

NeuroTree

A differentiable tree operator
for tabular data



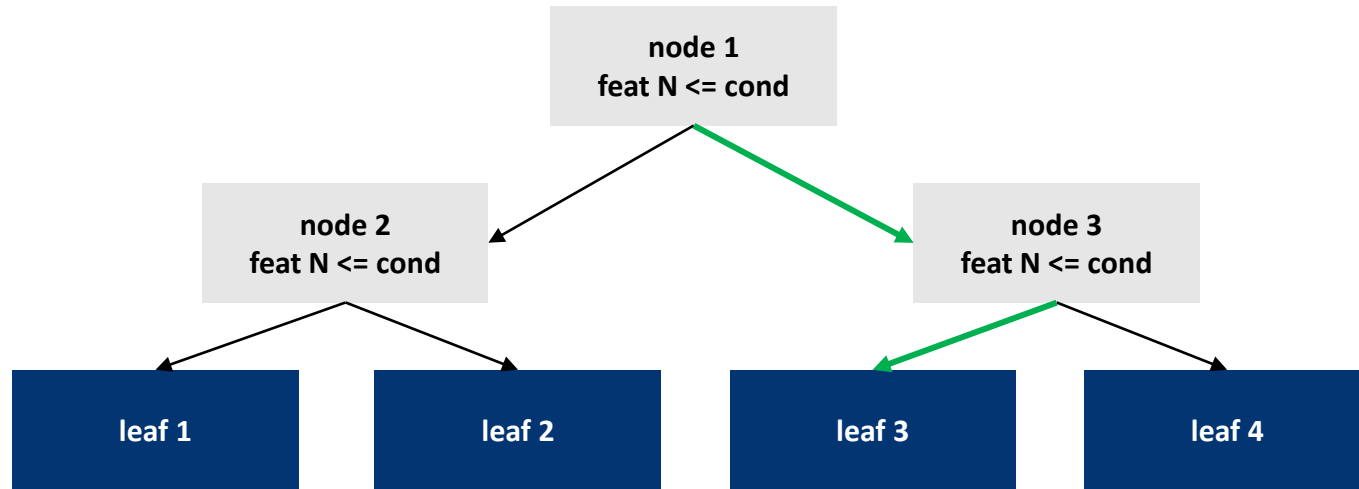
Jérémie Desgagné-Bouchard, FCAS
Head of Science - Evovest



Motivation

- Tabular data is the most commonly encountered form of data in many fields of practice, incl. finance and insurance
- Boosted tree based algos (XGBoost, CatBoost, LightGBM, EvoTrees.jl) have been top performing models for almost a decade
 - *Why do tree-based models still outperform deep learning on tabular data?* <https://arxiv.org/pdf/2207.08815>
- **Tackle limitations of trees:**
 - Myopic view during tree construction
 - Hard splits: a sub efficient way to represent linear or smooth features to predictions relationships
- **Opportunity for signal diversification**

