Non-Greedy Algorithms for Decision Tree Optimization: An Experimental Comparison

Arman Zharmagambetov, Suryabhan Singh Hada,
Magzhan Gabidolla, Miguel Á. Carreira-Perpiñán

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

IEEE IJCNN 2021

Overview and motivation

- Decision trees occupy a special place in ML/statistics [5, 1] and have been widely used in many applications [12, 8].

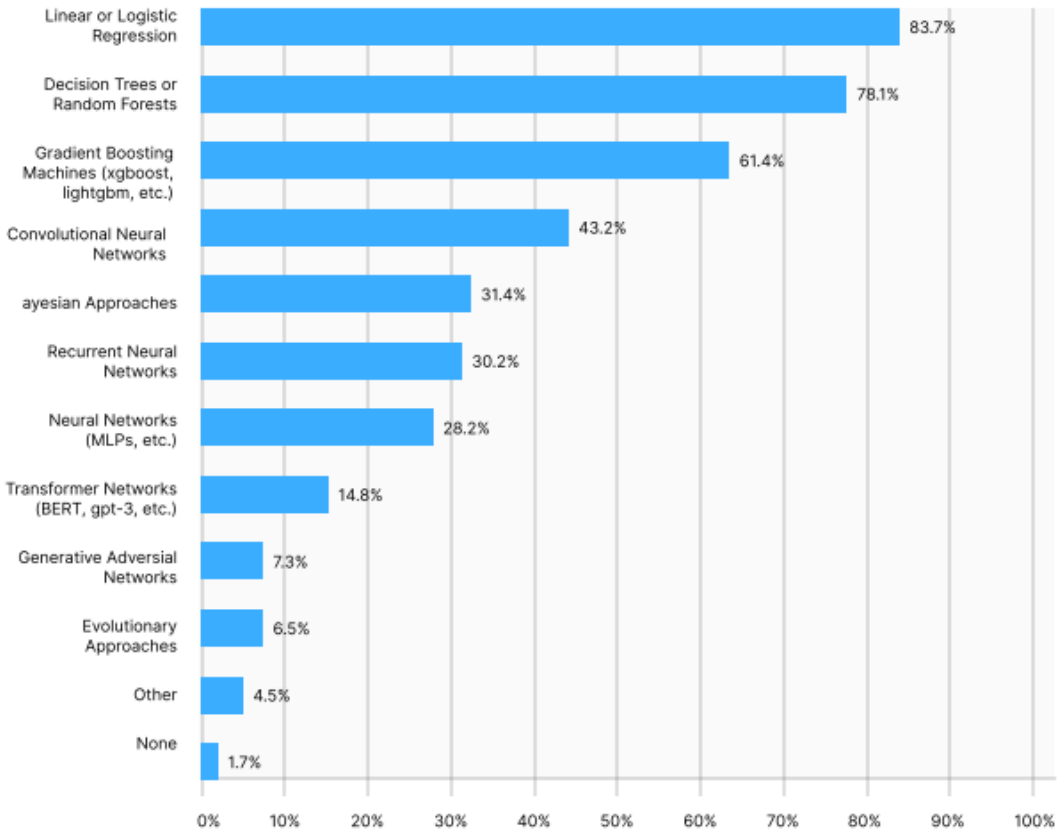

Figure 1: Methods and algorithms usage. Source: kdnuggets survey (Mar, 2020).

Overview and motivation

This is because of the unique properties which separate them from other models:

- ✓ interpretability: thanks to the hierarchical structure
- ✓ fast inference time: instance follows unique root-leaf path
- ✓ can naturally handle: categorical (discrete) features, nonlinear and multiclass problems
- ✓ often used as base learners in state-of-the-art ensemble models (e.g. XGBoost, Random forests, etc.)
- ✓ etc...

