Supplementary material for: Towards Better Decision Forests: Forest Alternating Optimization

Miguel Á. Carreira-Perpiñán Magzhan Gabidolla Arman Zharmagambetov*
Dept. CSE, University of California, Merced
{mcarreira-perpinan, mgabidolla, azharmagambetov}@ucmerced.edu

Abstract

We provide the following. 1) Pseudocode for FAO (section 1). 2) Description of the experiments' setup for reproducibility: datasets, comparison methods, and hyperparameters (section 2). 3) Additional experiments on tabular datasets, the results for CatBoost framework and the result of FAO on axis-aligned decision forests (section 3).

1 Pseudocode

input training set;
initial forest $\mathbf{F}(\cdot;\boldsymbol{\Theta})$ of T trees
number of FAO iterations FI
repeat
  Optimize all leaves jointly by solving
  a regularized linear regression problem
  for $t = 1$ **to** T
    Optimize decision nodes of t's tree $\boldsymbol{\tau}_t(\cdot;\boldsymbol{\Theta}_t)$ using TAO
until FI iterations
prune dead subtrees of **F**
return F

Figure 1: Pseudocode of FAO.

2 Experimental setup

We take 20% of training set (random stratified sample) as a hold-out validation set, and perform grid-search over the hyperparameters of avg-FAO and other baselines. For the found optimal hyperparameters we retrain the models on the whole training set 5 times with different random seed, and report the mean and standard deviation (except for SUSY dataset, because of slower training time.)

2.1 FAO

Implementation For the most part, FAO is implemented in C++ except for fitting leaf weights jointly, for which we use scikit-learn's implementation of Ridge/Lasso and logistic regression. To solve a reduced

*currently at Meta AI (FAIR)

problem at a decision node we use an ℓ_1 regularized logistic regression solver of LIBLINEAR (version 2.43 with support for instance weights). For joint leaf fitting we use scikit-learn (version 1.0.2). For regression we apply an ℓ_2 penalty on leaf weights, and for classification an ℓ_1 penalty both with a fixed parameter `alpha` $= 0.01$ (denoted by μ in the main paper). As an initial forest, we use complete, T trees of depth Δ and random node parameters (drawn from standard normal). We set the number of FAO iterations to 20.

Hyperparameters To save time on hyperparameter tuning, we tune $\lambda = \{0.01, 0.1, 1.0\}$ (parameter of ℓ_1 penalty on decision node weights) on a single decision tree and use it for the whole forest. We consider the following depth $\Delta = \{4, 6, 8, 10, 12\}$, but to save time, depending on the complexity of the dataset, we select only 2-3 of them. As a number of trees T in FAO, we check only $T = \{5, 10\}$, and as a number of forests in averaged FAO Q, we select it so that the total number of trees is at most 300, and the optimal Q is selected on a validation set.

We observe it is quite easier to set hyperparameter values for FAO (at least with avg-FAO) than for other variations GB forests. And we also observe that averaging different GB forests doesn't help.

2.2 Baselines

To compare models of different size, for the baselines we fix the number of trees to {10, 100, 300, 500, 1000}, and perform grid search over other hyperparameters.

XGBoost We use a Python package of version 1.4.1. We use the exact `tree_method`. During cross validation we perform grid search over the following hyperparameter values: `max_depth` = {4, 6, 8, 10, 20}, `eta` = {0.01, 0.05, 0.1, 0.3}.

LightGBM We use a Python package of version 3.2.1. During cross validation we perform grid search over the following hyperparameter values: `num_leaves` = {16, 31, 64, 128, 256, 512}, `learning_rate` = {0.01, 0.05, 0.1, 0.3}.

GB-TAO We quote the results from [2].

SPORF We use a Python package of version 2.0.5. During cross validation we perform grid search over the following hyperparameter values: `projection_matrix` = {RerF, S-RerF}, `max_depth` = {10, 20, None}, `max_features` = {sqrt, log2, None}.

