This is definitely not all you need

m-clark.github.io/posts/2021-07-15-dl-for-tabular

Author Michael Clark Published July 19, 2021 Citation Clark, 2021 July 19, 2021

Motivation

I've been a little perplexed at the lack of attention of deep learning (DL) toward what I consider to be 'default' data in my world, often referred to as *tabular data*, where typically we have a two dimensional input of observations (rows) and features (columns) and inputs are of varying type, scale and source. Despite the ubiquity of such data in data science generally, and despite momentous advances in areas like computer vision and natural language processing, at this time, it's not very clear what the status of DL for tabular data is.

There have been developments in the area recently though, with some modeling approaches, such as TabNet, gaining traction. In June of 2021, I actually came across three papers on Arxiv that were making related claims about the efficacy of DL for tabular data. As in many academic and practical (and life) pursuits, results of these studies are nuanced, so I thought I'd help myself and others by summarizing here.

Goal

I want to know if, e.g. time and/or resources are limited, whether it will be worth diving into a DL model if I have a satisfactory simpler/easier one ready to implement that does pretty well. Perhaps this answer is already, 'if it ain't broke, don't fix it', but given the advancements in other data domains, it would be good to assess what the current state of DL with tabular data is.

Caveats

- I'm not going to do more than give a cursory summary of the articles, and provide no in-depth explanation of the models. For more detail, see the corresponding articles and references for the models therein. You are not going to learn how to use TabNet, NODE, transformers, etc., for tabular data.
- There are other decent articles on the topic not covered here. Some are referenced in these more recent offerings, so feel free to peruse.

Quick Take

In case you don't want any detail, here's a quick summary based on my impressions from these articles. Right now, if you want to use DL on tabular data, don't make a fuss of it. A simple architecture, even a standard multi-layer perceptron, will likely do as well as more complicated ones. In general though, the amount of effort put into prep/tuning may not be worth it for many typical tabular data settings, for example, relative to a suitably flexible statistical model (e.g. GAMM) or a default fast boosting implementation like

XGBoost. However, DL models are already thinking ‘big data’, so for very large data situations, a DL model might make a great choice, as others may not be computationally very viable. It also will not be surprising at all that in the near future some big hurdle may be overcome as we saw with DL applications in other fields, in which case some form of DL may be ‘all you need’.

Now, on to the rest!

Tabular Data: Deep Learning is Not All You Need

Paper Info

- *Who*: Shwartz-Ziv & Armon
- *Where*: Intel
- *When*: 2021-06-21 V1
- Arxiv Link

From the Abstract

> We analyze the deep models proposed in four recent papers across eleven datasets, nine of which were used in these papers, to answer these questions. We show that in most cases, each model performs best on the datasets used in its respective paper but significantly worse on other datasets. Moreover, our study shows that XGBoost (Chen and Guestrin, 2016) usually outperforms the deep models on these datasets. Furthermore, we demonstrate that the hyperparameter search process was much shorter for XGBoost.

Overview

For each model they used the data that was implemented in the original model papers by the authors (e.g. the dataset used in the TabNet article), and also used their suggested parameter settings. They tested all the models against their own data, plus the other papers’ data, plus two additional data sets that were not used in any of the original papers.

Data

They use eleven total datasets. Nine datasets are those used in the original papers on TabNet, DNF-Net, and NODE, drawing three datasets from each paper. Additionally, Shwartz-Ziv & Armon use two Kaggle datasets not used in any of those papers. Sample sizes ranged from 7k to 1M, 10-2000 features, with two being numeric targets, while the other target variables ranged from 2-7 classes. Datasets are described in detail in the paper along with links to the source (all publicly available).

Models Explored

Brief summaries of the DL models are found in the paper.

- *XGBoost*
- *TabNet*

- *Neural Oblivious Decision Ensembles* (NODE)
- *DNF-Net*
- *1D-CNN*

Quick Summary

Not counting the ensemble methods…

- TabNet did best on all of its own data sets, but was not the best model on any other.
- NODE each did best on 2 of its own 3 data sets, but not on any other.
- DNF-Net best on one of its own 3 data sets, but not on any other.
- XGBoost was best on the remaining 5 datasets.

