IMPROVED MULTICLASS ADABOOST FOR IMAGE CLASSIFICATION: THE ROLE OF TREE OPTIMIZATION

Arman Zharmagambetov * *Magzhan Gabidolla* * *Miguel Á. Carreira-Perpiñán*

Dept. Computer Science & Engineering, University of California, Merced. Merced, CA, USA
Email: {azharmagambetov,mgabidolla,mcarreira-perpinan}@ucmerced.edu

ABSTRACT

Decision tree boosting is considered as an important and widely recognized method in image classification, despite dominance of the deep learning based approaches in this area. Provided with good image features, it can produce a powerful model with unique properties, such as strong predictive power, scalability, interpretability, etc. In this paper, we propose a novel tree boosting framework which capitalizes on the idea of using shallow, sparse and yet powerful oblique decision trees (trained with recently proposed *Tree Alternating Optimization* algorithm) as the base learners. We empirically show that the resulting model achieves better or comparable performance (both in terms of accuracy and model size) against established boosting algorithms such as gradient boosting or AdaBoost in number of benchmarks. Further, we show that such trees can directly and efficiently handle multiclass problems without using one-vs-all strategy employed by most of the practical boosting implementations.

Index Terms— image classification, adaboost, oblique trees, tree optimization

1. INTRODUCTION

Image classification is an important branch of pattern recognition and computer vision where machine learning techniques were able to show state-of-the-art performances. Particularly, deep learning has become extremely successful in recent years due to the representation learning capability (i.e. automatically extracting meaningful features) and possibility to efficiently train a complex architecture in end-to-end fashion. On the other hand, ensembles of decision trees (forests) are also powerful models which are widely used in many image processing applications [1], including object detection [2, 3], visual tracking [4], shape recognition [5]. Their success is attributed to the number of practical advantageous, such as strong generalization property, scalability to large data, fast inference, etc. Some examples of forests: random forests [6] train each tree independently on a different data sample and on a different subset of features, boosted trees [7, 8] sequentially train trees on the reweighted versions of data. Among others, AdaBoost [7] is often referred to as among the best out-of-the-box classifier. It has been well studied in machine learning and statistical literature, and has been shown to be successful in many applications [9].

The main idea behind AdaBoost is sequentially training and combining a collection of base (weak) learners to produce a strong model. This is a stepwise procedure where each consecutive step involves tranining a learner which is encouraged to improve overall model performance. Decision trees are known to be a common choice for the base learner [10, 11], though one can use any model satisfying the weak-learning condition (i.e. better than a random guess). These trees are usually trained using top-down greedy algorithms such as CART [12] which have a topical issue of being highly suboptimal [11]. Moreover, for multiclass problems with K classes, modern practical boosting frameworks (e.g. [13, 14]) fit K trees at each additive step (instead of one) implementing one-versus-all strategy which adds additional overhead to the training time and model size. One possible explanation for using K trees is that in order to use a single tree it has to be grown large enough to satisfy the weak-learning criterion which yields to overfitting, and, when boosted, results to poor performance [15]. Throughout this paper, we define one boosting step (or iteration) as fitting K class classifier (whether it is a single tree or K trees) and the corresponding update of the weights.

In this paper, we propose to use a better optimized shallow and sparse oblique decision tree, trained with the recently proposed Tree Alternating Optimization (*TAO*) algorithm [16, 17]), as the base learner. They achieve both low bias and low variance yielding a high predictive power, at the same time being fast (at training and inference). In combination with boosting, these trees achieve further improvement in terms of classification accuracy leveraging the power of ensemble learning. Moreover, it is now sufficient to train a single tree at each boosting step without necessity of growing a large tree. In section 4, we demonstrate that our boosted TAO trees not only show competitive performance, but also can achieve decent accuracy on well known image classification benchmarks.

2. BOOSTING FRAMEWORK

Although TAO trees can be embedded into any existing framework, we use AdaBoost [7] due to its simplicity and high predictive power. Originally developed for the binary classification task, AdaBoost has several multiclass extensions [18]. AdaBoost.MH is one of them which was specifically designed for multiclass problems and we use it to validate our method.

