Smaller, more accurate regression forests using tree alternating optimization

Arman Zharmagambetov and Miguel Á. Carreira-Perpiñán

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

ICML, July 2020

Overview

- A forest (= ensemble of decision trees) is a powerful machine learning method that has been successfully applied in many applications: computer vision, speech processing, NLP, etc. They are often (part of) the winning methods in ML competitions and challenges.
- 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.

  In both cases, the forest prediction is obtained by weighted averaging or voting.

We focus on regression forests, where the prediction is a continuous scalar or vector, using bagging.

Overview

Random forests (and their variations) for regression have important advantages:

- high predictive accuracy
- few hyperparameters to set
- reasonably fast to train and can be scaled to large datasets
- considered to be robust against overfitting.

But they have an important disadvantage: the individual trees they learn are far from accurate. This is due to two reasons:

- Each tree is axis-aligned (a decision node tests for a single feature, e.g. "if $x_7 \geq 3$ then go right"). This is a very restrictive model, particularly for particularly for correlated features.
- Standard tree learning algorithms, based on a greedy recursive tree growth (such as CART), are highly suboptimal.

There are exist few works that propose forests of more complex trees (Menze et al., 2000; Breiman, 2001; Frank & Kramer, 2004), but they mostly focus on classification rather than regression and improve marginally over conventional axis-aligned trees.

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.

  The resulting forests are smaller, shallower and much more accurate, consistently over various datasets.
- We build on a recently proposed algorithm for learning classification trees, Tree Alternating Optimization (TAO) (Carreira-Perpiñán et al. 2018). 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 adapt TAO to the regression case and then use it in combination with bagging to learn forests of oblique trees.

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, binary complete of depth Δ), consider the following optimization problem over its parameters:

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

given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{N}}$ is a set of parameters of all tree nodes. The loss function $L(\mathbf{y}, \mathbf{z})$ is the squared error $\|\mathbf{y} - \mathbf{z}\|_2^2$ in our case (although it is possible to use other losses, such as the least absolute deviation or a robust loss). The regularization term ϕ_i (e.g. ℓ_1 norm) 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:

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

The reduced set $\mathcal{R}_i$ contains the training instances that reach leaf i. Each leaf i has a predictor function $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{R}^K$ (we use a constant or linear regressor) 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).

Optimizing a single tree with TAO: decision nodes

An example of a set of non-descendant nodes are all the decision nodes at the same depth. Optimizing over the parameters of one decision node is given by the following theorem.

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:

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

where $\mathcal{R}_i$ 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 $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$). $l_{in}(\cdot)$ is the loss incurred if $\mathbf{x}_n$ chooses the right or left 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:

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

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, 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 by training a (binary) classifier in the decision nodes and a regressor in the leaves, while monotonically decr. the obj. function $E(\boldsymbol{\Theta})$.

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
$\mathcal{R}_1 \leftarrow \{1,\dots,N\}$
repeat
  for $d = 0$ **to** Δ
    parfor $i \in \mathcal{N}_d$
      if i is a leaf **then**
        $\boldsymbol{\theta}_i \leftarrow$ train regressor $\mathbf{g}_i$ on reduced set $\mathcal{R}_i$
      else
        $\boldsymbol{\theta}_i \leftarrow$ train decision function f_i on $\mathcal{R}_i$
        compute the reduced sets of each child of i
until stop
prune dead subtrees of $\mathbf{T}$
return $\mathbf{T}$

Ensemble of TAO trees

- Out of many ways to ensemble individual learners (bagging, boosting, etc.), we choose a simple one: we train each TAO tree independently (and in parallel) on a random subset of M samples of the available training data (N instances). If $M = N$ this is bagging.
  The forest prediction is the average of its trees' predictions.
- Although our TAO regression algorithm works more generally, our experiments use:
  - oblique trees ($\mathbf{f}_i$ is linear), which are more powerful than axis-aligned trees
  - constant- and linear-predictor leaves ($\mathbf{g}_i$).
- We initialize each TAO tree ($\mathbf{T}$) from a complete tree of depth Δ and random node parameters.
- We train each tree with an ℓ_1 regularizer to achieve a more compact model: it encourages the weight vectors of individual nodes to be sparse and leads to more dead subtrees which can be pruned.

Experiments: standard benchmarks and algorithms

TAO-c: our oblique trees with constant leaves,
TAO-l: our oblique trees with linear leaves.
See the paper for extended results, additional datasets, etc.

cpuact (N=8k,D=21)

Forest	E_{test}	T	Δ
CART	3.63±0.32	1	25
TAO-c	2.71±0.04	1	6
RF	2.62±0.04	100	36
ARF	2.62±0.01	50	15
AdaBoost	2.61±0.16	100	10
RF	2.60±0.01	1k	37
XGBoost	2.60±0.00	100	10
ET	2.58±0.03	100	45
TAO-l	2.58±0.02	1	5
XGBoost	2.57±0.00	1k	10
AdaBoost	2.56±0.11	1k	10
ET	2.49±0.03	1k	50
TAO-c	2.39±0.05	30	7
TAO-l	2.35±0.01	30	5