2.3 Estimation of model size

For axis-aligned GB forests (XGBoost and LightGBM) we count the number of parameters as follows: we sum the number of parameters of each node of all the trees in the forest, where an axis-aligned split node counts for two parameters (feature index and threshold) and a constant leaf counts for one parameter. In FAO we exactly estimate the number of nonzero parameters at a decision node. The interface of SPORF does not provide explicit access to tree parameters, and so we provide a reasonable upper bound: the `max_features` parameter controls how many features are used at a decision node, so by assuming that each split node uses exactly `max_features` parameters we estimate the total number of parameters in SPORF.

2.4 Classification datasets

MNIST a standard benchmark. We use direct pixel intensities scaled between 0 and 1 as input [4].

SUSY a problem in physics to classify a process into a signal which produces supersymmetric particles or into a background which does not [1].

CIFAR100 a standard image classification benchmark in computer vision. We use as features the output of the last convolutional layer of a pretrained VGG16 network [3].

News20 a standard document classification benchmark. The features are normalized word counts. Obtained from a LIBSVM multiclass data collection [1].

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

Dataset	N_{train}	N_{test}	D	$\bar{D}_{\text{nnz}}$	K
MNIST	60 000	10 000	784	150	10
CIFAR100 (VGG16 feats)	50 000	10 000	512	324	100
News20	15 935	3 993	62 061	80	20
SUSY	4.5M	0.5M	16	16	10

Table 1: Specs of the datasets used in our experiments for classification. N is a sample size, D is a feature dimension size, $\bar{D}_{\text{nnz}}$ is the average number of nonzero features, and K is the number of classes.

Dataset	N_{train}	N_{test}	D	$\bar{D}_{\text{nnz}}$
cpuact	4 915	3 277	21	21
CASP	29 999	15 731	9	9
CT slice	42 800	10 700	384	378
Year prediction MSD	463 715	51 630	90	90

Table 2: As Table 1, but for regression datasets. The output is 1D for all datasets.

2.5 Regression datasets

cpuact the task is to predict the percentage of time a CPU spends in user mode. The features consist of various statistics of memory and other operations. Obtained from the Delve project [2].

CT slice the task is to predict the relative location of the CT slice on the axial axis of the human body. The features are histograms describing bone structures and air inclusions. We split into train, test, validation sets using by patient ID, so that not to mix patient slices. Obtained from the UCI Machine Learning Repository [5].

CASP a dataset of Physicochemical Properties of Protein Tertiary Structure. The task is to predict the size of the residue. Obtained from the UCI Machine Learning Repository [5].

Year Prediction MSD the task is to predict the release year of a song from audio features. Obtained from the UCI Machine Learning Repository [5].

3 Additional experiment results

Epsilon N_{train}=50k, N_{test}=20k, D=2k, K=2					**HIGGS** N_{train}=100k, N_{test}=50k, D=28, K=2				
Forest	E_{test} (%)	#pars.	T	Δ	Forest	E_{test}	#pars.	T	Δ
CatBoost	16.81±0.04	19k	100	6	CatBoost	28.69±0.00	192k	1k	6
LightGBM	16.52±0.00	19k	100	13	GB-TAO	27.32±0.02	300k	50	8
XGBoost	16.05±0.00	4.6k	100	4	XGBoost	27.29±0.00	18k	100	6
XGBoost	13.68±0.00	173k	1k	6	GB-TAO	27.24±0.02	172k	100	6
CatBoost	13.31±0.02	768k	1k	8	XGBoost	27.08±0.00	42k	1k	4
LightGBM	13.15±0.00	46k	1k	13	LightGBM	26.98±0.00	383k	500	34
avg-FAO	12.38±0.05	59k	10	4	LightGBM	26.93±0.00	766k	1k	37
avg-FAO	11.13±0.04	193k	30	4	**avg-FAO**	26.81±0.02	262k	200	6
GB-TAO	11.05±0.02	353k	30	6	CatBoost	26.72±0.07	768k	1k	8
GB-TAO	10.97±0.02	625k	50	6	**avg-FAO**	26.70±0.01	204k	90	8
avg-FAO	10.90±0.03	382k	60	4	**avg-FAO**	26.51±0.03	335k	150	8

Table 3: Additional experiment results on 2 more datasets: subsampled Epsilon and subsampled HIGGS.