Consider a regular supervised classification task with the training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N \subset \mathbb{R}^D \times \mathbb{R}^K$ of D-dimensional real-valued input instances and their labels (in K classes). Here, we represent the ground truth label ($\mathbf{y}_n$) as a vector in $\mathbb{R}^K$, where the correct class is denoted as $y_{n,k} = 1$ (for $k \in \{1, \dots, K\}$) and all other classes as $y_{n,K\setminus k} = -1$. As in other AdaBoost variations, AdaBoost.MH trains an ensemble of base classifiers (decision trees in our case) in additive manner (i.e. stepwise, not in parallel) and each step involves minimizing the following regularized loss function:

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

where $\mathbf{w}_n$ is the weights used by the boosting algorithm; $\mathbf{T}(\mathbf{x}_n)$ is

*equal contribution

the vector valued output of a tree with parameters $\mathbf{\Theta} = \{\boldsymbol{\theta}_i\}_{i\in\mathcal{N}}$ and $\mathcal{N}$ is the set of all tree nodes. The regularization term ϕ_i (with hyperparameter $\alpha \geq 0$) penalizes the parameters $\boldsymbol{\theta}_i$ of each node. Notice that the original AdaBoost formulation does not introduce any penalty and it is our modification. In this paper, we choose ϕ_i to be ℓ_1 norm in order to achieve sparsity. Further, in AdaBoost.MH, the loss function $L(\cdot)$ is the weighted multiclass exponential loss defined below:

$$L(\mathbf{w}_n, \mathbf{y}_n, \mathbf{T}(\mathbf{x}_n)) = \sum_{k=1}^{K} w_{n,k} \cdot \exp(-y_{n,k} \cdot \mathbf{T}_k(\mathbf{x}_n)) \qquad (2)$$

where $w_{n,k}$ is the weight per class and per training instance. Minimizing (1) at each step with the loss function defined in (2) leads to the pseudocode of our modified version of the AdaBoost.MH provided in Algorithm 1. At each step t, a base classifier is trained with the current weights using the TAO algorithm (see section 3). Then the weights are updated according to the formula shown in the algorithm. The final prediction of the model is given as a sum of T base classifiers (one might need to take $\arg\max$ to obtain a predicted class).

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

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

- $\mathbf{T}_t \leftarrow$ train a TAO tree on the training set with the current weights and loss function (1) (see Algorithm 2)
- obtain predictions: $\{\hat{\mathbf{y}}_n\}_{n=1}^{N} \leftarrow \mathbf{T}_t(\{\mathbf{x}_n\}_{n=1}^{N})$;
- calculate the loss (eq. (1) without regularization): $\hat{L} = \sum_{n=1}^{N}\sum_{k=1}^{K} w_{n,k} \cdot \exp(-y_{n,k} \cdot \hat{y}_{n,k})$
- 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:* $\mathbf{F}(\mathbf{x}) = \sum_{t=1}^{T} \mathbf{T}_t(\mathbf{x})$

3. THE TAO TREES

The choice of the base learner is crucial in boosting. Though the framework is quite generic, decision trees have long thought to be the model of choice. That said, boosting of other types of learners (such as kernel SVMs, neural networks) have been studied before [19]. However, their superiority, in terms of performance, has not been observed empirically [20, sec. 2.1]. Decision trees are traditionally trained using recursive top-down algorithms such as CART [12] or C4.5 [21] which have several limitations: greedy nature of the algorithm yields suboptimal trees [11], they do not directly optimize a desired loss function, internal nodes of a tree typically use a single feature to make a split (i.e. axis-aligned or univariate split).

In this work, we propose to use a better optimized shallow and sparse oblique decision tree with constant leaves as a base learner. Here, oblique means that a split is done by the linear thresholding function of the form: send to the right child if $f(\mathbf{x}; \boldsymbol{\theta}, b) = \boldsymbol{\theta}^T\mathbf{x} - b \geq 0$ and to the left child otherwise, where $\boldsymbol{\theta} \in \mathbb{R}^D, b \in \mathbb{R}$ (bias term) are node parameters and $\mathbf{x}$ is the input feature vector. A linear split clearly takes advantage of the better feature utilization (versus univariate split) which typically results to the more powerful model. Unfortunately, previously proposed methods on boosting oblique trees [22, 23] did not produce expected improvements since the training of the individual tree was done based on the greedy approximate impurity minimization [12, 24] which can generate suboptimal trees, as we have discussed earlier. However, the recently proposed TAO algorithm [16, 17] can find good approximate optima of the decision tree optimization problem and shows dramatic improvement in training oblique trees [25]. Moreover, it has been already successfully applied with bagging [26, 27] and training more complex decision trees (e.g. neural trees) [28].