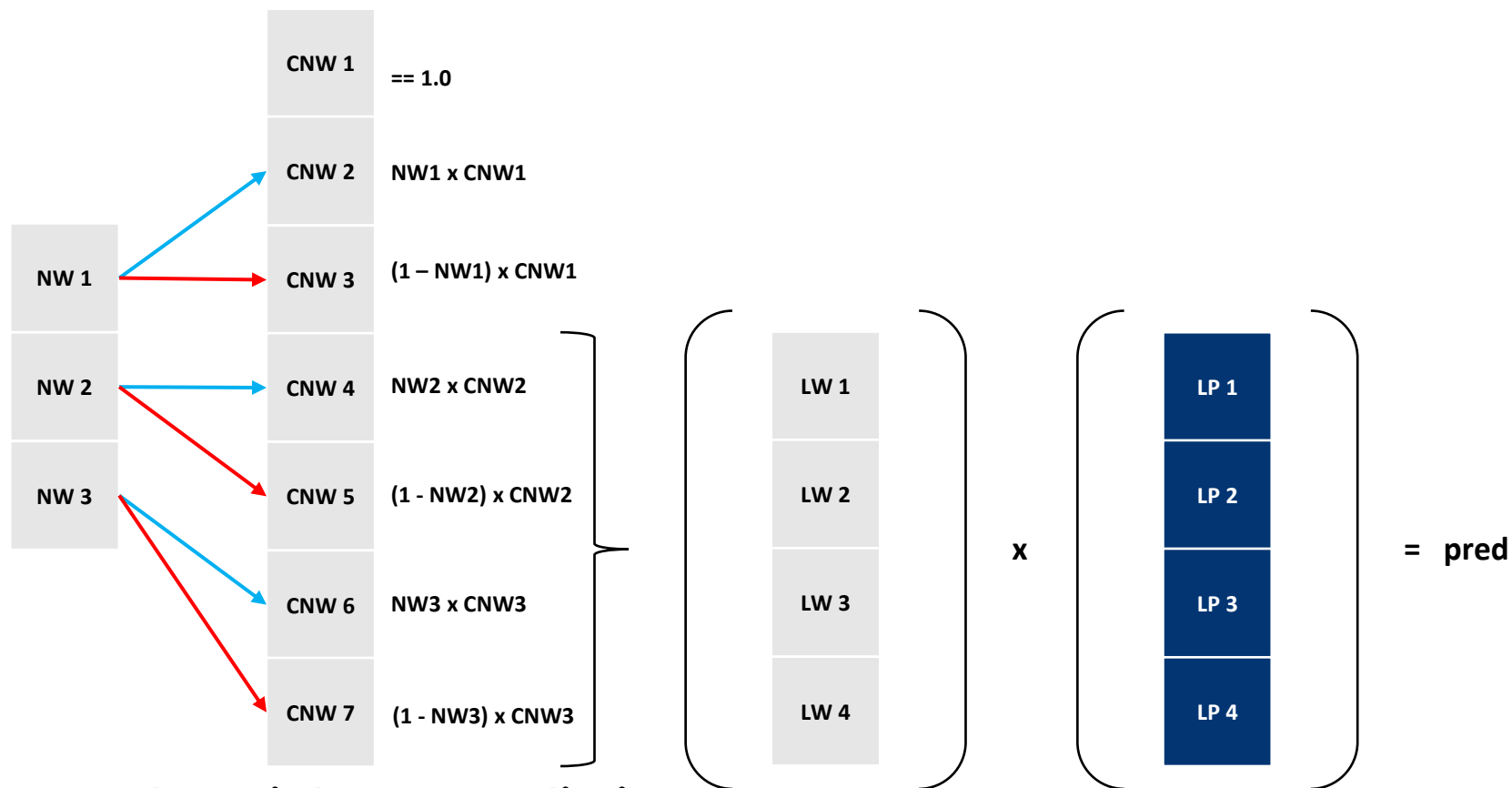
Trees Basics



Different perspectives on the nature of the task performed by trees:

1. Associate an observation to a leaf index based on its features using « depth » binary conditions
2. Associate an observation to a probability of belonging to each leaf, by using « depth » binary conditions

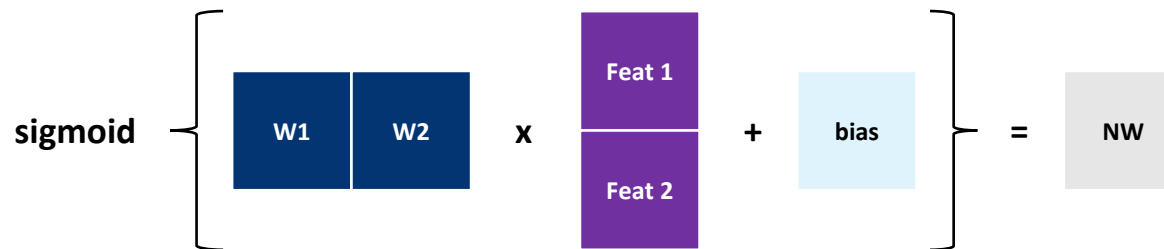
Single Tree Overview



From node weights to prediction

1. Cumulate the node weights. The latest 2^{depth} nodes are the leaf weights.
2. The dot product of the leaf weight and leaf predictions results in the tree prediction.
 - A single leaf prediction is a vector of values.
 - For classification, a vector of length K , where K is the number of classes
 - For a regression task, a vector of length 1: single regression estimate