Counting the ensemble methods…

- TabNet did best on 2 of its own 3 data sets, but was not the best model on any other.
- DNF-Net and NODE each did best on one of its own 3 data sets, but not on any other.
- XGBoost was best on one dataset.

Of those, XGB was notably better on 'unseen' data, and comparable to the best performing ensemble. A simple ensemble was also very performant. From the paper:

> The ensemble of all the models was the best model with 2.32% average relative increase, XGBoost was the second best with 3.4%, 1D-CNN had 7.5%, TabNet had 10.5%, DNF-Net had 11.8% and NODE had 14.2% (see Tables 2 and 3 in the appendix for full results).

As a side note, XGBoost + DL was best, but that defeats the purpose in my opinion. Presumably any notably more complicated setting will be potentially better with enough complexity, but unless there is an obvious way on how to add such complexity, it's mostly an academic exercise. However, as the authors note, if search is automated, maybe the complexity of combining the models is less of an issue.

Other stuff

Kudos

The authors cite the No Free Lunch theorem in the second paragraph, something that appears to be lost on many (most?) of these types of papers touting small increases in performance for some given modeling approach.

Issues

There are always things like training process/settings that are difficult to fully replicate. By the time authors publish any paper, unless exact records are kept, the iterations (including discussions that rule out various paths) are largely lost to time. This isn't a knock on this paper, just something to keep in mind.

Opinion

I liked this one in general. They start by giving the competing models their best chance with their own settings and data, which was processed and trained in the same way. Even then, those models still either didn't perform best, and/or performed relatively poorly on any other dataset.

Regularization is all you Need: Simple Neural Nets can Excel on Tabular Data

Paper Info

- *Who*: Kadra et al.
- *Where*: U of Freiburg, Leibniz U (Germany)
- *When*: 2021-06-06 V1
- Arxiv Link

From the Abstract

> Tabular datasets are the last “unconquered castle” for deep learning… In this paper, we hypothesize that the key to boosting the performance of neural networks lies in rethinking the joint and simultaneous application of a large set of modern regularization techniques. As a result, we propose regularizing plain Multilayer Perceptron (MLP) networks by searching for the optimal combination/cocktail of 13 regularization techniques for each dataset using a joint optimization over the decision on which regularizers to apply and their subsidiary hyperparameters.

> We empirically assess the impact of these regularization cocktails for MLPs on a large-scale empirical study comprising 40 tabular datasets and demonstrate that (i) well-regularized plain MLPs significantly outperform recent state-of-the-art specialized neural network architectures, and (ii) they even outperform strong traditional ML methods, such as XGBoost.

> We emphasize that some of these publications claim to outperform Gradient Boosted Decision Trees (GDBT) [1, 37], and other papers explicitly stress that their neural networks do not outperform GBDT on tabular datasets [38, 22]. In contrast, we do not propose a new kind of neural architecture, but a novel paradigm for learning a combination of regularization methods.**

Overview

This data is more about exploring regularization techniques (e.g. data augmentation, model averaging via dropout) rather than suggesting any particular model is superior. Even in the second paragraph they state their results do not suggest a performance gain over boosting methods. Their focus is on potentially improving DL for tabular data through regularization with two hypotheses:

- Regularization cocktails outperform state-of-the-art deep learning architectures on tabular datasets.
- Regularization cocktails outperform Gradient-Boosted Decision Trees, as the most commonly used traditional ML method for tabular data.

Data

Forty total datasets ranging from as little as ~400 observations to over 400k, and between 4 and 2000 features. All were categorical targets, with about half binary. All available at openml.org with target ID provided.