Instead of growing a tree, TAO takes as input an initial tree with predefined structure and parameters, and minimizes the loss function defined in eq. (1) jointly over the parameters $(\boldsymbol{\theta}_i, b_i)$ of all nodes i. Throughout this paper, we initialize the TAO tree (at each boosting iteration) from a complete tree of depth Δ and random node parameters. Then TAO solves the optimization problem in eq. (1) by fixing one part of the tree and training another part which involves optimizing over a set of independent nodes. Here, "independent" means any set of non-descendant nodes (e.g. at the same depth). Consider any pair of such independent nodes i and j. Since an input instance follows unique root-to-leaf path, i and j receive disjoint set of training instances. Furthermore, we fix the parameters of all the remaining nodes (including descendants and ancestors). Then according to the *separability condition* [16, 17], E from eq. (1) *separates* over the parameters of nodes i and j:

$$E(\mathbf{\Theta}) = E_i(\boldsymbol{\theta}_i, b_i) + E_j(\boldsymbol{\theta}_j, b_j) + E_{\text{rest}}(\mathbf{\Theta}_{\text{rest}}) \qquad (3)$$

where $\mathbf{\Theta}_{\text{rest}}$ is the parameters of the fixed part of a tree. In other words, we can minimize over the parameters of each node separately. The solution for the individual node minimization problem depends on its type:

Leaves In eq. (2), we need to obtain a vector valued output (in $\mathbb{R}^K$). It turns out that, for the constant leaves, the solution can be computed exactly [29] and it is equal to:

$$y_k^* = 0.5 \cdot \log \frac{w_k^+ + \epsilon}{w_k^- + \epsilon}, \text{ for } k = 1, \ldots, K \qquad (4)$$

where w_k^+ is the sum of the weights for which $y_{n,k} = 1$ and w_n^- is the sum of the weights for which $y_{n,k} = -1$, considering training instances n that reach the leaf i. A small number ϵ is added for numerical stability. We emphasize that we do not apply one-versus-all strategy at each boosting step but rather *handle multiple classes directly* at each leaf. This is possible with TAO trees since it achieves much better performance (without overfitting) with shallow trees satisfying the weak-learning condition.

Internal nodes There are only two possible outcomes: sending an input instance to the *left* or *right*. Therefore, it can be shown that the problem in eq. (1) reduces to the following weighted binary classification problem on the training instances that reach the node i:

$$\min_{\boldsymbol{\theta}_i} \sum_{n\in\mathcal{R}_i} \nu_n \overline{L}(\overline{y}_n, f_i(\mathbf{x}_n; \boldsymbol{\theta}_i, b_i)) + \lambda\phi_i(\boldsymbol{\theta}_i) \qquad (5)$$

where $\overline{L}$ 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; similarly, $f_i \in \{\text{right,left}\}$ is a linear thresholding function discussed above which sends the instance $\mathbf{x}_n$ to the corresponding child of i. Finally, $\nu_n = |L(\mathbf{w}_n, \mathbf{y}_n, \mathbf{T}_{\text{left}}(\mathbf{x}_n)) - L(\mathbf{w}_n, \mathbf{y}_n, \mathbf{T}_{\text{right}}(\mathbf{x}_n))|$ is the absolute difference of losses incurred of sending $\mathbf{x}_n$ to the right or left child. Here, $L(\cdot)$ comes from eq. (2) and it incorporates the currents

weights $\mathbf{w}_n$ coming from the previous boosting iteration. It is worth mentioning that TAO can *handle these weights directly*, whereas most of the boosting implementations use various sampling techniques. Since optimizing the misclassification loss in eq. (5) is an NP-hard problem, we further approximate it with a convex surrogate (e.g. logistic) loss and solve it efficiently using LIBLINEAR [30].

As a result, we obtain the Algorithm 2 presented below. It repeatedly updates nodes' parameters depth-wise. Additionally, it is possible to minimize them in parallel at each depth (this follows from the separability condition). One pass over all nodes defines a TAO iteration and one can continue to iterate until convergence occurs. In practice, we stop when the maximum number of iterations is reached.

