Neuronal Capacity. Thirty-second Conference on Neural Information Processing Systems.

[7] Garrett Hoffman. 2018. *How neural networks learn distributed representations: Deep learning's effectiveness is often attributed to the ability of neural networks to learn rich representations of data.* https://www.oreilly.com/ideas/how-neural-networks-learn-distributed-representations.

[8] Michael I. Jordan, Robert A. Jacobs. 1993. *Hierarchical Mixture of Experts*. Proceedings of 1993 International Joint Conference on Neural Networks.

[9] Ozan Irsoy, Olcay Taner Yildiz, Ethem Alpaydin. 2012. *Soft Decision Trees*. 21st International Conference on Pattern Recognition.

[10] Gunter Klambauer, Thomas Unterthiner, Andreas Mayr, Sepp Hochreiter. 2017. *Self-Normalizing Neural Networks*. 31st Conference on Neural Information Processing Systems.

[11] Zhi-Hua Zhou, Ji Feng. 2017. *Deep Forest*. 26th International Joint Conferences on Artificial Intelligence.

[12] Peter Kontschieder,Madalina Fiterau, Antonio Criminisi,Samuel Rota Bulo. 2016. *Deep Neural Decision Forests*. 25th International Joint Conference on Artificial Intelligence.

[13] Nicholas Frosst, Geoffrey Hinton. 2017. *Distilling a Neural Network Into a Soft Decision Tree*. CEX workshop at AI*IA 2017 conference.