This is because of the unique properties which separate them from other models:

- ✓ interpretability: thanks to the hierarchical structure
- ✓ fast inference time: instance follows unique root-leaf path
- ✓ can naturally handle: categorical (discrete) features, nonlinear and multiclass problems
- ✓ often used as base learners in state-of-the-art ensemble models (e.g. XGBoost, Random forests, etc.)
- ✓ etc...

However, they are difficult to train (non-differentiable, non-convex). Most learning methods are dominated by greedy recursive partitioning algorithms (such as CART [3] or C5.0 [11]) applied to axes-aligned (univariate) trees.

Our contributions:

- While CART and C5.0 date from the 1980s, many more algorithms for learning trees have been proposed since then (including **non-greedy ones**) and they were not included in the previous comparison papers and reviews [6, 8]. The question is: do they make a difference in practice? We explore this here on 8 representative tree learning algorithms on 20 classification and 7 regression datasets.

Our contributions:

- While CART and C5.0 date from the 1980s, many more algorithms for learning trees have been proposed since then (including **non-greedy ones**) and they were not included in the previous comparison papers and reviews [6, 8]. The question is: do they make a difference in practice? We explore this here on 8 representative tree learning algorithms on 20 classification and 7 regression datasets.
- We extend the comparison beyond traditional axes-aligned (univariate) trees and show the performance comparison of different methods to train oblique trees (linear multivariate). They are currently less used in practice but have a huge potential, as shown here.

The Algorithms

We group the algorithms that we compare into the following three categories:

- Greedy recursive partitioning algorithms
- Non-greedy, global optimization algorithms
- Non-greedy, approximate brute force search via branch-and-bound

The Algorithms

We group the algorithms that we compare into the following three categories:

- **Greedy recursive partitioning algorithms**
- Non-greedy, global optimization algorithms
- Non-greedy, approximate brute force search via branch-and-bound

Greedy recursive partitioning algorithms

- **CART** [3] applies a greedy recursive partitioning which optimizes an impurity measure (Gini index) at each node by enumerating all the features and thresholds. We use R (`rpart`) and Python (`scikit-learn`) implementations.

Greedy recursive partitioning algorithms

- **CART** [3] applies a greedy recursive partitioning which optimizes an impurity measure (Gini index) at each node by enumerating all the features and thresholds. We use R (`rpart`) and Python (`scikit-learn`) implementations.
- **C5.0** [11] is similar to CART with minor differences, such as the impurity measure (information gain). We use the single-threaded Linux version provided by the authors in the form of executables.

Greedy recursive partitioning algorithms

- **CART** [3] applies a greedy recursive partitioning which optimizes an impurity measure (Gini index) at each node by enumerating all the features and thresholds. We use R (`rpart`) and Python (`scikit-learn`) implementations.
- **C5.0** [11] is similar to CART with minor differences, such as the impurity measure (information gain). We use the single-threaded Linux version provided by the authors in the form of executables.
- **OC1** [9] considers oblique trees and uses coordinate descent to optimize the impurity measure at each split (with random restarts). We use the C implementation provided by authors.

Greedy recursive partitioning algorithms

- **CART** [3] applies a greedy recursive partitioning which optimizes an impurity measure (Gini index) at each node by enumerating all the features and thresholds. We use R (`rpart`) and Python (`scikit-learn`) implementations.
- **C5.0** [11] is similar to CART with minor differences, such as the impurity measure (information gain). We use the single-threaded Linux version provided by the authors in the form of executables.
- **OC1** [9] considers oblique trees and uses coordinate descent to optimize the impurity measure at each split (with random restarts). We use the C implementation provided by authors.
- **GUIDE** [8]. Like CART, it grows a tree greedily and recursively, but adds a number of heuristics, i.e., in how features and split points are selected. It can also learn oblique trees. We use the version provided by the authors in the form of executables.