CT slice (N=54k,D=384)

Forest	E_{test}	T	Δ
CART	2.71±0.06	1	51
TAO-c	1.54±0.05	1	7
AdaBoost	1.48±0.03	100	10
XGBoost	1.45±0.00	100	10
AdaBoost	1.31±0.01	1k	10
XGBoost	1.18±0.00	1k	10
TAO-l	1.16±0.02	1	5
ET	1.06±0.01	100	82
RF	1.03±0.01	100	71
cRF	1.00	1k	–
RF	0.97±0.01	1k	78
TAO-c	0.89±0.02	30	7
TAO-l	0.58±0.03	30	6

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

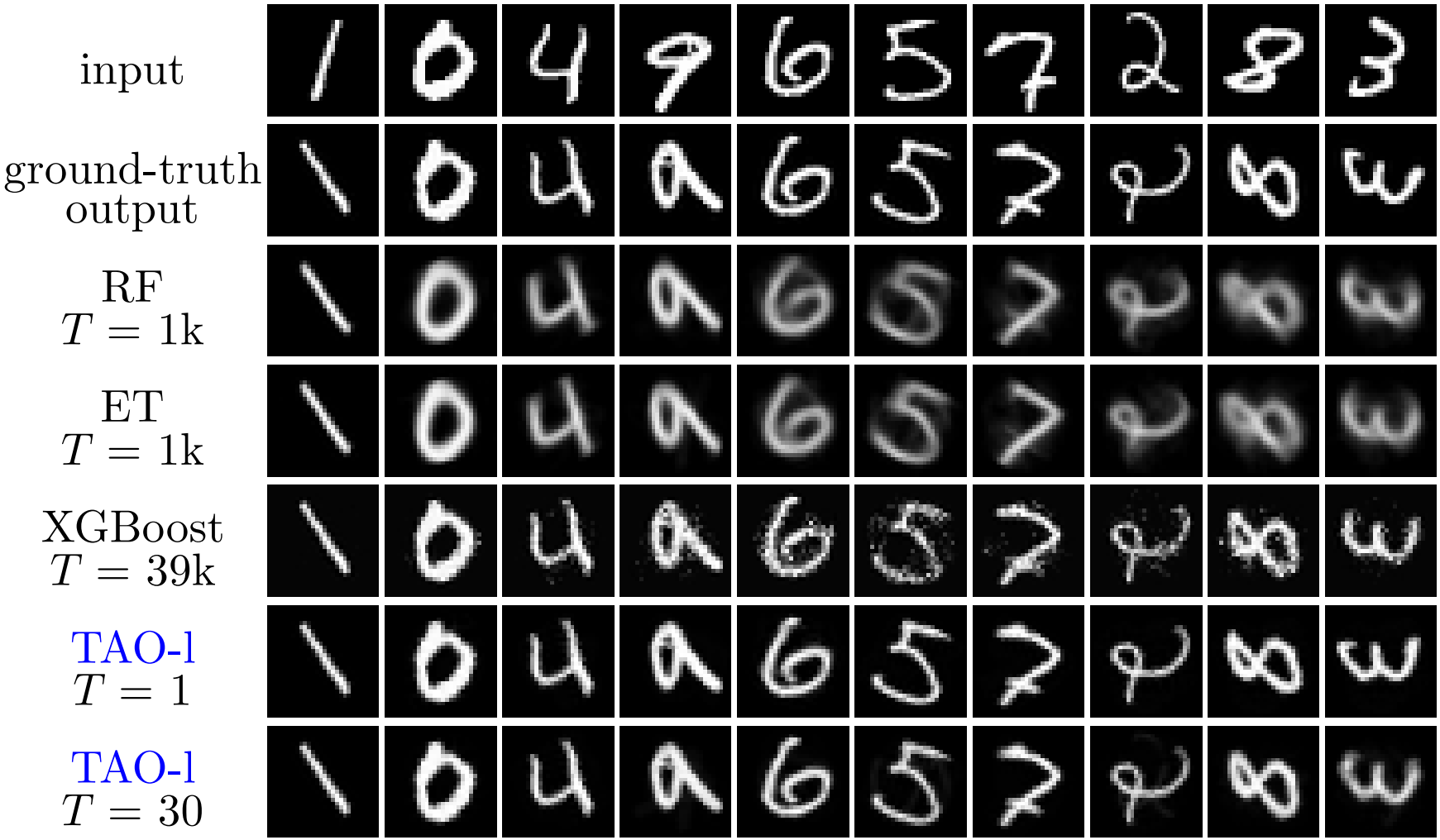
Experiments: MNIST digit rotation

Task: given an MNIST image, predict a class-dependent rotation of it (N=60k,D=784,K=784).

