Improved Multiclass AdaBoost Using Sparse Oblique Decision Trees

Magzhan Gabidolla, Arman Zharmagambetov and
Miguel Á. Carreira-Perpiñán

Dept. of Computer Science and Engineering
University of California, Merced

July 11, 2022

Introduction

- Ensembles of decision trees (= forests) have found numerous applications in many domains.
- They posses multiple advantages, such as strong generalization property, scalability to large data and fast inference time.
- Some examples of forests:
  - *Random forests* train each tree independently on a different data sample and on a different subset of features.
  - *Boosted Trees* sequentially train trees on reweighted versions of the data.

We focus on boosted decision trees for multiclass classification problems.

Overview

- Most of the papers on boosting and implementations of them use trees that are:
  - Axis-aligned (i.e. it uses a single feature at a decision node)
  - Trained with greedy recursive partitioning
- However, axis-aligned trees are not very suitable for many problems, especially for the ones with correlated features (e.g. pixels of an image).
- Greedy top-down induction produces suboptimal trees [4].

Our idea

- We propose the following to address these issues:
  - to use oblique decision trees (i.e. trees with hyperplane splits at decision nodes)
  - to use a non-greedy optimization algorithm to learn such trees
- We adapt the recently proposed algorithm for learning classification/regression trees, Tree Alternating Optimization (TAO) [2, 5], for a specific boosting framework and empirically evaluate its performance on several datasets.
- By monotonically decreasing an objective function over a tree with predetermined structure, TAO finds better approximate optima, and is quite flexible for the choices of objective function and the types of tree (axis-aligned, oblique, etc.).

Boosting algorithm: AdaBoost.M1 and SAMME

Algorithm 1 SAMME pseudocode using TAO trees. The pseudocode for AdaBoost.M1 is slightly different

input: training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ where $y_n \in \{1, \ldots, K\}$;
base learner $\mathbf{T}$; number of boosting steps T; shrinkage factor η;
initial weights (per instance): $\{w_n = \frac{1}{N}\}_{n=1}^N$;
for $t = 1$ **to** T **do**
Train a TAO tree ($\mathbf{T}_t$) on the training set with the current weights to minimize weighted 0-1 loss;
Obtain predictions: $\{\hat{y}_n\}_{n=1}^N \leftarrow \mathbf{T}_t(\{\mathbf{x}_n\}_{n=1}^N)$;
Compute weighted misclassification loss E;
if $E \geq 1 - \frac{1}{K}$ **then**
set $T = t - 1$; exit loop;
end if
Compute $\alpha_t = \eta \cdot (\log \frac{1-E}{E} + \log(K-1))$;
Set $w_n \leftarrow w_n \cdot \exp(\alpha_t \cdot I(y_n \neq \hat{y}_n))$; renormalize $w_1, \ldots, w_N$;
end for
return $F(\mathbf{x}) = \arg\max_k \Sigma_{t=1}^T \alpha_t \cdot I(\mathbf{T}_t(\mathbf{x}) = k)$;

Related work

The roots of DTs are in the 1950s, although they became really popular in the early 1980s. Since then, many approaches have been proposed to train them. Some common approaches:

- Greedy recursive splitting: start from the root and recursively split into two or more children based on solving a "purity" optimization problem (e.g. CART [1]). Simple and fast, but generates suboptimal trees.
- Approximate brute force search: attempts to find an optimal decision tree via mixed-integer programming. Do not scale beyond small or toy datasets.
- Non-greedy, global optimization algorithms: neither of the above. Tries to find approximate solution by optimizing over the entire tree. Well-known example: "soft decision trees". Our proposed algorithm, **Tree Alternating Optimization (TAO)** [2], is in this category but does not use soft trees.

Related work (cont.)

The literature of DTs is not restricted by training a single tree, Various methods have been proposed to ensemble them:

- **Bagging** train each tree independently on a different data sample: Random Forests, Extra Randomized Trees, etc...
- **Boosted Trees** sequentially train trees on reweighted versions of the data: AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost, etc...