Soft Node Weights



Differentiable node weights are derived by emulating the two-steps procedure found in classical binary trees:

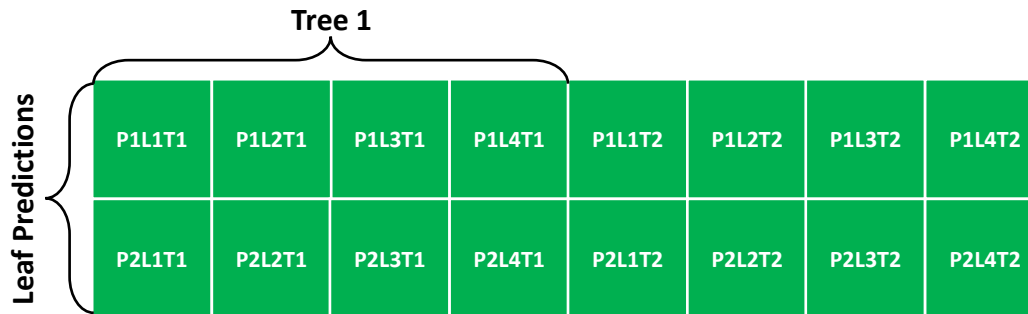
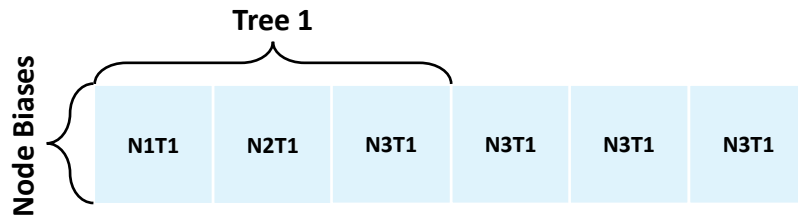
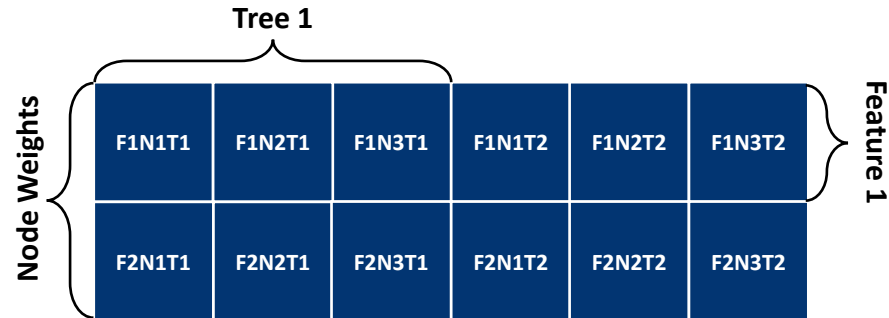
1. Select a feature

- dot product of (normalized) features with learnable weights
- Results in a linear reprojection of the features, not a hard single feature selection which wouldn't be differentiable

2. Select a threshold for that feature

- Apply a sigmoid activation following the addition of a bias to the above feature reprojection

NeuroTree Structure



Computing Node and Leaf Weights

```
function leaf_weights!(nw)

    cw = ones(eltype(nw), 2 * size(nw, 1) + 1, size(nw)[2:3]...)

    for batch in axes(nw, 3)
        for tree in axes(nw, 2)
            for i = 2:2:size(cw, 1)
                cw[i, tree, batch] = cw[i>>1, tree, batch] * nw[i>>1, tree, batch]
                cw[i+1, tree, batch] = cw[i>>1, tree, batch] * (1 - nw[i>>1, tree, batch])
            end
        end
    end

    @views lw = cw[size(nw, 1)+1:size(cw, 1), :, 1:size(nw, 3)]
    return (cw, lw)
end
```