The Algorithms

We group the algorithms that we compare into the following three categories:

- Greedy recursive partitioning algorithms
- **Non-greedy, global optimization algorithms**
- Non-greedy, approximate brute force search via branch-and-bound

Non-greedy, global optimization algorithms

- **CO2** [10]. Given the initial tree, CO2 formulates a convex-concave upper bound on the tree's empirical loss and optimizes it using SGD. The implementation is not available online and thus, we report the results from their paper.

Non-greedy, global optimization algorithms

- **CO2** [10]. Given the initial tree, CO2 formulates a convex-concave upper bound on the tree's empirical loss and optimizes it using SGD. The implementation is not available online and thus, we report the results from their paper.
- **TAO** [4] can be applied to train both axis-aligned and oblique trees. The algorithm starts with a decision tree of the given structure and minimizes the desired classification or regression objective function (with/without regularization) by applying an alternating optimization (i.e., fixing one part of the tree and optimizing over another part). Each iteration of the algorithm is guaranteed to decrease an objective function. We use the Python implementation.

The Algorithms

We group the algorithms that we compare into the following three categories:

- Greedy recursive partitioning algorithms
- Non-greedy, global optimization algorithms
- **Non-greedy, approximate brute force search via branch-and-bound**

Non-greedy, global optimization algorithms

- **OCT** (Optimal Classification Trees) [2] formulate the optimization as a mixed-integer optimization (MIO) problem by using binary variables to encode it. This is then solved via commercial MIO solvers. It can train both axis-aligned and oblique trees. We report the results from the corresponding paper (for both axis aligned and oblique).

Non-greedy, global optimization algorithms

- **OCT** (Optimal Classification Trees) [2] formulate the optimization as a mixed-integer optimization (MIO) problem by using binary variables to encode it. This is then solved via commercial MIO solvers. It can train both axis-aligned and oblique trees. We report the results from the corresponding paper (for both axis aligned and oblique).
- **GOSDT** (Generalized and Scalable Optimal Sparse Decision Trees) [7] use a custom branch-and-bound algorithm instead of MIO solvers, including some accelerations. It is restricted to axis-aligned trees having binary inputs and outputs only (i.e., boolean functions), which requires discretizing any continuous features. We use the C++ implementation and the Python interface provided by authors.

Experiments: datasets

	Dataset	N_{train}	N_{test}	D	K
classification	Iris	120	30	4	3
	Wine	142	36	13	3
	Dermatology	293	73	34	6
	Balance scale	500	125	4	3
	Breast Cancer	559	140	9	2
	Blood Trans	598	150	4	2
	German	800	200	20	2
	Banknote auth	1098	274	4	2
	Contraceptive	1178	295	9	3
	Car Eval	1382	346	6	4
	Segment	1848	462	19	7
	Spambase	3681	920	57	2
	Optical recog	3823	1797	64	10
	Landsat	4435	2000	36	6
	Pendigits	7494	3498	16	10
	Letter	16000	4000	16	26
	Connect4	54046	13511	126	3
	MNIST-pixels	60000	10000	784	10
	MNIST-LeNet5	60000	10000	800	10
	SensIT	78823	19705	100	3
regression	concrete	687	343	8	1
	airfoil	1002	501	5	1
	abalone	2506	1671	8	1
	cpuact	4915	3277	21	1
	ailerons	7154	6596	40	1
	CT slice	42800	10700	384	1
	YearPredictionMSD	463715	51630	90	1

Experiments: results

Average test accuracy for axis-aligned trees: Blue - the best performing method, Red - the 2nd best performing method.