Moreover, there are attempts to combine decision trees and other models (say neural nets) and train trees with more complex models at each node (e.g. SVMs, LDA). Some examples: neural decision trees, hierarchical mixture of experts (HME), Naive Bayes trees, etc.

TAO: general formulation

We consider trees whose nodes make hard decisions (not soft trees). Optimizing such trees is difficult because they are not differentiable. Assuming a tree structure $\mathbf{T}$ is given (say, binary complete of depth Δ), consider the following optimization problem over its parameters:

$$E(\mathbf{\Theta}) = \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n; \mathbf{\Theta})) + \alpha \sum_{i \in \mathcal{N}} \phi_i(\boldsymbol{\theta}_i)$$

given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. $\mathbf{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{N}}$ is a set of parameters of all tree nodes. The loss function $L(\mathbf{y}, \mathbf{z})$ can be any loss which separates over training instances (e.g. squared error, cross-entropy, etc.). The regularization term ϕ_i (e.g. ℓ_1 norm) penalizes the parameters $\boldsymbol{\theta}_i$ of each node (to prevent overfitting).

TAO: separability of nodes

The TAO algorithm is based on 3 theorems: separability condition, reduced problem over a leaf, reduced problem over a decision node.

1. Separability condition

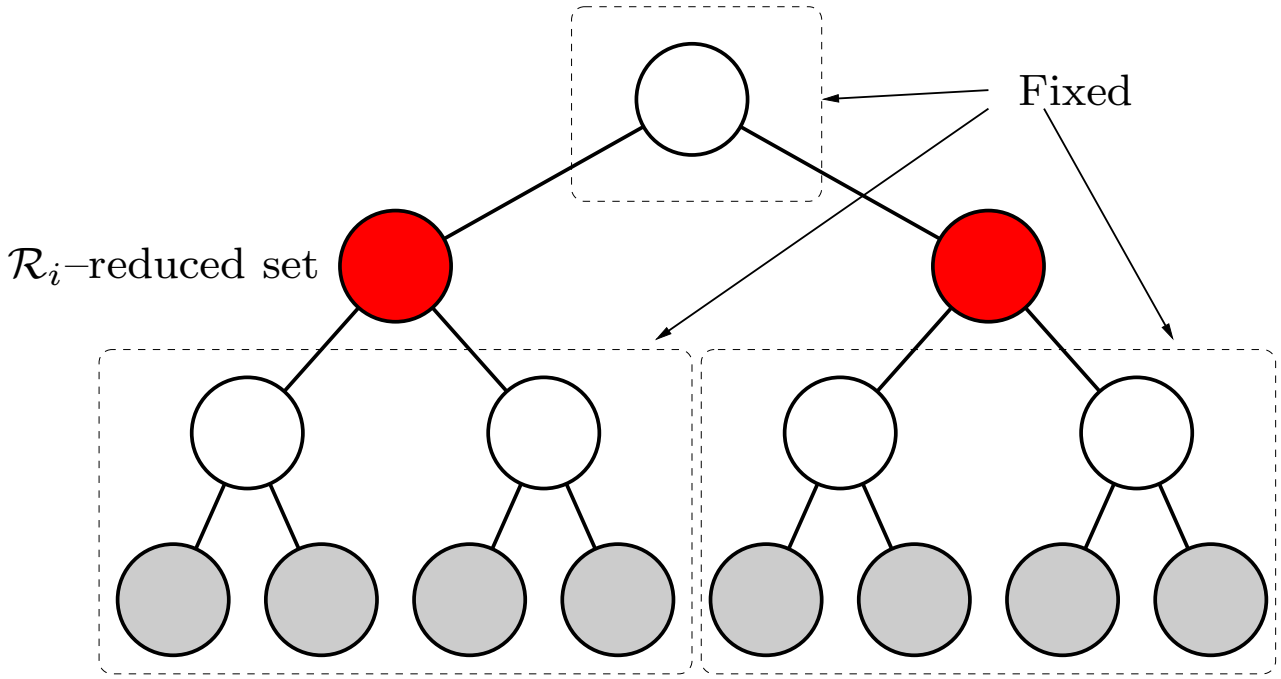
Consider any pair of nodes i and j. Assume the parameters of all other nodes ($\boldsymbol{\Theta}_{\text{rest}}$) are fixed. If nodes i and j are not descendants of each other, then $E(\boldsymbol{\Theta})$ can be rewritten as:

