Boosted Sparse Oblique Decision Trees

Magzhan Gabidolla **Arman Zharmagambetov** **Miguel Á. Carreira-Perpiñán**
Dept. of Computer Science and Engineering, University of California, Merced

Introduction Boosted decision trees are widely used machine learning algorithms, achieving state-of-the-art performance in many domains with little effort on hyperparameter tuning. Though much work on boosting has focused on the theoretical properties and empirical variations, there has been little progress on the tree learning procedure itself. To this day, boosting algorithms employ regular axis-aligned trees as base learners optimized by CART-style greedy top-down induction. These trees are known to be highly suboptimal due to their greedy nature, and they are not well-suited to model the correlation of features due to their axis-aligned partition. In fact, these suboptimality characteristics are commonly believed to be beneficial because of the *weak learning* criterion in boosting.

In this work we consider boosting better optimized sparse *oblique* decision trees trained with the recently proposed Tree Alternating Optimization (TAO) [1]. TAO generally finds much better approximate optima than CART-type algorithms due to the ability to monotonically decrease a desired objective function over a decision tree. Our extensive experimental results demonstrate that boosted sparse oblique TAO trees improve upon CART trees by a large margin, and achieve better test error than other popular tree ensembles such as gradient boosting (XGBoost) and random forests. Moreover, the resulting TAO ensembles require far smaller number of trees.

Boosted TAO trees Many boosting algorithms exist, but in this work we are focusing on two similar, effective, yet simple AdaBoost variations: AdaBoost.M1 and SAMME. Although TAO can optimize the base learner's objective in any boosting algorithm including AdaBoost.MH, LogitBoost and gradient boosting, the purpose of this work is to provide convincing evidence that boosting better optimized and more powerful trees can consistently improve over traditional tree learning methods.

In both AdaBoost.M1 and SAMME, the objective of the base learner is to minimize weighted misclassification loss, and here we will briefly describe how TAO operates to optimize that objective function. TAO takes an initial tree of a predetermined structure, randomly initializes its parameters and performs iterative updates on the node parameters to monotonically decrease the objective function. As the initial tree structure, we take a complete binary tree $\mathbf{T}$ of depth Δ with nodes i and parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}$. Then, at each boosting step, TAO optimizes the following objective:

$$\min_{\boldsymbol{\Theta}} E(\boldsymbol{\Theta}) = \sum_{n=1}^{N} w_n L_n(y_n, \mathbf{T}(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda \sum_{\text{nodes } i} \phi(\boldsymbol{\theta}_i) \tag{1}$$

where L is 0/1 classification loss defined per data point $\mathbf{x}_n$ with ground-truth label y_n, and ϕ is the regularization term that penalizes the parameters $\boldsymbol{\theta}_i$ of each node. In this work, we use sparsity penalty $\phi(\boldsymbol{\theta}_i) = \|\boldsymbol{\theta}_i\|_1$ with hyperparameter $\lambda \geq 0$ that controls the regularization severity. w_n is the weight per data point coming from the boosting algorithm and it satisfies the following condition: $\sum_n w_n = 1$ and $w_n \geq 0$. Note that we are including extra regularization term to the base learner's objective in AdaBoost.M1 and SAMME.

TAO is based on two theorems which we describe briefly here (refer to [1] for details) and adapt them for the boosting framework. We define the *reduced set* $\mathcal{R}_i \subset \{1, \dots, N\}$ of node i (decision node or leaf) as the training instances that reach i given the current tree parameters.

Separability condition For the nodes i and j that are not descendants of each other, we can rewrite E equivalently as separable functions of θ_i and θ_j.

Reduced problem Given the separability condition, we can optimize a non-descendant set of nodes independently. Optimizing eq. (1) over a leaf is equivalent to training its model parameters $\boldsymbol{\theta}_i$ on subset of points that reach the leaf, i.e. solving the original classification problem on its reduced set. Since we are using a constant label leaf, this can be solved exactly by majority vote.