NeuroTreeModels.jl

Recipes to build (Flux) models with NeuroTree operator:

```
chain =  
    Chain(  
        BatchNorm(nfeats),  
        NeuroTree(nfeats => outsize; depth=config.depth, ntrees=config.ntrees),  
    )  
  
chain =  
    Chain(  
        BatchNorm(nfeats),  
        NeuroTree(nfeats => hiddensize;  
        SkipConnexion(NeuroTree(hiddensize => hiddensize), +),  
        Dense(hiddensize => outsize)  
    )
```


API

A simple, MLJ compatible, API for training and inference

```
using NeuroTreeModels, DataFrames

config = NeuroTreeRegressor(
    loss = :mse,
    nrounds = 100,
    num_trees = 16,
    depth = 5,
)

nobs, nfeats = 1_000, 5
dtrain = DataFrame(randn(nobs, nfeats), :auto)
dtrain.y = rand(nobs)
feature_names, target_name = names(dtrain, r"x"), "y"

m = NeuroTreeModels.fit(config, dtrain; feature_names, target_name)
p = m(dtrain)
```

Benchmarks - Regression

Boston 506 observations, 13 features

model_type	train_time	mse	gini
neurotrees	16.6	13.2	0.951
evotrees	0.392	23.5	0.932
xgboost	0.103	21.6	0.931
lightgbm	0.406	26.7	0.931
catboost	0.127	14.9	0.944

Year 515,345 observations, 90 features

model_type	train_time	mse	gini
neurotrees	308.0	76.8	0.651
evotrees	71.9	80.4	0.626
xgboost	33.8	82.0	0.614
lightgbm	15.2	79.4	0.633
catboost	127.0	80.2	0.630

Benchmarks - Classification

Titanic 891 observations, 7 features

model_type	train_time	logloss	accuracy
neurotrees	7.95	0.445	0.821
evotrees	0.11	0.405	0.821
xgboost	0.0512	0.412	0.799
lightgbm	0.128	0.388	0.828
catboost	0.264	0.393	0.843

Higgs 11,000,000 observations, 28 features

model_type	train_time	logloss	accuracy
neurotrees	15900.0	0.453	0.781
evotrees	2710.0	0.465	0.775
xgboost	1390.0	0.464	0.776
lightgbm	993.0	0.464	0.774
catboost	8020.0	0.463	0.776

Benchmarks – Ranking (MSE)

MSRank 1,200,192 observations, 136 features

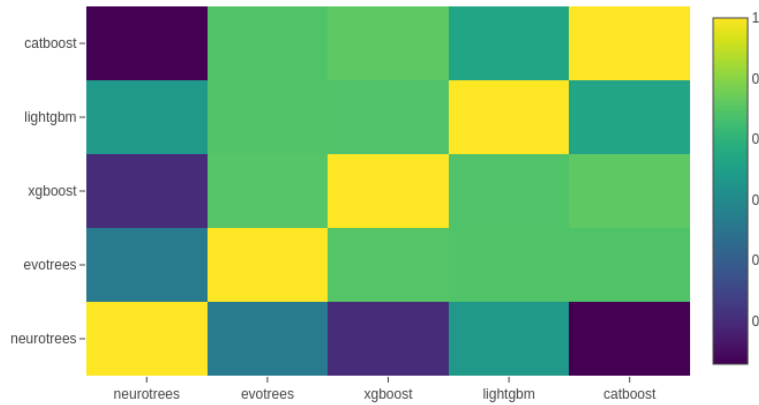
model_type	train_time	mse	ndcg
neurotrees	85.1	0.577	0.467
evotrees	39.8	0.554	0.505
xgboost	19.4	0.554	0.501
lightgbm	38.5	0.553	0.507
catboost	112.0	0.553	0.504