$$E(\boldsymbol{\Theta}) = E_i(\boldsymbol{\theta}_i) + E_j(\boldsymbol{\theta}_j) + E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$$

In other words, the separability condition states that any set of non-descendant nodes of a tree can be optimized independently. Note that $E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$ can be treated as a constant since we fix $\boldsymbol{\Theta}_{\text{rest}}$.

TAO: separability of nodes

- Any set of non-descendant nodes of a tree can be optimized independently:



TAO: optimizing over leaves

A set of non-descendant nodes are all the leaves. Optimizing over the parameters of one leaf is given by the following theorem.

2. Reduced problem over a leaf

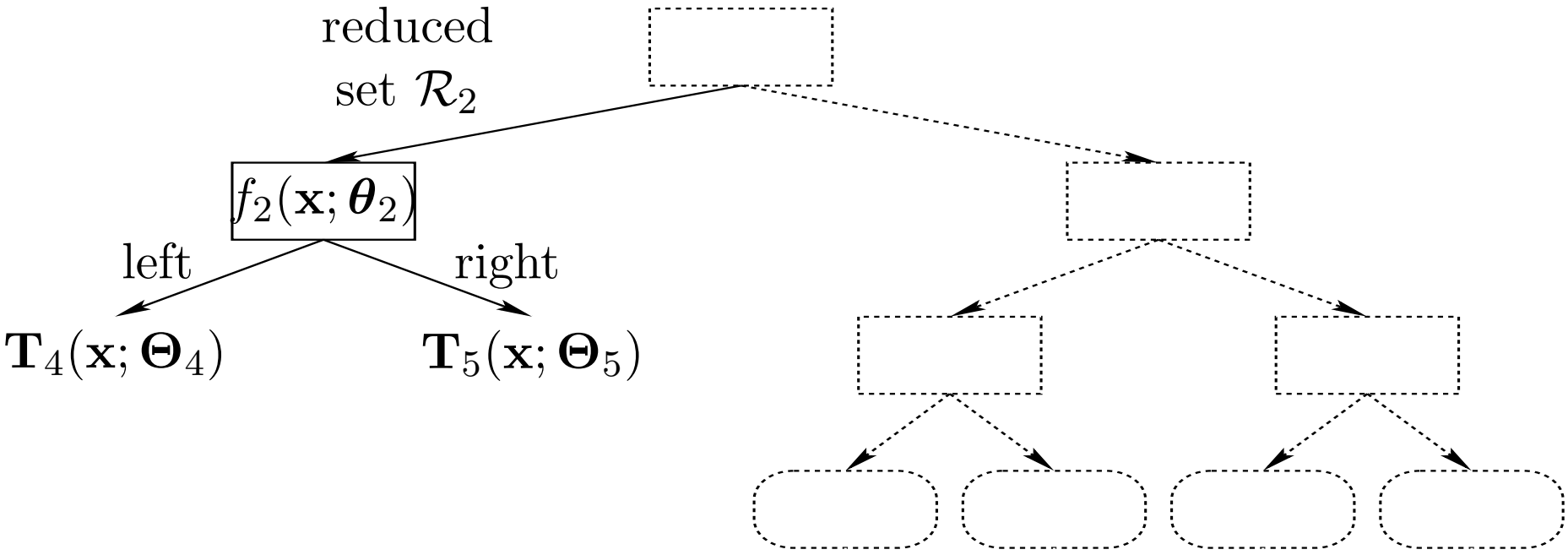
If i is a leaf, the optimization of $E(\boldsymbol{\Theta})$ over $\boldsymbol{\theta}_i$ can be equivalently written as:

