# Ensembles of Bagged TAO Trees Consistently Improve over Random Forests, AdaBoost and Gradient Boosting

Miguel Á. Carreira-Perpiñán and Arman Zharmagambetov

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

FODS, October 2020

# Introduction

We consider ensembles of trees (e.g. random forests, boosted trees) for classification. They have received much praise in the statistical and machine learning literature and widely used in many applications.

# Introduction

We consider ensembles of trees (e.g. random forests, boosted trees) for classification. They have received much praise in the statistical and machine learning literature and widely used in many applications.

Advantages:

- ability to achieve low bias and low variance by combining weakly correlated trees.
- usually easy to tune (even default parameters work well).
- reasonably fast to train and can be scaled to large datasets.

# Introduction

We consider ensembles of trees (e.g. random forests, boosted trees) for classification. They have received much praise in the statistical and machine learning literature and widely used in many applications.

Advantages:

- ability to achieve low bias and low variance by combining weakly correlated trees.
- usually easy to tune (even default parameters work well).
- reasonably fast to train and can be scaled to large datasets.

Disadvantages: the individual trees they learn are far from accurate. This is due to two main 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 correlated features.
- Standard tree learning algorithms, based on a greedy recursive tree growth (such as CART), are highly suboptimal.

# Our proposal

- We address both of these issues by:
  - using trees with more complex nodes (oblique, i.e., hyperplane)
  - using a better optimization algorithm to learn an individual tree.

  The resulting forests are smaller, shallower and more accurate. We support this claim by an extensive comparison with SoA forest classifiers across multiple datasets.

# Our proposal

- We address both of these issues by:
  - using trees with more complex nodes (oblique, i.e., hyperplane)
  - using a better optimization algorithm to learn an individual tree.

  The resulting forests are smaller, shallower and more accurate. We support this claim by an extensive comparison with SoA forest classifiers across multiple datasets.
- We build on top of recently proposed algorithm for learning decision 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 use TAO in combination with bagging to learn forests of oblique trees.
- There are exist few works that propose forests of more complex trees (Menze et al., 2000; Breiman, 2001; Frank & Kramer, 2004), but they improve marginally over conventional axis-aligned trees.

# Learning 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 $\Delta$), consider the following optimization problem over its parameters:

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

given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}$ is a set of parameters of all tree nodes. The loss function $L(\mathbf{y}, \mathbf{z})$ is 0/1 classification loss (although it is possible to use other losses, such as logistic or hinge). The regularization term $\phi_i$ (e.g. $\ell_1$ norm) penalizes the parameters $\boldsymbol{\theta}_i$ of each node.

# Learning a single tree with TAO: separability of nodes

Our TAO algorithm is based on 3 theorems: separability condition, reduced problem over a leaf, reduced problem over a decision node. Here, we briefly mention them.

## 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 $\boldsymbol{\Theta}_{\text{rest}}$.

# Learning a single tree with TAO: leaves

All leaves are non-descendants of each others. Therefore, we can optimize over each of them independently (according to separability condition).

## 2. Reduced problem over a leaf

Assume node $i$ is a leaf, then the optimization of $E(\boldsymbol{\Theta})$ over $\boldsymbol{\theta}_i$ can be equivalently rewritten 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 $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{C}$ (we use a constant or linear classifier) that produces the output class. 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. misclassification error).

# Learning a single tree with TAO: decision nodes

To optimize decision (internal) nodes, we again consider a set of non-descendant (e.g. all nodes at the same depth) nodes. 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 rewritten 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 at 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.

# Learning 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 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 $\phi_i$ is an $\ell_1$ norm, this requires solving an $\ell_1$-regularized logistic regression.

# Pseudocode for TAO

TAO repeatedly alternates optimizing over sets of nodes by training a (binary) classifier in the decision nodes and a (multiclass) classifier in the leaves, while monotonically decreasing the obj. function $E(\boldsymbol{\Theta})$.

