Pushing the Envelope of Gradient Boosting Forests via Globally-Optimized Oblique Trees

Magzhan Gabidolla and **Miguel Á. Carreira-Perpiñán**, Dept. Computer Science & Engineering, UC Merced

1 Abstract

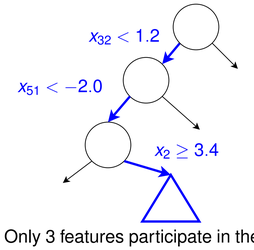
Ensemble methods based on decision trees, such as Random Forests or boosted forests, have long been established as some of the most powerful, off-the-shelf machine learning models, and have been widely used in computer vision and other areas. In recent years, a specific form of boosting, gradient boosting (GB), has gained prominence. This is partly because of highly optimized implementations such as XGBoost or LightGBM, which incorporate many clever modifications and heuristics. However, one gaping hole remains unexplored in GB: the construction of individual trees. To date, all successful GB versions use axis-aligned trees trained in a suboptimal way via greedy recursive partitioning. We address this gap by using a more powerful type of trees (having hyperplane splits) and an algorithm that can optimize, globally over all the tree parameters, the objective function that GB dictates. We show, in several benchmarks of image and other data types, that GB forests of these stronger, well-optimized trees consistently exceed the test accuracy of axis-aligned forests from XGBoost, LightGBM and other strong baselines. Further, this happens using many fewer trees and sometimes even fewer parameters overall.

Work supported by NSF award IIS–2007147

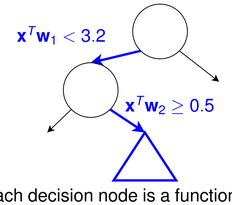
2 Gradient Boosting (GB) Forests

- Ensembles of decision trees have long been established as some of the most powerful, off-the-shelf machine learning models.
- In recent years, one type of forest, Gradient Boosting (GB), has gained prominence due to their:
  - Strong empirical performance on many problems
  - The development of extremely efficient implementations such as XGBoost or LightGBM.
- They typically require little effort on hyperparameter tuning and are thus considered "off-the-shelf".
- Given the tremendous effort put on the development and refinement of the popular GB toolkits, how can we further improve GB forests?

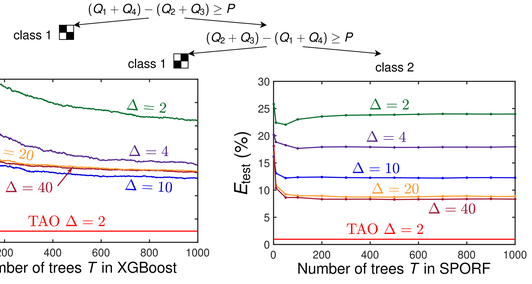
3 Modeling high-order feature interactions: axis-aligned trees vs oblique trees



Only 3 features participate in the routing function of the above leaf. Maximum order of feature interactions is limited by the depth Δ in axis-aligned trees.



Each decision node is a function of all the features. Their non-linear combination is a much complex order-D interaction. For modeling complex functions, a forest of oblique trees should achieve higher accuracy and require fewer trees.



Consider a synthetic binary classification of MNIST digit images where class 1 satisfies that $(Q_1 + Q_4) - (Q_2 + Q_3) \geq P$ or $(Q_2 + Q_3) - (Q_1 + Q_4) \geq P$, where the 28×28 pixel image is split into 4 quadrants $\left[\begin{smallmatrix}1&2\\3&4\end{smallmatrix}\right]$ and Q_i is the sum of the $[0, 1]$ pixel intensities in quadrant i, and $P = 30$. Class 1 is an "(anti)diagonally dominant" image like ▚ or ▞. A single depth-2 tree trained with TAO achieves a near-perfect training/test error of 0.0%/0.96%. XGBoost forests do much worse. A recent algorithm to train oblique forests (SPORF) performs far worse than the XGBoost, suggesting that proper oblique tree optimization is essential.