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} L(\mathbf{y}_n, \mathbf{g}_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha \, \phi_i(\boldsymbol{\theta}_i)$$

The reduced set $\mathcal{R}_i$ contains the training instances that reach leaf i. Each leaf i has a predictor function $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{R}^K$ that produces the actual output. Therefore, solving the reduced problem over a leaf i amounts to fitting the leaf's predictor $\mathbf{g}_i$ to the instances in its reduced set to minimize the original loss (e.g. squared error).

TAO: optimizing over decision nodes

An example of a set of non-descendant nodes are all the decision nodes at the same depth:



Here, $\mathcal{R}_i$ is the reduced set of node i and (assuming binary trees) $f_i(\mathbf{x};\boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left},\texttt{right}\}$ is a decision function in node i which sends instance $\mathbf{x}_n$ to the corresponding child of i. We consider oblique trees, having hyperplane decision functions "go to right if $\mathbf{w}_i^T\mathbf{x} + w_{i0} \geq 0$" (where $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$).

TAO: optimizing over decision nodes (cont.)

The reduced problem over a decision node can be written as a weighted 0/1 loss binary classification problem on the node's reduced set instances:

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} \overline{L}_{in}(\overline{y}_{in}, f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha\, \phi_i(\boldsymbol{\theta}_i)$$

where the weighted 0/1 loss $\overline{L}_{in}(\overline{y}_{in}, \cdot)$ for instance $n \in \mathcal{R}_i$ is defined as $\overline{L}_{in}(\overline{y}_{in}, y) = l_{in}(y) - l_{in}(\overline{y}_{in})\ \forall y \in \{\texttt{left}, \texttt{right}\}$, where $\overline{y}_{in} = \arg\min_y l_{in}(y)$ is a "pseudolabel" indicating a child which gives the lowest value of the regression loss L for instance $\mathbf{x}_n$ under the current tree.

For hyperplane nodes (oblique trees), this is NP-hard, but can be approximated by using a convex surrogate loss (we use the logistic loss). Hence, if ϕ_i is an ℓ_1 norm, this requires solving an ℓ_1-regularized logistic regression.

Pseudocode for training a single TAO tree

TAO repeatedly alternates optimizing over sets of nodes while monotonically decreasing the objective function.

input training set; initial tree $\mathbf{T}(\cdot; \boldsymbol{\Theta})$ of depth Δ
$\mathcal{N}_0, \dots, \mathcal{N}_\Delta \leftarrow$ nodes at depth $0, \dots, \Delta$, respectively
generate $\mathcal{R}_1 \leftarrow \{1, \dots, N\}$ using initial tree
repeat
  for $d = \Delta$ **down to** 0
    parfor $i \in \mathcal{N}_d$
      if i is a leaf **then**
        $\boldsymbol{\theta}_i \leftarrow$ fit a regressor/classifier (const, linear, neural net, etc.)
          $\mathbf{g}_i$ on reduced set $\mathcal{R}_i$
      else
        generate pseudolabels $\overline{y}_n$ for each point $n \in \mathcal{R}_i$
        $\boldsymbol{\theta}_i \leftarrow$ fit a binary classifier on $\mathcal{R}_i$
  update $\mathcal{R}_i$ for each node
until stop
prune dead subtrees of $\mathbf{T}$
return $\mathbf{T}$

TAO: some success stories

- TAO has been successfully applied in training a single oblique/axis-aligned tree [7]
- Extension for regression appeared in [5]
- Moreover, it has been successfully applied to train hybrid of trees and other models (e.g. neural nets [6])
- Ensemble of TAO trees achieves state-of-the-art performance in number of benchmarks: [3, 5]

