Improved Multiclass AdaBoost for Image Classification: the Role of Tree Optimization

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

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

IEEE ICIP, September 2021

Introduction

- Ensembles of decision trees (= forests) have found numerous applications in image processing and computer vision [2].
- 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 [3].
- Because of these, boosting algorithms usually need to induce K (= number of classes) such trees at each boosting step, which adds an extra overhead.
- And, to find a suitable splitting criterion for specific objective functions (as is the case with many boosting algorithms) is not straightforward with these greedy algorithms.

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) [1, 8], for a specific boosting framework and empirically evaluate its performance on image classification 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.MH

- In this work, we focus on AdaBoost.MH [6]:
    - One of the extensions of the original AdaBoost for multiclass/multilabel problems.
    - Has been empirically observed to be more dominant extensions of AdaBoost in terms of accuracy [9].
- Previous implementations of AdaBoost.MH used K trees at each boosting step similar to the one-vs-all strategy. In this work, instead, we use a single oblique tree at each step trained with TAO.
- The base learner in AdaBoost.MH must output a K dimensional vector.

AdaBoost.MH pseudocode

Algorithm 1: AdaBoost.MH with TAO trees

input training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$; number of trees T;

initial weights $\{w_{n,k} = \frac{1}{2N}, w_{n,K\backslash k} = \frac{1}{2N(K-1)}\}_{n=1,k=1}^{N,K}$;

for $t = 1$ *to* T **do**

  $\mathbf{T}_t \leftarrow$ train a TAO tree;

  obtain predictions: $\{\hat{\mathbf{y}}_n\}_{n=1}^N \leftarrow \mathbf{T}_t(\{\mathbf{x}_n\}_{n=1}^N)$;

  calculate the loss:

  $\hat{L} = \sum_{n=1}^N \sum_{k=1}^K w_{n,k} \cdot \exp(-y_{n,k} \cdot \hat{y}_{n,k})$

  update the weights: $w_{n,k} \leftarrow w_{n,k} \frac{\exp(-y_{n,k} \cdot \hat{y}_{n,k})}{\hat{L}}$

  for $n = 1, \ldots, N$ and $k = 1, \ldots, K$

end

return *Final classifier:* $\mathbf{F}(\mathbf{x}) = \sum_{t=1}^T \mathbf{T}_t(\mathbf{x})$

Objective function of the base learner

A loss per point of the base learner's objective function:

$$L(\mathbf{w}_n, \mathbf{y}_n, \mathbf{T}(\mathbf{x}_n)) = \sum_{k=1}^{K} w_{n,k} \cdot \exp(-y_{n,k} \cdot \mathbf{T}_k(\mathbf{x}_n)) \qquad (1)$$

- $\mathbf{T}(\cdot)$ is a base learner, in our case it is an oblique decision tree with constant leaf vectors.
- $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ is a training set with weights $\{w_{n,k}\}_{n=1,k=1}^{N,K}$ maintained by the boosting algorithm.

Optimizing a single tree with TAO for AdaBoost.MH

Given a tree structure $\mathbf{T}$ (e.g. a complete tree of depth Δ), TAO considers the following optimization problem over the tree parameters:

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

- $\mathcal{N}$ is the set of all nodes
- $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{N}}$ is a set of parameters of all tree nodes
- 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

Separability condition

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

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

In other words, the separability condition states that any set of non-descendant nodes of a tree can be optimized independently.

TAO tree for AdaBoost.MH: leaves

Optimization of a leaf

If i is a constant leaf vector, then there is a closed form solution of $E(\cdot)$ over its constant output vector $\mathbf{y}^*$ [6]:

$$y_k^* = 0.5 \cdot \log \frac{w_k^+ + \epsilon}{w_k^- + \epsilon}, \text{ for } k = 1, \dots, K \tag{2}$$

where (considering points n that reach the leaf i):

- w_k^+ is the sum of the weights for which $y_{n,k} = 1$
- w_n^- is the sum of the weights for which $y_{n,k} = -1$
- A small number ϵ is added for numerical stability.