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

input training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$;
initial tree $\mathbf{T}(\cdot; \boldsymbol{\Theta})$ of depth Δ;
Boosting weights $\{w_{n,k}\}_{n=1,k=1}^{N,K}$;
for *depth* $d = 0$ *to* Δ **do**
  for $i \in$ *nodes at depth* d **do**
    if i *is a leaf* **then**
      $\mathbf{y}_i \leftarrow$ fit a constant classifier at a leaf eq. (4) with the current boosting weights;
    else
      $\boldsymbol{\theta}_i \leftarrow$ fit a weighted binary classifier (eq. (5)) with the current boosting weights;
    end
  end
end
return *trained tree* $\mathbf{T}$

4. EXPERIMENTS

Dataset	N_{train}	N_{test}	D	K
Letter	16 000	4 000	16	26
MNIST	60 000	10 000	784	10
ImageNet subset	62 855	12 800	8192	64

Table 1. Datasets used in our experiments: number of points for training (N_{train}) and test (N_{test}) sets, feature vector dimensionality D, number of classes K.

We perform extensive evaluations of our proposed method on image classification problems. First, we validate our boosting framework on two mid-sized popular character recognition tasks: MNIST and Letter. For MNIST, we directly use grayscale pixel values (in [0,1]), whereas, for Letter, the features are image descriptors such as edge counts and statistical moments. We provide a basic dataset characteristics, such as its size (N), input dimensionality (D) and number of classes (K) in table 1. Training/test partition is given for MNIST, and we use the last 4 000 samples as the test set for Letter. Next, we perform experiments on larger scale dataset - Imagenet [33] (a subset of arbitrarily selected 64 classes, given in suppl. mat.). Unfortunately, applying boosted trees directly on image pixels does not produce a desired performance. Therefore, we use as features the output of the last convolutional layer from a pretrained VGG16 deep net. Datasets used in our experiments and their descriptions can be found in the *supplementary material* for this paper.

	Forest	E_{test} (%)	#pars.	FLOPS	T	Δ
MNIST	RF	3.05±0.06	1M	(3 482)	100	46
	SAMME	2.96±0.05	6M	(29 489)	1 000	30
	RF	2.84±0.06	10M	(34 507)	1 000	48
	sNDF [31]	2.80±0.12	22M	(22M)	80	10
	XGBoost	2.73±0.00	390k	(16 812)	1 000	30
	MH-CART	2.73±0.00	307k	(1 400)	200	7
	ADF [3]	2.71±0.10	3.6M	(2 500)	100	25
	XGBoost	2.67±0.00	324k	(8 000)	1 000	8
	SAMME	2.28±0.02	13.3M	(16 000)	1 000	16
	XGBoost	2.17±0.00	540k	(57 385)	10 000	30
	rRF[32]	2.05±0.02	(160k)	(2 500)	100	25
	MH TAO	1.96±0.06	837k	54 041	20	8
	XGBoost	1.94±0.00	615k	(51 873)	10000	8
	MH-TAO	1.92±0.07	2.3M	93 523	30	8
	MH-TAO	1.72±0.08	7.9M	312 904	100	8
Letter	XGBoost	4.30±0.00	353k	(25 277)	2 600	10
	RF	3.77±0.06	419k	(2 836)	100	34
	ADF [3]	3.52±0.12	(1M)	(2 500)	100	25
	RF	3.44±0.09	4.2k	(27 820)	1000	36
	XGBoost	3.35±0.00	768k	(118k)	26k	6
	rRF[32]	2.98±0.15	(180k)	(2 500)	100	25
	sNDF [31]	2.92±0.17	2.4M	(2.4M)	70	10
	SAMME	2.83±0.15	651k	(1 600)	100	16
	SAMME	2.58±0.09	6.7M	(16 000)	1 000	16
	MH-CART	2.53±0.00	524k	(4 500)	500	9
	MH-TAO	2.00±0.05	778k	4 998	30	11
	MH-TAO	1.65±0.05	2.7M	16 815	100	11
ImageNet subset	MH-CART	>8 days runtime			100	9
	MH-CART	25.07	9400	(500)	100	5
	RF	13.62±0.32	2.5M	(23k)	100	220
	RF	12.67±0.13	12.7M	(109k)	500	218
	RF	12.51±0.11	25.4M	(224k)	1000	220
	XGBoost	12.51±0.00	596k	(81k)	6400	50
	XGBoost	11.01±0.00	782k	(124k)	32000	50
	XGBoost	10.78±0.00	973k	(181k)	64000	50
	MH-TAO	10.65±0.05	3.4M	105k	30	12