Experiments: MNIST dataset

M1 – AdaBoost.M1, S – SAMME

Δ – max depth of the forest, T – number of trees

	Forest	E_{test} (%)	#parameters	T	Δ
MNIST	CART	12.11±0.04	6k	1	50
	TAO	5.25±0.20	24k	1	8
	RF	3.05±0.06	1M	100	46
	S-CART	2.96±0.05	6M	1k	30
	RF	2.84±0.06	10M	1k	48
	sNDF	2.80±0.12	22M	80	10
	ADF	2.71±0.10	(3.6M)	100	25
	XGBoost	2.67±0.00	0.3M	1k	8
	S-CART	2.28±0.02	13M	1k	16
	M1-TAO	2.09±0.04	0.8M	30	8
	rRF	2.05±0.02	(160k)	100	25
	XGBoost	1.94±0.00	0.6M	10k	8
	S-TAO	1.93±0.02	0.8M	30	8
	M1-TAO	1.74±0.02	2.6M	100	8
	S-TAO	1.67±0.04	2.6M	100	8

Boosted TAO trees are smaller (fewer and shallower trees) yet consistently more accurate.

Experiments: Letter dataset

M1 – AdaBoost.M1, S – SAMME

Δ – max depth of the forest, T – number of trees

	Forest	E_{test} (%)	#parameters	T	Δ
Letter	CART	13.06±0.15	3k	1	27
	TAO	9.59±0.31	10k	1	11
	XGBoost	4.30±0.00	0.4M	2.6k	10
	RF	3.77±0.06	0.4M	100	34
	ADF	3.52±0.12	(1M)	100	25
	RF	3.44±0.09	4.2M	1k	36
	XGBoost	3.35±0.00	0.8M	26k	6
	S-CART	2.83±0.15	0.7M	100	16
	rRF	2.98±0.15	(180k)	100	25
	sNDF	2.92±0.17	2.4M	70	10
	S-CART	2.58±0.09	6.7M	1k	16
	M1-TAO	1.85±0.09	0.2M	30	11
	S-TAO	1.79±0.07	0.2M	30	11
	M1-TAO	1.40±0.09	0.6M	100	11
	S-TAO	1.38±0.03	0.6M	100	11

Boosted TAO trees are smaller (fewer and shallower trees) yet consistently more accurate.