TAO tree for AdaBoost.MH: decision nodes

Optimization of a decision node

If i is a decision node, the optimization of $E(\boldsymbol{\Theta})$ over $\boldsymbol{\theta}_i$ reduces to the following weighted binary classification problem:

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

- $\overline{L}$ is the 0/1 misclassification loss
- $\overline{y}_n \in \{\text{right,left}\}$ is a "pseudolabel" indicating the child which gives a lower value of E for input $\mathbf{x}_n$ under the current tree
- $f_i \in \{\text{right,left}\}$ is a linear thresholding function which sends the instance $\mathbf{x}_n$ to the corresponding child of i
- $\nu_n = |L(\mathbf{w}_n, \mathbf{y}_n, \mathbf{T}_{\text{left}}(\mathbf{x}_n)) - L(\mathbf{w}_n, \mathbf{y}_n, \mathbf{T}_{\text{right}}(\mathbf{x}_n))|$ is the absolute difference of losses incurred of sending $\mathbf{x}_n$ to the right or left child

Pseudocode for training a TAO tree

Algorithm 2: Learning a base classifier (tree) with TAO

input training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$;
initial tree $\mathbf{T}(\cdot; \mathbf{\Theta})$ of depth Δ;
Boosting weights $\{w_{n,k}\}_{n=1,k=1}^{N,K}$;

```
repeat
    for depth d = 0 to Δ do
        for i ∈ nodes at depth d do
            if i is a leaf then
                y_i ← fit a constant classifier at a leaf eq. (2);
            else
                θ_i ← fit a weighted binary classifier (eq. (3));
            end
        end
    end
until convergence occurs or max iteration;
return trained tree T
```

Experiments: standard benchmarks and algorithms

MH-TAO: AdaBoost.MH with TAO trees

MH-CART: AdaBoost.MH with CART trees

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

	Forest	E_{test}	T	Δ
MNIST	SAMME	2.96±0.05	1k	30
	RF	2.84±0.06	1k	48
	sNDF [4]	2.80±0.12	80	10
	MH-CART	2.73±0.00	200	7
	ADF [7]	2.71±0.10	100	25
	XGBoost	2.67±0.00	1k	8
	SAMME	2.28±0.02	1k	16
	rRF[5]	2.05±0.02	100	25
	MH-TAO	1.96±0.06	20	8
	XGBoost	1.94±0.03	10k	8
	MH-TAO	1.92±0.07	30	8
	MH-TAO	1.72±0.08	100	8

	Forest	E_{test}	T	Δ
Letter	XGBoost	4.30±0.00	2.6k	10
	RF	3.77±0.06	100	34
	ADF [7]	3.52±0.12	100	25
	RF	3.44±0.09	1k	36
	XGBoost	3.35±0.00	26k	6
	rRF[5]	2.98±0.15	100	25
	sNDF [4]	2.92±0.17	70	10
	SAMME	2.83±0.15	100	16
	SAMME	2.58±0.09	1k	16
	MH-CART	2.53±0.00	500	9
	MH-TAO	2.00±0.05	30	11
	MH-TAO	1.65±0.05	100	11

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

Comparison with other forests

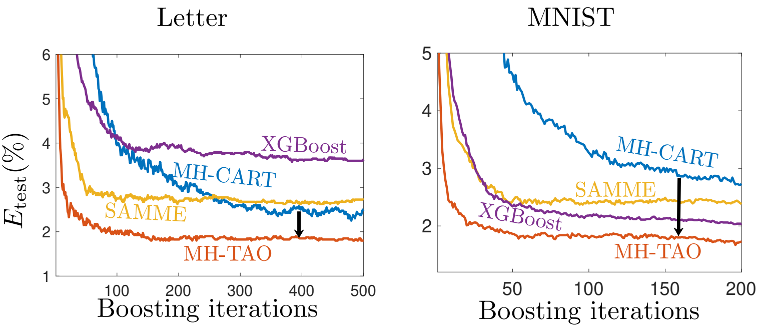

Figure: Comparison of different forest-based models on Letter and MNIST datasets as a function of the number of boosting iterations. MH-TAO and MH-CART refers to AdaBoost.MH with the corresponding base learners.

Conclusion

- Directly and non-greedily optimizing the base learner's objective function in AdaBoost.MH 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

[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] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag, 2013.

[3] 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.

[4] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Buló. Deep neural decision forests. In *Proc. 15th Int. Conf. Computer Vision (ICCV'15)*, pages 1467–1475, Santiago, Chile, Dec. 11–18 2015.

[5] S. Ren, X. Cao, Y. Wei, and J. Sun. Global refinement of random forest. In *Proc. of the 2015 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'15)*, pages 723–730, Boston, MA, June 7–12 2015.

[6] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, Dec. 1999.

[7] S. Schulter, P. Wohlhart, C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. Alternating decision forests. In *Proc. of the 2013 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'13)*, pages 508–515, Portland, OR, June 23–28 2013.

[8] 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.

[9] J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class AdaBoost. *Statistics and Its Interface*, 2(3):349–360, 2009.