Table 2. Comparison of different forests (sorted by decreasing test error). MH-TAO – our method, SAMME – AdaBoost.SAMME with a CART base learner, RF – Random Forest, MH-CART – AdaBoost.MH version with CART style trees as base learners. We report the test error (avg±stdev over 5 repeats), number of parameters and inference FLOPS (numbers in parentheses are estimates), number of trees T and maximum depth of the forest Δ. Arrows indicate the improvement of MH-TAO over MH-CART. Note: only 1 iteration of MH-CART was performed on ImageNet subset due to long training time.

We compare our boosted TAO trees (denoted as *MH-TAO*) with the state-of-the-art tree ensembling algorithms: Random Forests [6] (denoted as RF, implemented in scikit-learn [13]), AdaBoost.SAMME [34] with a CART base learner (denoted as SAMME, implemented in scikit-learn [13]); and XGBoost [14] (note that it trains K trees at each boosting iteration). Another important baseline is AdaBoost.MH with traditional axis-aligned trees (denoted as MH-CART) since here we can directly compare the effects of different base learners within the same framework. We use `multiboost` framework [18] available online which implements AdaBoost.MH version with a tree learner similar to CART (called Hamming trees). Hyperparameter (e.g. T ensemble size, depth Δ, learning rate, etc.) optimization for all these methods is carried out by grid-search on a validation set. We also report the published results of some of the recent tree based ensembles ([3, 31, 32]) from

the corresponding papers on the same datasets.

Specifically for the TAO algorithm, we implemented it in C++ with parallel processing using OpenMP. For each dataset, we tune the regularization penalty α and the number of TAO iterations. We initialize a TAO tree from a complete tree of depth Δ and random node parameters at each boosting iteration. We use ℓ_1 norm as a regularizer (ϕ in eq. (1)) in order to achieve sparsity. Therefore, the node optimization reduces into ℓ_1-regularized logistic regression (see section 3) which we solve using LIBLINEAR [30].

4.1. Results

Table 2 reports the results. We sort all methods by the decreasing test error for each dataset. First, the comparison with the direct baseline (MH-CART) convincingly demonstrates superiority of our proposed method and it is consistent throughout all datasets. MH-TAO was able to achieve significantly less test error using much smaller or comparable number of trees which additionally have smaller depths. It is worth to mention that [18] reports better results for MH-CART on MNIST and Letter but it was achieved using enormous ensemble size (10^5 trees) which we were unable to replicate due to long training time. As for the other baselines, experiments show similar outcomes: MH-TAO beats all other forests on all three datasets we tried: MNIST, Letter and Imagenet subset. It is quite surprising given MH-TAO has a less number of trees (T) and the maximum depth (Δ) is much smaller compared to other methods. For instance, we were able to achieve a decent performance using only 30 boosting iterations (equivalent to the number of trees). However, keep in mind that oblique trees generally have more parameters since each node stores a vector in $\mathbb{R}^D$. Therefore, we additionally show the actual or estimated number of parameters (by summing parameters of all nodes) and inference time as a number of floating point operations (by finding the average flops in all root-to-leaf paths, see table 2: columns "#pars." and "FLOPS", respectively). Results show that MH-TAO generally has comparable model sizes. For example, it outperforms all other methods on MNIST and Letter using only 20-30 trees which have similar (or fewer) number of parameters and inference time. Increasing the number of boosting iterations can further improve the performance (e.g. 100) but requires more parameters. This was possible thanks to the sparsity penalty that we apply since it helps to eliminate redundant features but preserves the accuracy. On Imagenet subset, roughly speaking, the number of parameters increases as XGBoost $<$ MH-TAO $\lesssim$ RF. This is partly due to the high input vector dimensionality, though the depth is still smaller compared to other methods. Note that even though MH-CART has smaller model size, it performs significantly worse compared to other forests.