4 Learning oblique trees in the GB framework

To learn oblique trees in GB, we build on a recent algorithm, tree alternating optimization (TAO). Given a decision tree of some predetermined structure $\boldsymbol{\tau}(\mathbf{x}; \boldsymbol{\Theta})$ with parameters $\boldsymbol{\Theta} = \{(\mathbf{w}_i, w_{i0})\}_{i\in\mathcal{D}} \cup \{\boldsymbol{\theta}_i\}_{i\in\mathcal{L}}$, decision nodes in set $\mathcal{D}$, leaves in set $\mathcal{L}$, we optimize:

$$\min_{\boldsymbol{\Theta}} \sum_{n=1}^{N} l(\mathbf{g}_n, \mathbf{H}_n, \boldsymbol{\tau}(\mathbf{x}_n; \boldsymbol{\Theta})) + \alpha \sum_{i\in\mathcal{D}} \|\mathbf{w}_i\|_1 \quad \text{with} \quad l(\mathbf{g}, \mathbf{H}, \gamma) = \mathbf{g}^T\gamma + \tfrac{1}{2}\gamma^T\mathbf{H}\gamma. \tag{1}$$

TAO is based on two theorems. First, eq. (1) *separates over any subset of non-descendant nodes* (e.g. all the nodes at the same depth); this follows from the fact that the tree makes hard decisions. Second, optimizing over the parameters of a single node i simplifies to a well-defined *reduced problem* over the instances that currently reach node i (the *reduced set* $\mathcal{R}_i \subset \{1, \dots, N\}$). The form of the reduced problem depends on the type of node:

Decision node It is a *weighted 0/1 loss binary classification problem*, where the two classes correspond to the left and right child, which are the only possible outcomes for an instance. Child $\texttt{left}_i$ ($\texttt{right}_i$) incurs a loss (weight) given by the prediction of the leaf reached from the left (right) child's subtree. Thus, each instance is assigned as *pseudolabel* the child with lower loss. This problem is NP-hard but can be well approximated with a convex surrogate; we use ℓ_1-regularized logistic regression where each instance is weighted by the loss difference between the winner and the other child.

Leaf The reduced problem consists of optimizing the original loss but over the leaf classifier on its reduced set:

$$\min_{\boldsymbol{\theta}_i} \sum_{n\in\mathcal{R}_i} \mathbf{g}_n^T\boldsymbol{\theta}_i + \frac{1}{2}\boldsymbol{\theta}_i^T\mathbf{H}_n\boldsymbol{\theta}_i. \tag{2}$$

This is solved similarly as in XGBoost and other frameworks.

Given an initial tree structure with initial parameter values, the resulting algorithm repeatedly visits nodes in reverse breadth-first search order. Each iteration trains all nodes at the same depth (in parallel) from the leaves to the root, by solving either an ℓ_1-regularized logistic regression at each decision node, or the above exact solution as each leaf.

4 Experiments: classification and regression

Pendigits				
Forest	E_{test} (%)	#pars.	T	Δ
LightGBM	3.49±0.00	90k	1k	11
XGBoost	3.46±0.00	18k	1k	4
XGBoost	3.46±0.00	137k	10k	4
LightGBM	3.31±0.00	895k	10k	4
GB-TAO	3.15±0.25	1.3k	1	8
SPORF	2.91±0.09	(1.6M)	1k	20
SPORF	2.87±0.01	(105k)	100	20
GB-TAO	2.17±0.02	13k	10	7
GB-TAO	2.00±0.04	44k	30	7

News20				
Forest	E_{test} (%)	#pars.	T	Δ
GB-sklearn	23.42±0.03	156k	2k	6
SPORF	22.51±0.09	(1.3M)	100	569
XGBoost	21.39±0.00	705k	20k	6
XGBoost	21.34±0.00	188k	6k	6
LightGBM	20.69±0.00	1.8M	20k	27
LightGBM	19.78±0.00	546k	6k	28
GB-TAO	18.13±0.01	479k	400	4
GB-TAO	18.76±0.01	746k	50	6
GB-TAO	16.65±0.04	1.6M	800	4