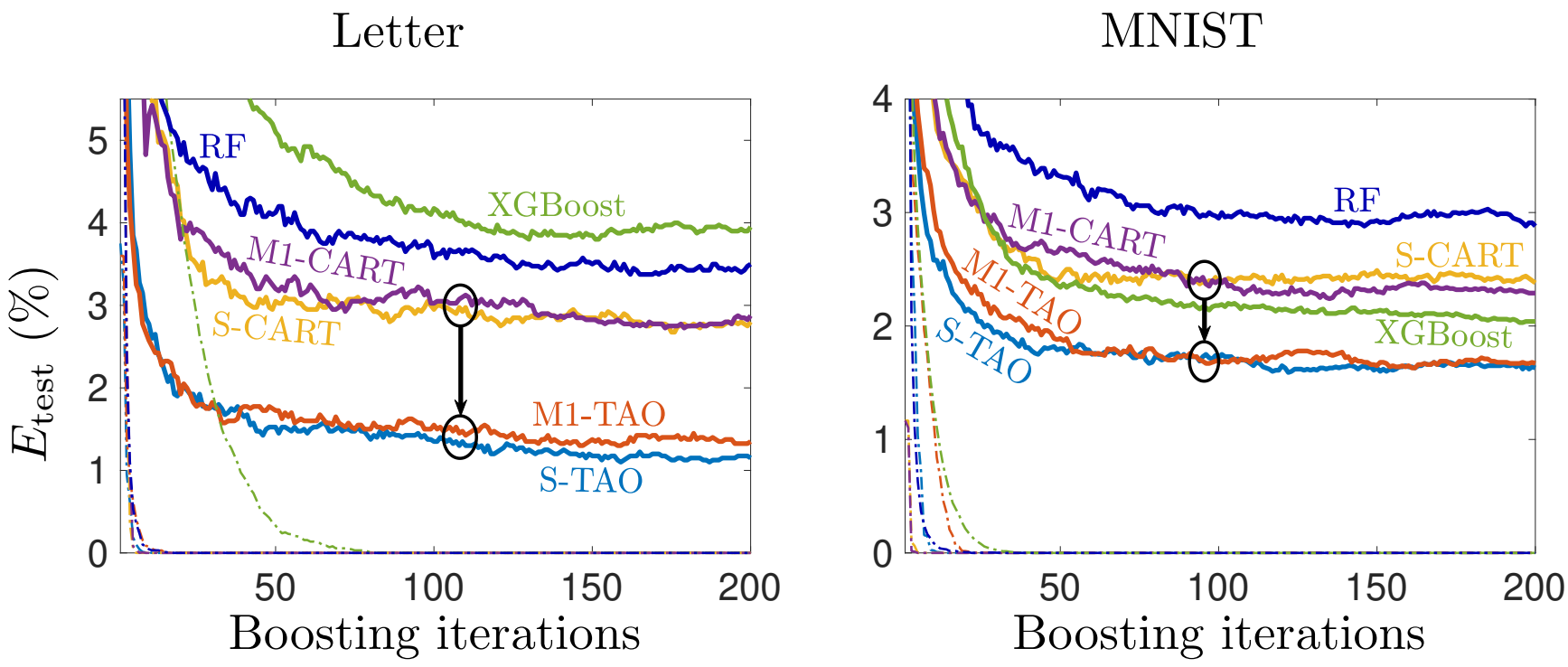
Experiments: R8 dataset

M1 – AdaBoost.M1, S – SAMME
Δ – max depth of the forest, T – number of trees

	Forest	E_{test} (%)	#parameters	T	Δ
R8	TAO	6.64±1.04	3k	1	7
	RF	6.16±0.35	93k	100	27
	S-CART	5.85±0.07	61k	100	20
	RF	5.57±0.56	0.9M	1k	27
	XGBoost	5.34±0.00	51k	800	6
	S-CART	5.11±0.09	0.6M	1k	20
	XGBoost	4.89±0.00	83k	8k	6
	S-TAO	3.12±0.10	0.6M	100	7
	M1-TAO	3.06±0.14	0.6M	100	7

Boosted TAO trees are smaller (fewer and shallower trees) yet consistently more accurate.