Figure 1 shows the comparison of the test error as a function of the number of trees for several boosting methods: MH-TAO, MH-CART, SAMME and XGBoost. Results clearly indicate that MH-TAO considerably outperforms all other methods achieving the best test error for the fixed number of trees (T). Moreover, it starts to improve drastically within the first several iteration. In terms of runtime, training MH-TAO, as in any other method, depends on a scale of the problem and choice of hyperparameters: dataset size, maximum depth, number of iterations, etc. Non-greedy nature of the TAO algorithm and stepwise learning in boosting definitely add some overhead to the training time. That said, we claim that MH-TAO has a manageable training time. For example, 100 boosted trees on MNIST were trained in 228 minutes and the same number of iterations on Letter took 8.7 minutes. As a comparison, it took about 928 minutes for MH-CART, 25 minutes for SAMME, 6.5 minutes for XGBoost to complete 100 boosting iterations on MNIST. Similarly, training a model on Letter (100 boosting iterations) took about 12 minutes for MH-CART, 8.0 seconds for SAMME, 8.8 seconds for XGBoost. It is true that XGBoost and SAMME have faster training times for the same number of iterations on the given benchmarks. However, keep in mind that our current implementation is still in development phase and can be improved. Moreover, the higher runtime of MH-TAO is justified by the low test error it achieves.

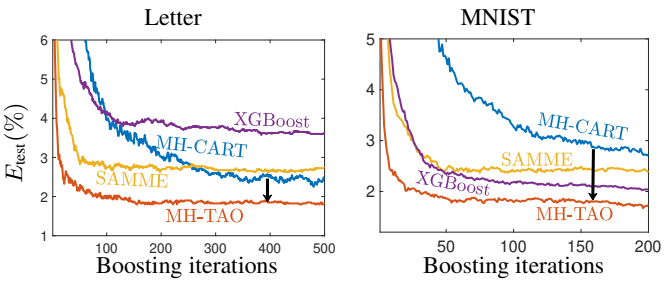

Fig. 1. Comparison of different boosting methods as a function of the number of boosting iterations. MH-TAO and MH-CART refers to AdaBoost.MH with the corresponding base learners. SAMME refers to AdaBoost.SAMME with traditional CART tree as a base learner. Arrows indicate the improvement of MH-TAO over MH-CART. Train error is not shown since it quickly approaches zero.

5. CONCLUSION

In this paper we propose a new boosting framework which combines AdaBoost.MH with the more powerful and better optimized (using TAO) oblique trees. Our idea builds on the hypothesis that commonly used base learners such as axis-aligned trees, trained with CART-style algorithms, are suboptimal, and thus, replacing them opens a possibility for an improvement. For this to succeed we rely on the TAO algorithm, which leverages the power of optimization in tree learning, and we train oblique trees using it. Indeed, on several image recognition tasks, we demonstrated that the boosted TAO trees outperform or achieve comparable performance not only in terms of accuracy but also in terms of model size and inference time. Methodologically, the proposed method directly fits multiclass base learner without using one-vs-all strategy which helps to avoid additional overhead. To the best of our knowledge, this is the first successful application of the boosted oblique trees which shows a significant improvement over the boosted traditional trees. Moreover, the final algorithm is straightforward to implement and has a manageable training time which lead to the immediate practical applicability in various fields, such as image classification, signal processing, data mining, etc. The main directions of our future work include: merging TAO trees with other more recent boosting algorithms (e.g. XGBoost, LightGBM) and extending our framework to handle multi-label problems. In a separate paper, TAO boosted trees have shown considerable improvement in regression problems [35].

Acknowledgments. Work supported by NSF award IIS–2007147.

6. REFERENCES

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

[2] Paul Viola and Michael J. Jones, "Robust real-time face detection," *Int. J. Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[3] Samuel Schulter, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter M. Roth, and Horst Bischof, "Alternating decision forests," in *CVPR'13*, Portland, OR, June 23–28 2013, pp. 508–515.

[4] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends in Computer Graphics and Vision*, vol. 7, no. 2–3, pp. 81–227, 2012.

[5] Yali Amit and Donald Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, Oct. 1997.

[6] Leo Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[7] Robert E. Schapire and Yoav Freund, *Boosting. Foundations and Algorithms*, Adaptive Computation and Machine Learning Series. MIT Press, 2012.

[8] Jerome H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[9] Rich Caruana and Alexandru Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *ICML'06*, Pittsburgh, PA, June 25–29 2006, pp. 161–168.