MNIST				
Forest	E_{test} (%)	#pars.	T	Δ
XGBoost	4.38±0.00	70k	100	10
GB-TAO	4.17±0.08	21k	1	12
SPORF	2.89±0.04	(143M)	1k	50
XGBoost	2.20±0.00	107k	1k	6
LightGBM	2.02±0.00	121k	1k	10
GB-TAO	1.94±0.00	671k	30	10
XGBoost	1.91±0.00	505k	10k	6
LightGBM	1.62±0.00	642k	10k	21
GB-TAO	1.55±0.02	7.2M	1.4k	7

CT-slice				
Forest	E_{test} (%)	#pars.	T	Δ
LightGBM	1.53±0.00	153k	100	97
LightGBM	1.52±0.00	91k	1k	23
XGBoost	1.50±0.00	107k	100	10
GB-TAO	1.28±0.02	28k	1	8
GB-sklearn	1.26±0.03	900k	1k	10
XGBoost	1.26±0.00	767k	1k	10
GB-TAO	0.90±0.02	81k	30	4
GB-TAO	0.45±0.01	1.2M	100	6

casp				
Forest	E_{test} (%)	#pars.	T	Δ
XGBoost	3.66±0.00	119k	100	10
XGBoost	3.58±0.00	793k	1k	10
GB-sklearn	3.58±0.01	854k	1k	10
LightGBM	3.54±0.00	153k	100	114
GB-TAO	3.49±0.01	256k	50	12
LightGBM	3.48±0.00	766k	1k	109
GB-TAO	3.43±0.00	481k	100	12
GB-TAO	3.39±0.01	887k	200	12

Year				
Forest	E_{test}	#pars.	T	Δ
GB-TAO	9.17±0.01	19k	1	8
XGBoost	9.05±0.00	153k	100	10
LightGBM	9.03±0.00	153k	100	37
GB-sklearn	8.96±0.02	171k	1k	6
LightGBM	8.92±0.00	1.5M	1k	43
XGBoost	8.91±0.00	1.8M	1k	10
GB-TAO	8.88±0.02	78k	20	6
GB-TAO	8.73±0.01	402k	100	6

Table 1: Comparison of different forest-based models for classification (top) and regression (bottom), sorted by decreasing test error. We report 0-1 test error or RMSE E_{test} (mean±std over 5 repeats), and the number of parameters in the model. T refers to the number of trees. Δ is the max depth of the forest.

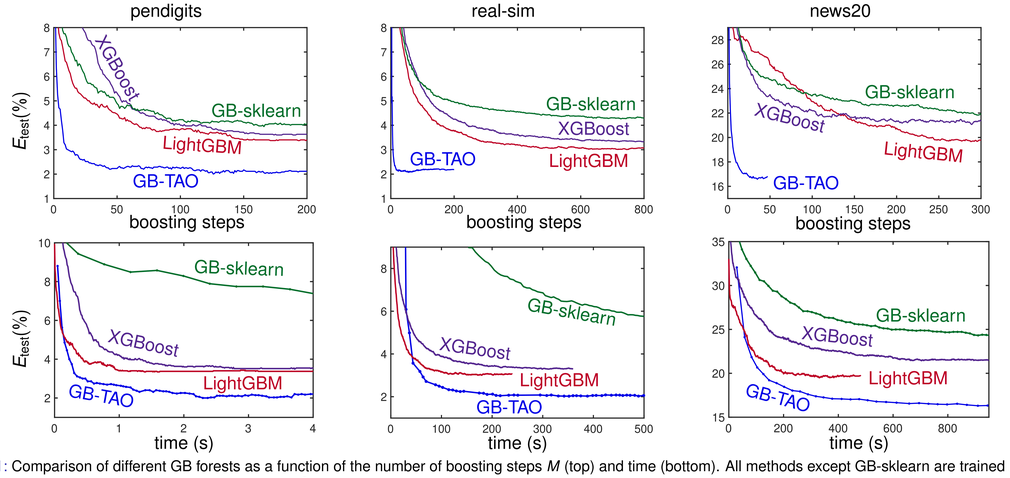


Figure 1: Comparison of different GB forests as a function of the number of boosting steps M (top) and time (bottom). All methods except GB-sklearn are trained using parallel processing on a shared memory system with 8 processors.