**input** training set; initial tree $\mathbf{T}(\cdot;\boldsymbol{\Theta})$ of depth $\Delta$
$\mathcal{N}_0,\dots,\mathcal{N}_\Delta \leftarrow$ nodes at depth $0,\dots,\Delta$, respectively
$\mathcal{R}_1 \leftarrow \{1,\dots,N\}$
**repeat**
  **for** $d = 0$ **to** $\Delta$
    **parfor** $i \in \mathcal{N}_d$
      **if** $i$ is a leaf **then**
        $\boldsymbol{\theta}_i \leftarrow$ train classifier $\mathbf{g}_i$ 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 prediction is the majority vote of its trees' predictions.
- Although our TAO algorithm works more generally, our experiments use:
  - oblique trees ($\mathbf{f}_i$ is linear), which are more powerful than the axis-aligned trees
  - constant- and linear-predictor leaves ($\mathbf{g}_i$).
- We initialize each TAO tree ($\mathbf{T}$) from a complete tree of depth $\Delta$ and random node parameters.
- We train each tree with $\ell_1$ 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.
$T$ - number of trees, $\Delta$ - max depth.

| | Forest | $E_{\text{test}}$ (%) | $T$ | $\Delta$ |
|---|---|---|---|---|
| MNIST (60k,784,10) | CART | 12.11±0.04 | 1 | 50 |
| | TAO-c | 5.25±0.20 | 1 | 8 |
| | TAO-l | 5.07±0.13 | 1 | 5 |
| | AdaB | 2.96±0.05 | 1k | 30 |
| | RF | 2.84±0.06 | 1k | 48 |
| | ADF[54] | 2.71±0.10 | 100 | 25 |
| | TAO-c | 2.31±0.08 | 40 | 8 |
| | XGB | 2.17±0.00 | 10k | 30 |
| | rRF[48] | 2.05±0.02 | 100 | 25 |
| | TAO-l | 2.02±0.06 | 30 | 6 |

| | Forest | $E_{\text{test}}$ | $T$ | $\Delta$ |
|---|---|---|---|---|
| Char74k (67k,64,62) | CART | 32.14±0.15 | 1 | 55 |
| | TAO-c | 23.94±0.37 | 1 | 12 |
| | TAO-l | 20.82±0.31 | 1 | 4 |
| | XGB | 17.04±0.00 | 62k | 50 |
| | AdaB | 16.93±0.18 | 1k | 60 |
| | ADF[54] | 16.67±0.21 | 100 | 25 |
| | RF | 16.61±0.14 | 1k | 65 |
| | rRF[48] | 15.40±0.10 | 100 | 25 |
| | TAO-c | 15.19±0.18 | 30 | 12 |
| | TAO-l | 15.00±0.17 | 30 | 4 |