[10] Ludmila I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, John Wiley & Sons, second edition, 2014.

[11] Trevor J. Hastie, Robert J. Tibshirani, and Jerome H. Friedman, *The Elements of Statistical Learning—Data Mining, Inference and Prediction*, Springer Series in Statistics. Springer-Verlag, second edition, 2009.

[12] Leo J. Breiman, Jerome H. Friedman, R. A. Olshen, and Charles J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, Calif., 1984.

[13] Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011, Available online at `https://scikit-learn.org`.

[14] Tianqi Chen and Carlos Guestrin, "XGBoost: A scalable tree boosting system," in *SIGKDD 2016*, San Francisco, CA, Aug. 13–17 2016, pp. 785–794.

[15] B. Kégl, "The Return of AdaBoost.MH: Multi-Class Hamming Trees," arXiv:1312.6086, Dec. 2013.

[16] Miguel Á. Carreira-Perpiñán and Pooya Tavallali, "Alternating optimization of decision trees, with application to learning sparse oblique trees," in *Advances in Neural Information Processing Systems (NeurIPS'18)*.

[17] Miguel Á. Carreira-Perpiñán, "The Tree Alternating Optimization (TAO) algorithm: A new way to learn decision trees and tree-based models," arXiv, 2021.

[18] Balázs Kégl and Róbert Busa-Fekete, "Boosting products of base classifiers," in *ICML'09*, Montreal, Canada, June 14–18 2009, pp. 497–504.

[19] Leo J. Breiman, "Bias, variance, and arcing classifiers," Tech. Rep. 460, Dept. of Statistics, University of California, 1996.

[20] Zhi-Hua Zhou, *Ensemble Methods: Foundations and Algorithms*, Chapman & Hall/CRC Machine Learning and Pattern Recognition Series. CRC Publishers, 2012.

[21] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[22] C. Yu and D. B. Skillicorn, "Parallelizing boosting and bagging," Tech. Rep. 2001–442, Dept. of Computing and Information Science, Queens University, Feb. 2001.

[23] Claudia Henry, Richard Nock, and Frank Nielsen, "Real boosting *a la Carte* with an application to boosting oblique decision tree," in *Proc. of the 20th Int. Joint Conf. Artificial Intelligence (IJCAI'07)*, Hyderabad, India, Jan. 6–12 2007, pp. 842–847.

[24] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artificial Intelligence Research*, vol. 2, pp. 1–32, 1994.

[25] Arman Zharmagambetov, Suryabhan Singh Hada, Magzhan Gabidolla, and Miguel Á. Carreira-Perpiñán, "Non-greedy algorithms for decision tree optimization: an experimental comparison," in *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, July 18–22 2021.

[26] Arman Zharmagambetov and Miguel Á. Carreira-Perpiñán, "Smaller, more accurate regression forests using tree alternating optimization," in *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*.

[27] Miguel Á. Carreira-Perpiñán and Arman Zharmagambetov, "Ensembles of bagged TAO trees consistently improve over random forests, AdaBoost and gradient boosting," in *Proc. of the 2020 ACM-IMS Foundations of Data Science Conference (FODS 2020)*, Seattle, WA, Oct. 19–20 2020, pp. 35–46.

[28] Arman Zharmagambetov and Miguel Á. Carreira-Perpiñán, "Learning a tree of neural nets," in *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'21)*, Toronto, Canada, June 6–11 2021, pp. 3140–3144.

[29] Robert E. Schapire and Yoram Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, pp. 297–336, Dec. 1999.

[30] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "LIBLINEAR: A library for large linear classification," *J. Machine Learning Research*, vol. 9, pp. 1871–1874, Aug. 2008.

[31] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Buló, "Deep neural decision forests," in *Proc. 15th Int. Conf. Computer Vision (ICCV'15)*, Santiago, Chile, Dec. 11–18 2015, pp. 1467–1475.

[32] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun, "Global refinement of random forest," in *CVPR'15*, Boston, MA, June 7–12 2015, pp. 723–730.

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR'09*, Miami, FL, June 20–26 2009, pp. 248–255.

[34] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie, "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[35] Arman Zharmagambetov, Magzhan Gabidolla, and Miguel Á. Carreira-Perpiñán, "Improved boosted regression forests through non-greedy tree optimization," in *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, July 18–22 2021.