Improved Boosted Regression Forests Through Non-Greedy Tree Optimization

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

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

IEEE IJCNN, July 2021

Introduction

- An ensemble of decision trees (= forest) is a widely established method in machine learning with many successful applications.
- They often produce very accurate results with very little effort on hyperparameter tuning, and are part of many winning methods on ML challenges and competitions.
- Some examples of forests:
    - *Random forests* train each tree independently on a different data sample (bagging).
    - *Boosted Trees* sequentially train trees on reweighted versions of the data.

We focus on boosted decision trees for regression problems, where the output can be a scalar or a vector.

Overview

- Majority of papers and implementations of boosting algorithms use decision trees that are:
  - Axis-aligned (i.e. tests a single feature at a decision node e.g. "if $x_7 \geq 3$ then go right")
  - Trained with greedy top-down induction.
- However, axis-aligned trees are quite restricted models, especially for problems with correlated features.
- And, it has been shown and widely accepted that the trees produced by greedy top-down induction are suboptimal (such as CART trees) [4].

Our idea

- We address both issues by:
  - using trees with more complex nodes (oblique, i.e., hyperplane)
  - using a better optimization algorithm to learn the tree.
- We build on a recently proposed algorithm for learning classification/regression trees, Tree Alternating Optimization (TAO) [1, 5]. TAO finds good approximate optima of an objective function over a tree with predetermined structure and it applies to trees beyond axis-aligned splits.
- We use an oblique decision tree as a base learner in a specific boosting algorithm for regression problems, and adapt the TAO algorithm to optimize the base learner's objective function.

Boosting for regression: AdaBoost.R2

- Quite a few boosting algorithms exist for regression:
  - AdaBoost.R2, AdaBoost.RT, gradient boosting, etc.
- In this work, we focus on AdaBoost.R2 [3]:
  - The first practical boosting algorithm for regression
  - It has similar intuitive behavior with instance weights as AdaBoost.M1: more difficult points obtain higher weight (importance) in the next boosting step.
- The main point of this work is to show that stronger base learners trained with TAO can improve the overall performance of the ensemble, and we believe that this will apply to any boosting algorithm.

AdaBoost.R2 pseudocode

Algorithm 1: AdaBoost.R2 with TAO modification

```
Result: Forest of boosted TAO trees $\mathbf{F}$
input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$,   number of trees $T$,
  learning rate $\eta$,   initial weights $\{w_n = 1/N\}_{n=1}^N$
for $t = 1$ to $T$ do
    $\mathbf{T} \leftarrow$ fit TAO with instance weights $\{w_n\}_{n=1}^N$
    obtain predictions $\{\hat{y}_n\}_{n=1}^N \leftarrow \mathbf{T}(\{\mathbf{x}_n\}_{n=1}^N)$
    calculate a loss per instance $l_n \leftarrow \frac{|\hat{y}_n - y_n|}{D}$
      where $D = \max_n |\hat{y}_n - y_n|$
    calculate the average loss $\bar{L} = \Sigma_{n=1}^N w_n l_n$
    if $\bar{L} > 0.5$ then
        set $T = t - 1$, exit the loop
    end
    $\beta_t = \frac{\bar{L}}{1-\bar{L}}$, weight of the current tree $\alpha_t = \log(1/\beta_t)$
    update the instance weights $w_n \leftarrow w_n \beta_t^{1-l_n}$
end
$F(\mathbf{x})$ = weighted median of $\{\mathbf{T}(\mathbf{x})\}_{t=1}^T$
```

Optimizing a single tree with 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, complete tree of depth Δ), consider the following optimization problem over its parameters:

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

- $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ is a training set with weights $\{w_n\}_{n=1}^N$
- $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{N}}$ is a set of parameters of all tree nodes
- $L(\mathbf{y}, \mathbf{z})$ is a loss function. We use the squared error $\|\mathbf{y} - \mathbf{z}\|_2^2$ (it is possible to use other losses, e.g. the least absolute deviation or a robust loss)
- ϕ_i is a regularization term (e.g. ℓ_1 norm), which penalizes the parameters $\boldsymbol{\theta}_i$ of each node.