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

# Experiments: ImageNet

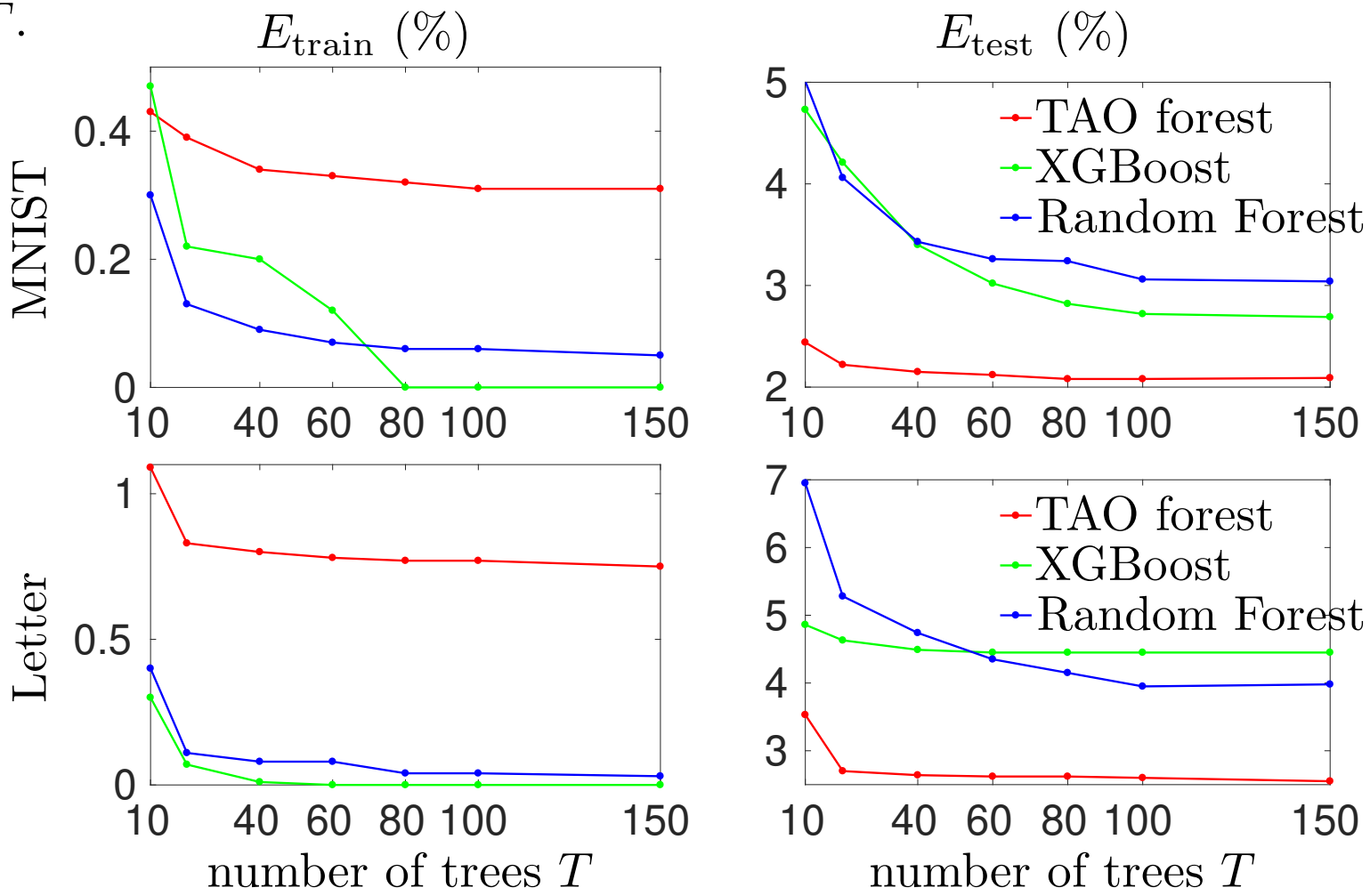
A subset of 64 classes of the full ImageNet, we use features extracted using VGG16 neural net. "Time" – training runtime.

| Forest | $E_{\text{test}}$ (%) | #params. | FLOPS | $T$ | $\Delta$ | Time |
|---|---|---|---|---|---|---|
| AdaBoost | >24 hours runtime | | | 100 | 50 | |
| CART | 44.62± 0.32 | 19 913 | (296) | 1 | 296 | 10.3 |
| TAO-c | 14.69± 0.18 | 3.7M | 16k | 1 | 13 | 409 |
| RF | 13.62± 0.32 | 2.6M | (22k) | 100 | 220 | 2.2 |
| RF | 12.51± 0.11 | 25M | (224k) | 1k | 220 | 8.4 |
| XGBoost | 12.51 | 596k | (81k) | 6.4k | 50 | 417 |
| XGBoost | 10.78 | 973k | (180k) | 64k | 50 | 493 |
| TAO-l | 10.45± 0.19 | 1.1M | 254k | 1 | 3 | 295 |
| TAO-c | 08.25± 0.11 | 100M | 0.5M | 30 | 13 | 516 |
| TAO-l | 08.19± 0.13 | 32M | 8.0M | 30 | 3 | 361 |

The improvement of TAO classification forests over other methods is clear. The runtime of TAO forests is similar to that of XGBoost.