Models Explored

Comparison models:

- *TabNet*: (with author's proposed defaults)
- *NODE*: (with author's proposed defaults)
- *Autogluon*: Tabular: can use other techniques but restricted to ensembles of neural nets for this demo
- *ASK-GBDT*: GB via Auto-sklearn (Note this tool comes from one of the authors)
- *XGBoost*: Original implementation
- *MLP*: Multilayer Perceptron - 9 layers with 512 hidden units each.
- *MLP+D*: MLP with Dropout
- *MLP+C*: MLP with regularization cocktail

Quick Summary

- To begin, their regularization cocktail approach is the clear winner on these datasets, having one outright on over 40% of them (based on table 2).
- Standard XGB performed best (or tied for best) 8 of the 40 data sets, while it or ASK-GBDT were best for 12 datasets combined.
- Simple MLP was best once, while MLP with dropout was best 5 times, while the cocktail method was best in general, across 19 datasets.
- The 'fancy' DL models were the worst performers across the board. TabNet never performed best, and NODE only did once, but the latter also repeatedly failed due to memory issues or run-time limitations (this memory issue was mentioned in the previous paper also).
- Head-to-head, the cocktail beat the standard XGB 26 out of 38 times with three ties. So it wins 65% of the time against XGB, 70% against ASK-GBDT, but 60% against either (i.e. some XGB approach).

Other Stuff

Kudos

- Recognize that tabular data is understudied in mainstream DL literature
- They used a lot of datasets
- They look at the simplest DL models for comparison

Issues

- I wonder why there was not a single numeric outcome among so many datasets. Furthermore, some of the data are image classification (e.g. Fashion-MNIST), so I'm not sure why they're included. I wouldn't use a 'tabular' technique when standard computer vision approaches already work so well.

- I'm not familiar with the augmentation techniques they mention, which were devised for image classification, but there have been some used for tabular data for a couple decades at this point that were not mentioned, including simple upsampling, or imputation methods (e.g. SMOTE). That's not a beef with the article at all, I've long wondered why people haven't been using data augmentation for tabular data given it's success elsewhere (including for tabular data!).

- They use a standard t-tests of ranks, but if we're going to use this sort of approach, we'd maybe want to adjust for all the tests done, and probably for all pairwise comparisons (they show such a table for the regularization methods). Depending on the approach and cutoff, the XGB vs. Cocktail difference may not be significant.

- Also, I couldn't duplicate these p-values with R's default settings for Wilcoxon signed rank tests, and there does in fact seem to be inconsistency between the detailed results and Wilcoxon summaries. For example, in the regularization tests of Table 9, `Cocktail` vs. `WD` and `DO` shows two ties in the first four data sets, yet only 1 tie is reported in the comparison chart for both (Figure 4). For the models, Table 2 show 3 ties of `XGB` & the `Cocktail`, with 1 for `ASK-G` and `Cocktail`, but 2 and 0 are reported for their Wilcoxon tests. It's not clear what they did for NODE with all the NAs. I do not believe these discrepancies, nor adjusting for multiple comparisons, will change the results (I re-did those myself).

Opinion

If we ignore the regularization focus and just look at the model comparisons, I'm not overly convinced we have a straightforward victory for cocktail vs. GB as implied in the conclusion. Results appear to be in favor of their proposed method, but not enough to be a near-guarantee in a particular setting, so we're back to square one of just using the easier/faster/better tool. I'm also not sure who was questioning the use of regularization for neural networks or modeling in general, so the comparison to any model without some form of regularization isn't as interesting to me. What is interesting to me is that we have another round of evidence that the fancier DL models like TabNet do not perform that well relative to GB or simpler DL architectures.

Revisiting Deep Learning Models for Tabular Data

Paper Info

- *Who*: Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, Artem Babenko
- *Where*: Yandex (Russia)
- *When*: 2021-06-22
- Arxiv Link
- Source code

From the Abstract

> The necessity of deep learning for tabular data is still an unanswered question addressed by a large number of research efforts. The recent literature on tabular DL proposes several deep architectures reported to be superior to traditional "shallow" models like Gradient Boosted Decision Trees. However, since existing works often use different benchmarks and tuning protocols, it is unclear if the proposed models universally outperform GBDT. Moreover, the models are often not compared to each other, therefore, it is challenging to identify the best deep model for practitioners.