Optimizing a single tree with TAO: separability of nodes

Our TAO algorithm for regression 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 $\boldsymbol{\Theta}_{\text{rest}}$.

Optimizing a single tree with TAO: 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:

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

- Reduced set $\mathcal{R}_i$ is the set of training points that reach leaf i
- $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{R}^K$ is a predictor function of leaf i that produces the tree output. We use a constant or linear model.
- Solving the reduced problem over a leaf i amounts to fitting $\mathbf{g}_i$ to the instances in $\mathcal{R}_i$ to minimize the original loss.

Optimizing a single tree with TAO: decision nodes

3. Reduced problem over a decision node

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

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} l_{in}(f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \lambda\, \phi_i(\boldsymbol{\theta}_i)$$

- $\mathcal{R}_i$ is the reduced set of node i
- $f_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left}, \texttt{right}\}$ 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 $\mathbf{w}_i^T \mathbf{x} + b_i \geq 0$" ($\boldsymbol{\theta}_i = \{\mathbf{w}_i, b_i\}$)
- $l_{in}(\cdot)$ is the loss incurred if $\mathbf{x}_n$ goes to the left or right subtree.

Optimizing a single tree with TAO: decision nodes (cont.)

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

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} \overline{w}_n \overline{L}_{in}(\overline{y}_{in}, f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \lambda\, \phi_i(\boldsymbol{\theta}_i) \qquad (1)$$

- $\overline{L}_{in}(\overline{y}_{in}, \cdot)$ is a 0/1 misclassification loss
- $\overline{y}_{in} \in \{\text{left}, \text{right}\}$ is a "pseudolabel" indicating the child which gives a lower value of E under the current tree.
- $\overline{w}_i = l_{in}(\hat{y}) - l_{in}(\overline{y}_{in})$ where $\hat{y}$ is the other child $\hat{y} \neq \overline{y}$

For hyperplane nodes, 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

Algorithm 2: Learning a single TAO tree with boosting weights

```
Result: trained tree $\mathbf{T}$
input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$; random tree $\mathbf{T}(\cdot;\boldsymbol{\Theta})$ of depth $\Delta$;
  AdaBoost weights $\{w_n\}_{n=1}^N$;
repeat
    for depth $d = 0$ to $\Delta$ do
        for $i \in$ nodes at depth $d$ do
            if $i$ is a leaf then
                $\boldsymbol{\theta}_i \leftarrow$ fit a weighted regressor (constant or linear) on a
                  reduced set $\mathcal{R}_i$ with the current weights $\{w_n\}$;
            else
                $\boldsymbol{\theta}_i \leftarrow$ fit a weighted binary classifier to minimize eq. (1)
            end
        end
    end
until convergence occurs or max iteration;
postprocessing: remove dead or pure subtrees;
```

Experiments: standard benchmarks and algorithms

R2 TAO-c: boosted oblique trees with constant leaves,

R2 TAO-l: boosted oblique trees with linear leaves.

See the paper for extended results, additional datasets, etc.

	Forest	E_{test}	T	Δ
CT slice	CART	2.71±0.06	1	51
	TAO-c	1.54±0.05	1	7
	R2 CART	1.48±0.03	100	10
	XGBoost	1.45±0.00	100	10
	R2 CART	1.31±0.01	1k	10
	XGBoost	1.18±0.00	1k	10
	RF	1.03±0.01	100	71
	cRF[2]	1.00	1k	–
	RF	0.97±0.01	1k	78
	R2 TAO-c	0.59±0.00	30	8
	R2 TAO-c	0.51±0.00	100	8
	R2 TAO-c	0.31±0.00	100	12

	Forest	E_{test}	T	Δ
YearPredictionMSD	CART	13.41±0.11	1	49
	RF	9.31±0.00	100	68
	R2 CART	9.25±0.01	100	15
	RF	9.23±0.00	1k	73
	R2 CART	9.21±0.03	1k	15
	TAO-c	9.11±0.05	1	8
	XGBoost	9.04±0.00	100	10
	XGBoost	9.01±0.00	1k	10
	cRF[2]	8.90	1000	–
	R2 TAO-c	8.85±0.00	30	10
	R2 TAO-c	8.83±0.00	100	10
	R2 TAO-l	8.82±0.00	50	7

Boosted TAO regression trees are smaller (fewer and shallower trees) yet consistently more accurate, particularly if using linear predictors at the leaves.