Comparison in terms of Boosting Iterations



Comparison in terms of Training Time

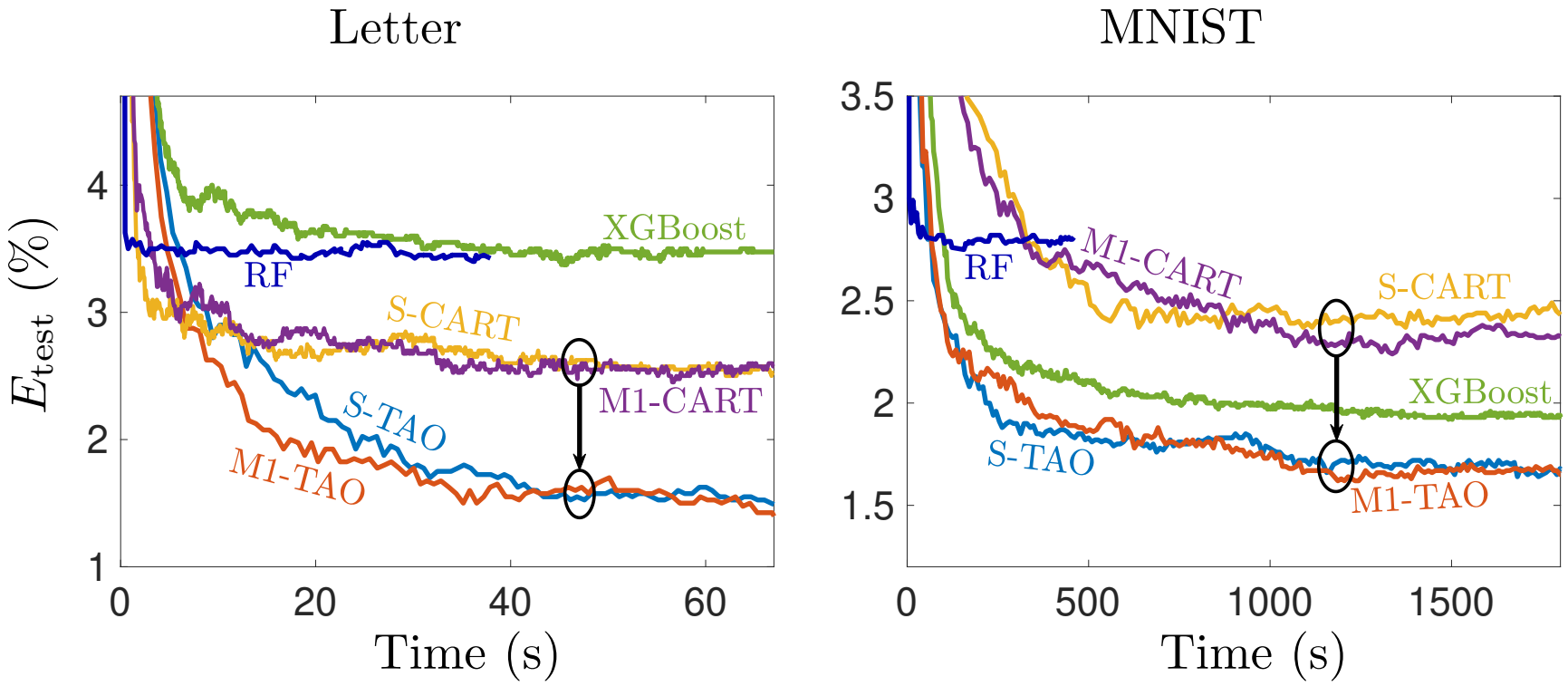


Figure: All methods except "*-CART" use parallel training with 8 threads.

Conclusion

- Directly and non-greedily optimizing the base learner's objective function in AdaBoost with TAO significantly improves the performance of the ensemble.
  - Boosted TAO trees outperform all competing algorithms we tested in terms of accuracy.
  - The TAO forests are small in terms of model size: number of trees, total number of parameters, depth.
- The design in terms of hyperparameter tuning remains as simple as the original boosting: we choose the tree depth and number of trees as large as computationally possible, but without overfitting.
- This makes our TAO forests a model of immediate, widespread practical applicability and impact

References

[1] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

[2] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NEURIPS)*, volume 31, pages 1211–1221. MIT Press, Cambridge, MA, 2018.

[3] 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)*, New Orleans, LA, June 19–24 2022.

[4] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning—Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer-Verlag, second edition, 2009.

[5] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Smaller, more accurate regression forests using tree alternating optimization. In H. Daumé III and A. Singh, editors, *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pages 11398–11408, Online, July 13–18 2020.

[6] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Learning a tree of neural nets. In *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'21)*, pages 3140–3144, Toronto, Canada, June 6–11 2021.

[7] A. Zharmagambetov, S. S. Hada, M. Gabidolla, and M. Á. Carreira-Perpiñán. Non-greedy algorithms for decision tree optimization: An experimental comparison. In *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, July 18–22 2021.