Yahoo 709,877 observations, 519 features

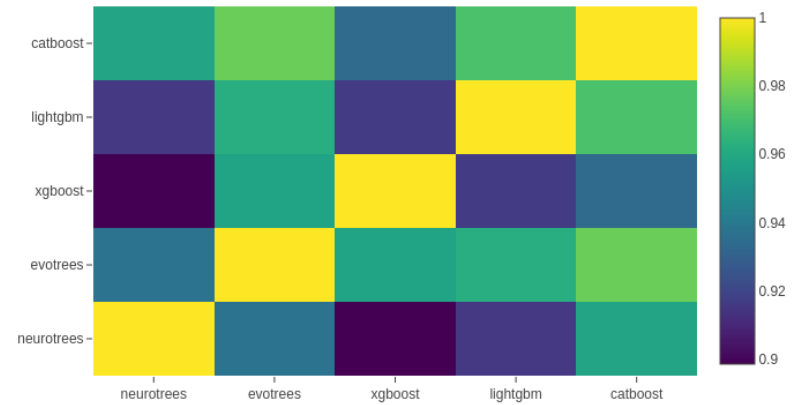
model_type	train_time	mse	ndcg
neurotrees	299.0	0.583	0.781
evotrees	442.0	0.545	0.797
xgboost	129.0	0.544	0.797
lightgbm	215.0	0.539	0.798
catboost	241.0	0.555	0.796

Signal Diversification

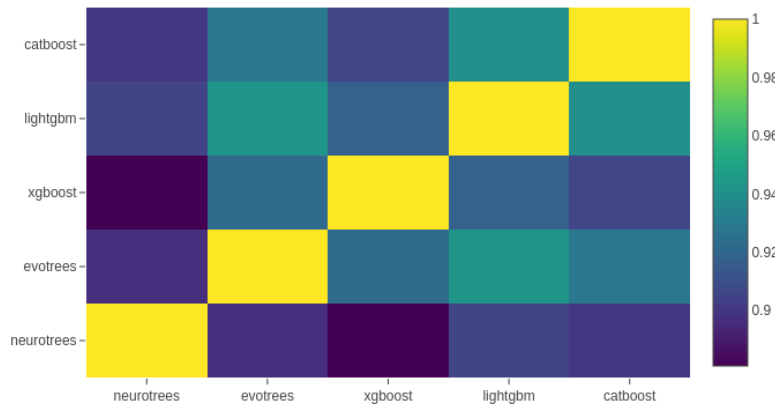
Boston



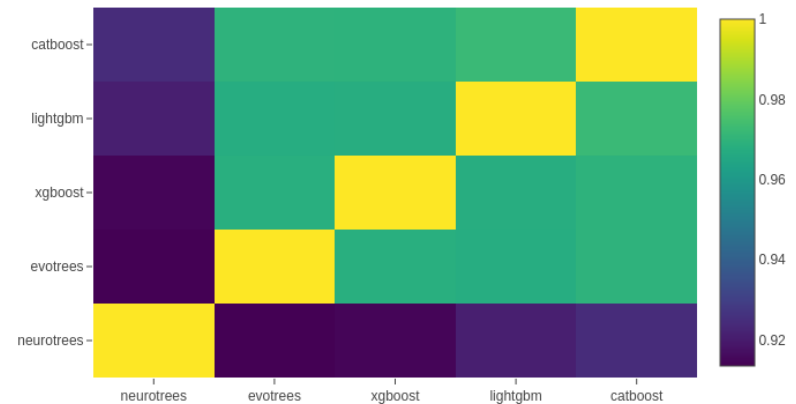
Titanic



Year



Higgs



Future

- **Performance optimisation**
 - Current parallelization, both on CPU and GPU, is currently fairly naïve.
- **Support categorical features through embeddings**
- **Integration with Enzyme.jl**
 - Avoiding the need to manually implement the backward rules while maintaining forward implementation performance is a key target feature for Julia's ML
- **Expand the scope of the NeuroTreeModels to support generic tabular oriented differentiable models (MLP, TabNet...)**

Questions?