[2] http://www.cs.toronto.edu/~delve/data/comp-activ/desc.html

Dataset	E_{test}	#pars.	T	Δ
MNIST	2.31±0.05	767k	10k	8
CIFAR100	27.67±0.06	4.7k	10k	4
CIFAR100	27.15±0.04	48k	100k	4
SUSY	19.81±0.02	3M	100	8
SUSY	19.55±0.00	768k	1k	8
cpuact	2.76±0.03	4.8k	100	4
cpuact	2.36±0.03	48k	1k	4
CT-slice	6.90±0.08	77k	100	8
CT-slice	6.27±0.06	768k	1k	8
year	8.91±0.02	768k	1k	8
casp	3.60±0.00	768k	1k	8

Table 4: CatBoost results on several datasets from the main paper (hyperparameter search is done identical to XGBoost). Overall, the results are on par with LightGBM and XGBoost, and inferior to avg-FAO.

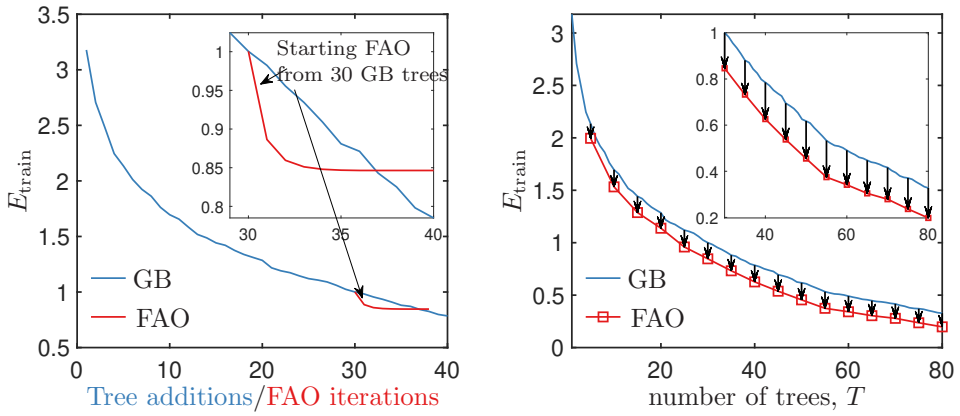

Figure 2: As fig. 1 in the main paper, but for axis-aligned trees.

In Table 3 we provide results for additional tabular datasets: Epsilon and HIGGS. Both datasets are obtained from the LIBSVM binary dataset collection [3]. We subsample the datasets to accelerate the training. In general the results are qualitatively as in the main paper: avg-FAO achieves best accuracy while using fewer trees and parameters.

Table 4 shows results for CatBoost on several datasets from the main paper. We perform hyperparameter search exactly as for XGBoost. Overall, the results are on par with LightGBM and XGBoost, and inferior to avg-FAO.

Fig. 2 shows the result of optimizing a forest of *axis-aligned* decision trees using FAO. The dataset and experimental setup is the same as for fig. 1 of the main paper: a regression problem on the cpuact dataset with trees of depth $\Delta = 6$. The results are qualitatively very similar with oblique decision trees, but the improvement is less dramatic.

References

[1] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5:4308, 2014.

[2] M. Gabidolla and M. Á. Carreira-Perpiñán. Pushing the envelope of gradient boosting forests via globally-optimized oblique trees. In *Proc. of the 2022 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'22)*, pages 285–294, New Orleans, LA, June 19–24 2022.

[3] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

[3] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Dept. of Computer Science, University of Toronto, Apr. 8 2009.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, Nov. 1998.

[5] M. Lichman. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2013.