> In this work, we start from a thorough review of the main families of DL models recently developed for tabular data. We carefully tune and evaluate them on a wide range of datasets and reveal two significant findings. First, we show that the choice between GBDT and DL models highly depends on data and there is still no universally superior solution. Second, we demonstrate that a simple ResNet-like architecture is a surprisingly effective baseline, which outperforms most of the sophisticated models from the DL literature. Finally, we design a simple adaptation of the Transformer architecture for tabular data that becomes a new strong DL baseline and reduces the gap between GBDT and DL models on datasets where GBDT dominates.

Overview

This paper compares different models on a variety of datasets. They are interested in the GB vs. DL debate, but like the previous paper, also interested in how well a simpler DL architecture might perform, and what steps might help the more complicated ones do better.

Data

They have 11 datasets with a mix of binary, multiclass and numeric targets. Sizes range from 20K to 1M+. There appears to be some overlap with the first paper (e.g. Higgs, Cover type).

Models Explored

'Baselines'

- *XGBoost*
- *CatBoost*
- *MLP*
- *ResNet*

DL Comparisons

- *SNN*
- *NODE*
- *TabNet*
- *GrowNet*
- *DCN V2*
- *AutoInt*

In addition, they look at ensembles of these models, but this is not of interest to me for this post.

Quick Summary

Note that these refer to the 'single model' results, not the results for ensembles.

- Some form of boosting performed best on 4 of the 11 datasets.
- ResNet was best on four classification tasks, but not once for numeric targets.
- At this point you won't be surprised at what doesn't perform as well- TabNet, NODE, and similar. TabNet, DCN, and GrowNet were never the best performer, and the other three were best one time a piece.
- MLP did not perform best on any data, however the authors note that it 'is often on par or even better than some of the recently proposed DL models'.
- They also looked at models with a 'simple' transformer architecture. Their results suggest better performance than the other DL models, and similar performance to ResNet.

Other Stuff

Kudos

- Sharing the source code!
- Recognizing that results at this point are complex at best given the lack of standard datasets

Issues

> They note a distinction between *heterogeneous* vs. other types of data. They call data heterogeneous if the predictors are of mixed data types (e.g. categorical, numeric, count), while something like pixel data would be *homogeneous* because all the columns are essentially the same type. The latter isn't as interesting to me for this sort enterprise, and I think the former is what most are thinking about for 'tabular' data, otherwise we'd just call it what it is (e.g. image or text data), and modeling/estimation is generally quite a bit easier when all the data is the same type. I do think it's important that they point out that GB is better with heterogeneous data, and I think if you only look at such data, you'd likely see GB methods still outperforming or at worst on par with the best DL methods.

Opinion

These results seem consistent with others at this point. Complex DL isn't helping, and simpler architectures, even standard MLP show good performance. In the end, we still don't have any clear winner over GB methods.

Overall Assessment

These papers put together are helpful in painting a picture of where we are at present with deep learning for tabular data, especially with mixed data types. In this setting, it seems that more complicated DL models do not seem to have any obvious gain over simpler architectures, which themselves do not consistently beat boosting methods. It may also be the case that for data of mixed data types/sources, boosting is still the standard to beat.

Even though these articles are geared toward comparisons to GB/XGBoost, in several settings I've applied them, I typically do not necessarily have appreciably greater success compared to a default setting random forest (e.g. from the ranger package in R), or sufficiently flexible statistical model. Unfortunately this comparison is lacking from the papers, and would have been nice to have, especially for smaller data settings where such models are still very viable. I think a viable fast model, preferably one without any tuning required (or which simply is taken off the shelf) should be the baseline.

In that light, for tabular data I think one should maybe start with a baseline of a penalized regression with appropriate interactions (e.g. ridge/lasso), or a more flexible penalized approach (GAMM) as a baseline, the latter especially, as it can at least automatically incorporate nonlinear relationships, and tools like mgcv or gpboost in R can do so with very large data (1 million +) in a matter of seconds. In settings of relatively higher dimensions, interactions and nonlinearities should be prevalent enough such that basis function, tree, and DL models should be superior. Whether they are practically so is the key concern even in those settings. With smaller, noisier data of less dimension, I suspect the tuning/time effort with present day DL models for tabular data will likely not be worth it. This may change very soon however, so such an assumption should be regularly checked.

last updated: 2022-05-02

Neural Net image source from UC Business Analytics R Programming Guide