Forest	$E_{\text{test}} \times 10^{-2}$	#pars.	FLOPS	T	Δ
AdaBoost	>24 hours runtime			39k	25
CART	23.08±0.12	120k	28	1	28
RF	14.38±0.23	7.6M	(2 830)	100	39
RF	14.08±0.25	68M	(28k)	1k	40
ET	13.83±0.12	12M	(3 091)	100	35
TAO-c	13.76±0.09	9M	42k	30	29
ET	13.72±0.13	109M	(3 360)	1k	38
XGBoost	10.35±0.00	180M	(613k)	39k	25
TAO-l	9.63±0.17	288k	4 491	1	7
TAO-l	6.59±0.11	7.7M	126k	30	7

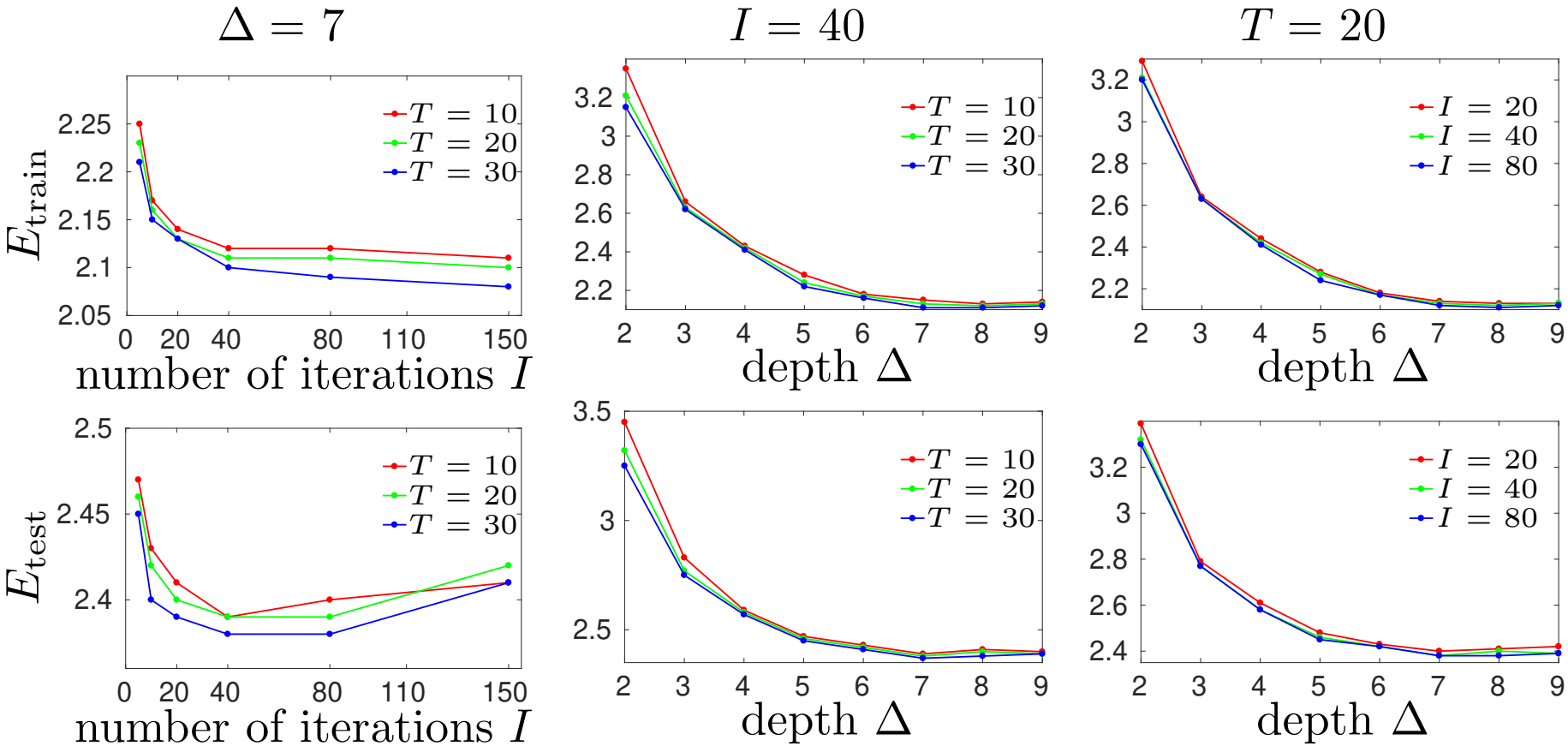
The improvement of TAO regression forests over other methods is clear, confirming our earlier results even more drastically.

Experiments: MNIST digit rotation (cont.)



Experiments: forest hyperparameters

Hyperparameters' exploration (cpuact dataset): depth Δ, # of TAO iterations I and # of trees T. Each column fixes one factor and varies the other two. In the paper, we also explore the effect of various diversity mechanisms.



Note how the forest can eventually overfit, which suggests that TAO is optimizing each tree well.

Conclusion

- Our hypothesis was that using more complex trees and a better tree optimization would produce more accurate forests. This was indeed the case, across a range of datasets we tried:
  - Our TAO regression forests 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.
- Their design in terms of hyperparameter tuning remains as simple as with random forests or 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, and suggests it could become the state-of-the-art in tree ensembles.

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