Using the separability condition, we can rewrite eq. (1) to optimize a decision node i equivalently as a function of $\boldsymbol{\theta}_i$:

$$\min_{\boldsymbol{\Theta}} E(\boldsymbol{\Theta}) = \min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \min_{\boldsymbol{\theta}_i} \sum_{n \in \mathcal{R}_i} w_n l(f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \lambda \phi_i(\boldsymbol{\theta}_i) \tag{2}$$

where $\mathcal{R}_i$ is the reduced set of that particular node. Since $f_i \in \{\text{right,left}\}$ can only have two possible values, we define $l(\cdot)$ as an incurring loss of choosing right or left subtree. Hence, we can rewrite eq. (2) as the following equivalent optimization problem:

$$\min_{\boldsymbol{\theta}_i} \sum_{n \in \mathcal{R}_i} w_n L(\overline{y}_n, f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \lambda \phi_i(\boldsymbol{\theta}_i) \tag{3}$$

where L is the same 0/1 classification loss mentioned before and $\overline{y}_n \in \{\text{right,left}\}$ is a "pseudolabel" indicating the child which gives a lower value of E for instance $\mathbf{x}_n$ under the current tree. The problem (3) above is a weighted 0/1 loss binary classification problem and optimizing it is NP-hard problem [3, 4] even for unweighted case. However, we can approximate it with a convex surrogate loss such as logistic loss (possibly with various regularizations). We use LIBLINEAR [2] which also can handle weights per data point.

TAO algorithm processes the nodes of a tree in a breadth-first order (depth by depth) and repeatedly trains a binary classifier (decision node) and a K-class classifier (leaf). All nodes at the same depth are trained independently due to separability condition and in parallel.

Experiments Below we compare boosted TAO trees against boosted CART trees, XGBoost and random forests as a function of the number of trees and training time. For all the models, we select best hyperparameters using cross validation. Clearly, TAO forests significantly improve upon all the other tree ensembles.

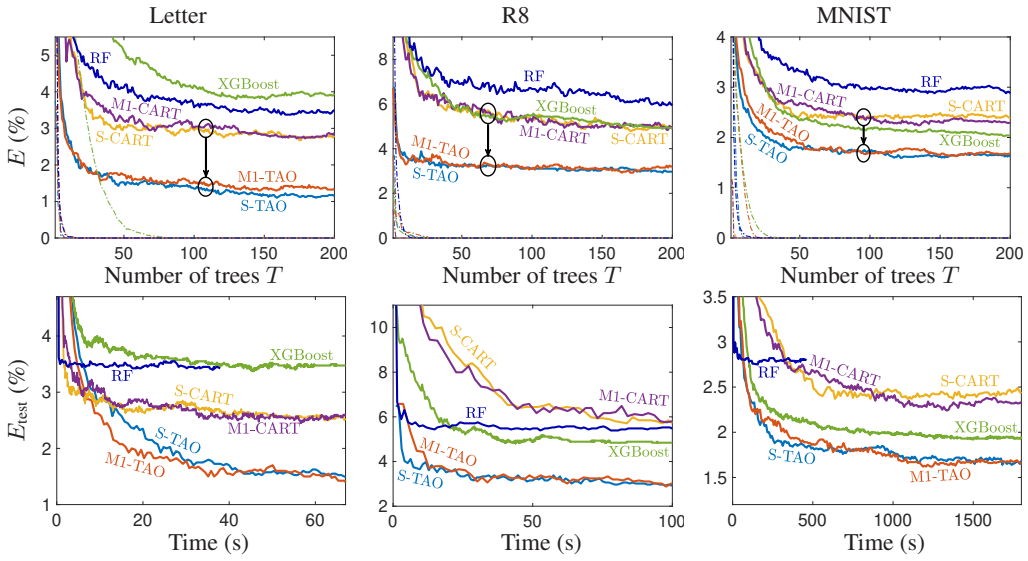

Figure 1: Comparison between different tree ensembles as a function of the number of trees T (top) and training time (bottom). Solid lines - test errors, dashed lines - train errors. "M1" - AdaBoost.M1, "S" - SAMME, RF - random forest. All methods except "*-CART" use parallel training with 8 threads.

References

[1] 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. 1

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *J. Machine Learning Research*, 9:1871–1874, Aug. 2008. 2

[3] K.-U. Hoffgen, H. U. Simon, and K. S. Vanhorn. Robust trainability of single neurons. *J. Computer and System Sciences*, 50(1):114–125, Feb. 1995. 2

[4] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, Oct. 1988. 2