Comparison with other forests

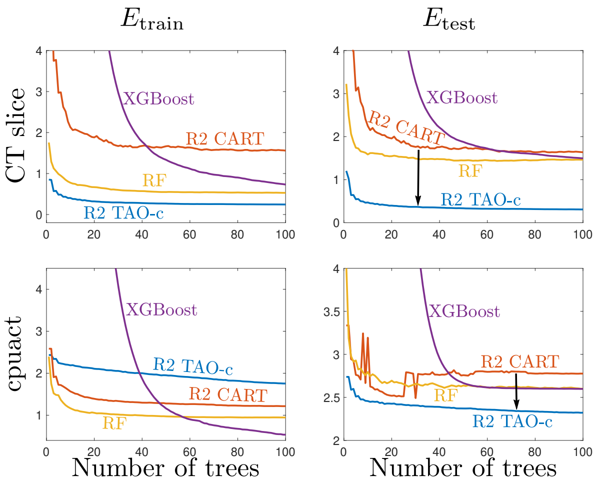

Figure: Comparison of different regression forests on the CT slice and cpuact datasets as a function of the number of trees. “R2” refers to AdaBoost.R2, “RF” refers to Random Forest. All errors are RMSE.

Hyperparameter exploration

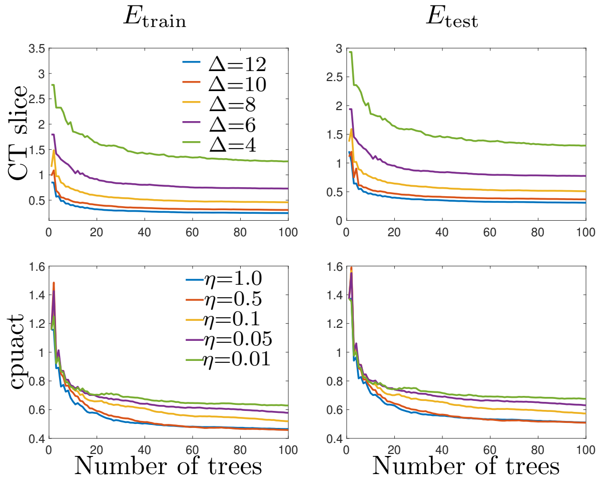

Figure: The effect of the depth Δ (top row) and the learning rate η (bottom row) of the boosted TAO trees on CT slice and cpuact. All errors are RMSE.

Conclusion

- Using a better optimized and more complex tree as a base learner in a specific boosting algorithm improves the overall performance of the ensemble.
    - Our boosted TAO regression trees outperform all competing algorithms we tested in terms of accuracy.
    - The TAO forests are smaller 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, and the optimal learning rate can be found using cross validation.
- This makes our TAO forests a model of immediate, widespread practical applicability and impact

In separate papers, we have also found that TAO trees improve significantly in classification (rather than regression) and with bagging (rather than boosting).

[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.

[2] M. Denil, D. Matheson, and N. de Freitas. Narrowing the gap: Random forests in theory and in practice. In E. P. Xing and T. Jebara, editors, *Proc. of the 31st Int. Conf. Machine Learning (ICML 2014)*, pages 665–673, Beijing, China, June 21–26 2014.

[3] H. Drucker. Improving regressors using boosting techniques. In D. H. Fisher, editor, *Proc. of the 14th Int. Conf. Machine Learning (ICML'97)*, pages 107–115, Nashville, TN, July 6–12 1997.

[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.