Tree-based models: quicker learning time than deep learning models, since NN struggle learn irregular patterns, and rotation invariance lower their performance.

2. background

Deep learning for tabular data: data encoding techniques to make it more suitable for DL, hybrid methods (keep flexibility of NN and inductive bias of tree-based model), or factorization machines and tabular-specific transformers architecture.

Paper focus on transformers & multi-layer-perceptron (MLP)

Disadvantages of NN vs tree-based models: inflexible, small no. of datasets, biased towards author’s model.

No std benchmark for tab. data

Why TBM may outperform NN: MLPs are expressive enough but may lack proper regularization.

3. benchmarking

Heterogeneous columns, not high-dimensional, remove undocumented datasets, remove stream-like data, use real-world data, not small dataset, not too easy, not deterministic

Side issues: truncate to 10000 samples, remove missing data, use binary classification, remove categorical columns with >20 values, remove numerical features with <10 features

Hyperparameter tuning 🡪 uncontrolled variance on benchmark 🡪 do random searches for ~400 iterations per datasets

Data prep: Gaussianise NN training features with sklearn’s QuantileTransformer, Transform regression targets then transform it back for eval with sklearn’s TransformedTargetRegressor and QuantileTransformer, OneHotEncoder for models not handling categorical variables natively

4. TBM > Deep learning on tab data

Use 3 models for TBM: sklearn’s randomforest, gradientboostingtrees, xgboost

Benchmark models: MLP (use pytorch’s ReduceOnPlateau learning rate scheduler), Resnet

5. Why is it better

Best methods on tab data: ensemble methods, bagging (random forest), or boosting (xgboost, Gbts)