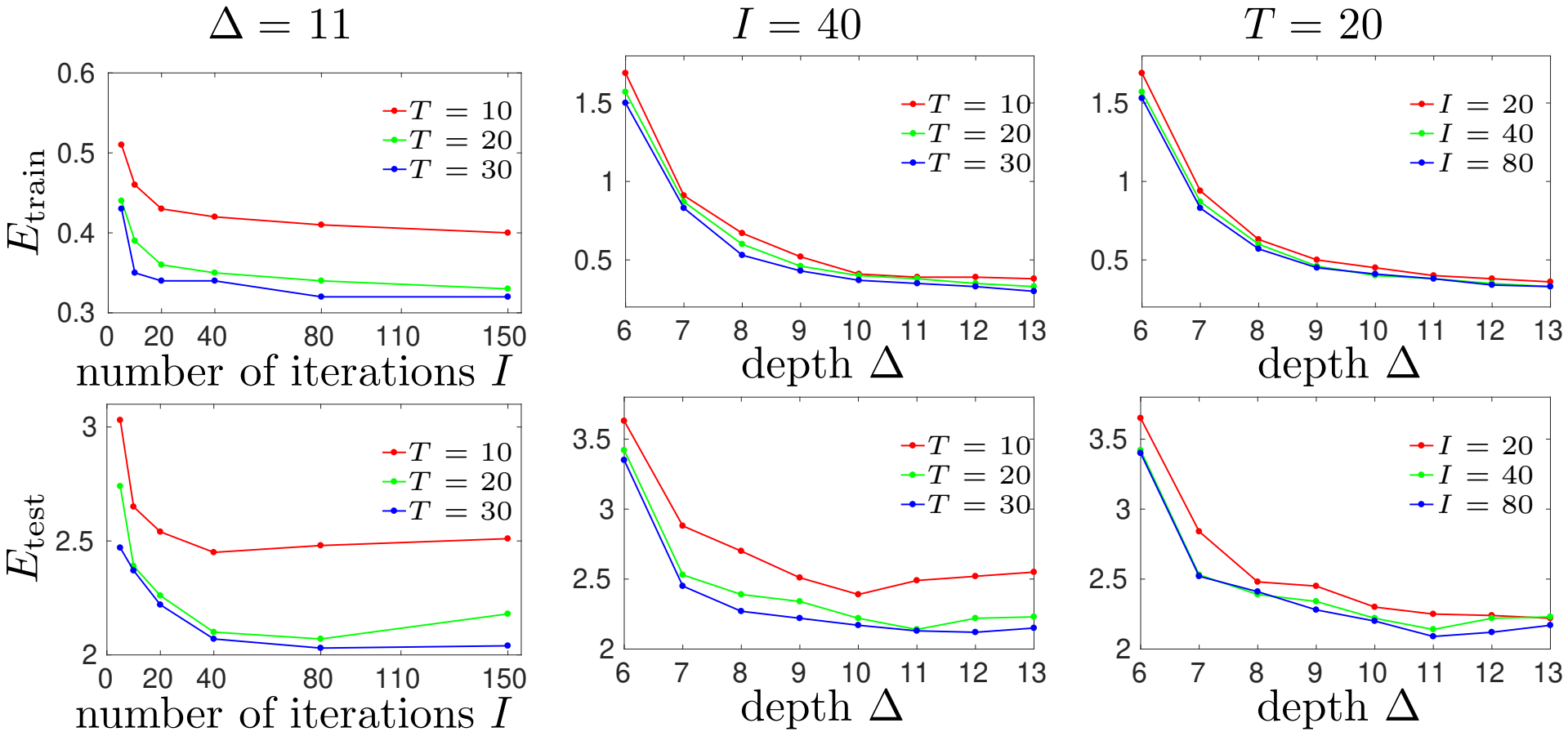
# Experiments: forest size

Training (left) and test error (right) on MNIST (top) and Letter (bottom) for TAO forests, XGBoost and Random Forests as a function of the number of trees $T$.



# Experiments: forest hyperparameters

Hyperparameters' exploration (cpuact dataset): depth $\Delta$, # 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 classification 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 regression (rather than classification) and with boosting (rather than bagging).