Research Paper Review *06/11/2024*

Neural Oblivious Decision Ensembles (NODE) For Deep Learning On Tabular Data

*Submitted as part of the **CSCI E-82** course offered by **Harvard Extension School**.*

By: Daud Waqas

Abstract:

Nowadays, deep neural networks (DNNs) have become the main instrument for machine learning tasks within a wide range of domains, including vision, NLP, and speech. Meanwhile, in an important case of heterogenous tabular data, the advantage of DNNs over shallow counterparts remains questionable. In particular, there is no sufficient evidence that deep learning machinery allows constructing methods that outperform gradient boosting decision trees (GBDT), which are of- ten the top choice for tabular problems. In this paper, we introduce Neural Oblivious Decision Ensembles (NODE), a new deep learning architecture, designed to work with any tabular data. In a nutshell, the proposed NODE architecture generalizes ensembles of oblivious decision trees, but benefits from both end-to-end gradient-based optimization and the power of multi-layer hierarchical representation learning. With an extensive experimental comparison to the leading GBDT packages on a large number of tabular datasets, we demonstrate the advantage of the proposed NODE architecture, which outperforms the competitors on most of the tasks. We open-source the PyTorch implementation of NODE and believe that it will become a universal framework for machine learning on tabular data.

Link: https://paperswithcode.com/method/node

Context: The reason I chose this paper was to see if neural networks did have a place when it comes to dealing with tabular data; in HW3, I actually skipped doing neural networks after FNNs since I determined that it would ultimately be too time-costly versus the gradient boosting models which were already performing quite well.

Extension *(from last paragraph onwards)*:

The NODE architecture represents a notable step forward in applying deep learning to tabular data, introducing *back-propagation* and *differentiable oblivious decision trees (ODTs)* to better capture *complex hierarchical relationships*. NODE's ability to form distinct hierarchal layers with back-propagation means that it is able to piece together complex relationships within datasets that have a lack of geometric relationships between features; this made NODE incredibly effective versus Gradient Boosting Decision Trees (GBDT) models like XGBoost for datasets such as Epsilon and YearPrediction, both of which were high-dimensional, complex datasets with loosely defined feature relationships; this advantage is established through NODE's utilisation of densely connected "NODE layers", where each layer contains several "trees" whose outputs are then linked and averaged for the final output between all the layers *(refer to Figure 2)*. However, NODE's design *(while powerful for high-dimensional datasets which have a lack of explainability between features)* does reveal limitations in scenarios where more precise, interpretable feature splits are of more importance. On ranking datasets like Microsoft and Yahoo, which contain interpretable, structured feature relationships common in ranking tasks, commonly used GBDT models *(ie. XGBoost and CatBoost covered in this study)* did end up outperformed NODE. This discrepancy suggests that NODE's reliance on ODTs might impose a "restrictive inductive" bias, limiting its ability to adapt to feature-rich datasets that benefit from simplified, tailored, non-uniform splits. Additionally, NODE's design incurs *longer training and inference times* compared to traditional GBDTs. While GBDTs like CatBoost and XGBoost are optimised for fast inference and compute times on CPUs *(with even faster compute on GPUs, which both XGBoost and CatBoost now natively support with NVIDIA CUDA)*, NODE requires significantly longer compute times even on GPUs. This is primarily due to the multi-layer configurations, the complexity of its layered structure, the reliance on the \`entmax\` function in tree routing, and the added back-propagation between the layers. In practical applications, this increased compute cost can limit NODE's appeal, particularly for time-sensitive tasks or applications with constrained computational resources. It is also important for data scientists to note that, since NODE is based on neural networks, it also comes with many of the advantages and disadvantages that neural networks are often associated with. This includes the advantage of a centralised deep learning framework *(in NODE's case, PyTorch)* alongside compatibility with many types of devices supported by PyTorch *(ie. unlike XGBoost and CatBoost which only natively support NVIDIA CUDA, PyTorch also comes with AMD ROCm and Metal Shaders support)*, and the disadvantage of lack of explainability, which could be a critical issue depending on the criticality of use case *(ie. financial and medical institutions would often want reasoning behind why a certain decision or outcome was reached)*. Based on the observations from this study, it is evident that neural networks hold significant potential for analysing tabular data. However, given the current computational demands, their advantages are most pronounced for specific types of datasets where *high-dimensional complexity and lack of feature explainability* outweighs the costs associated with increased compute requirements.

NOTE: The results for classification datasets were given in classification error (lower is better), and the results for regression datasets were given in mean squared error (lower is better).

Potential Improvements *(how I would take this study forward)*:

To extend NODE's applicability, I would try to explore integrating non-oblivious trees or adaptive decision structures, allowing the model to selectively apply diverse split types based on dataset characteristics. Such adaptations would improve NODE's flexibility and make it more suitable for datasets with clearly defined feature relationships. Not only that, but it may also mitigate one of the core caveats of NODE based on this study, and get rid of the issue associated with the "restrictive inductive" bias. *Please note that the authors of this study did mention this to be a subject of future research in the paper, where it was quoted "this implies that NODE should be extended to non-oblivious trees, which we leave for future work"*. It may also be worth exploring options to reduce computational load; this would include *developing dedicated GPU kernels (incl. batch inference optimisation)* and *using an alternative to \`entmax\`*. By developing dedicated GPU kernels, including the \`entmax\` function *(or something similar to it)* and ODT-based decision structures, we could theoretically allow for NODE to take full advantage of GPU parallelisation. Note here that GPU support has already been implemented through the PyTorch platform, but more tailored GPU operations are still possible for the sake of compute reduction *(even a 10-15% improvement would be substantial)*. It may also help to evaluate lighter alternatives to \`entmax\`, such as \`sparsemax\`, for the sake of reducing compute as well. The only other notable issue is the lack of explainability, though that is more of a global issue throughout neural networks and deep learning, and isn't really related to this study in particular.