Dataset	TAO-ax	CART(R)	CART(Py)	C5.0	GUIDE-ax	GOSDT	OCT-ax
Iris	95.41	93.33	94.75	92.64	93.67	90.00	93.5
Wine	91.21	90.00	90.10	85.19	91.94	timeout	94.2
Dermatology	96.14	93.51	94.44	90.44	95.54	timeout	89.2
Balance scale	82.21	78.96	79.62	78.19	77.44	65.52	71.6
Breast Cancer	95.91	94.57	93.64	94.83	94.57	91.17	91.5
Blood Trans	78.95	75.20	76.45	78.40	78.80	75.67	77.0
German	73.35	72.21	72.29	69.13	69.95	timeout	69.8
Banknote auth	98.16	97.93	96.33	98.70	98.25	timeout	90.7
Contraceptive	56.95	54.37	54.25	49.39	55.19	timeout	53.3
Car Eval	96.67	96.35	96.51	87.59	70.23	68.67	78.8
Segment	96.18	95.91	92.17	94.86	95.37	timeout	—
Spambase	93.19	91.92	89.62	92.85	92.77	timeout	86.1
Optical recog	86.08	84.26	85.19	80.33	84.56	timeout	54.7
Landsat	86.10	85.94	85.21	84.12	85.24	timeout	78.0
Pendigits	92.52	91.62	90.19	91.32	89.93	timeout	—
Letter	86.39	86.04	86.07	85.26	83.93	timeout	—
Connect4	79.88	78.29	77.55	77.84	71.66	timeout	—
MNIST-pixels	88.52	88.03	88.05	88.31	78.52	timeout	—
MNIST-LeNet5	93.52	93.31	93.32	93.48	85.25	timeout	—
SensIT	81.84	81.71	81.00	81.41	78.52	timeout	—
wins	18	0	0	1	0	0	1

Experiments: results (cont.)

The same as before but for oblique trees.

Dataset	TAO-ob	OC1	GUIDE-ob	OCT-ob	CO2
Iris	94.40	85.67	94.33	95.1	—
Wine	92.00	84.45	93.33	91.6	—
Dermatology	97.97	84.46	97.84	92.6	—
Balance scale	94.72	88.96	85.60	87.6	—
Breast Cancer	97.71	81.07	95.64	94.0	—
Blood Trans	80.42	77.93	79.87	77.4	—
German	81.24	68.65	70.15	71.0	—
Banknote auth	99.18	91.64	98.80	98.7	—
Contraceptive	57.47	49.66	56.58	53.3	—
Car Eval	98.03	95.49	70.23	87.5	—
Segment	96.48	88.53	95.48	—	96
Spambase	93.31	78.20	92.24	86.6	—
Optical recog	91.27	62.00	79.19	54.3	—
Landsat	87.81	73.54	85.97	78.2	—
Pendigits	96.80	84.42	91.80	—	92
Connect4	81.09	75.42	72.01	—	78
Letter	90.41	65.81	82.65	—	87
MNIST-pixels	94.74	74.34	73.79	—	90
MNIST-LeNet5	98.22	87.97	85.25	—	—
SensIT	85.44	73.70	79.25	—	82
wins	18	0	1	1	0

Results for regression problems are similar. **Note:** TAO oblique trees (almost consistently) also performed better compared to the axis-aligned trees which makes it the best performing method.

Conclusion

- Majority of the recently proposed algorithms do not significantly improve over classical greedy algorithms (e.g. CART).
- The exception is non-greedy optimization which jointly trains over all the nodes' parameters. In particular with the **TAO algorithm**, it is possible to train highly accurate trees.
- Moreover, TAO makes oblique trees preferable to axis-aligned ones in many cases, since they are much more accurate while remaining small and interpretable. This suggests a change in paradigm in practical applications of decision trees.
- Work supported by NSF award IIS–2007147

Thank you!

References

[1] E. Alpaydın. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, third edition, 2014.

[2] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, July 2017.

[3] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

[4] 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.

[5] 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.

[6] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, Sept. 2000.

[7] J. Lin, C. Zhong, D. Hu, C. Rudin, and M. Seltzer. Generalized and scalable optimal sparse decision trees. In H. Daumé III and A. Singh, editors, *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pages 6177–6187, Online, July 13–18 2020.

[8] W.-Y. Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348 (with discussion, pp. 349–370), Dec. 2014.

[9] S. K. Murthy, S. Kasif, S. Salzberg, and R. Beigel. OC1: A randomized algorithm for building oblique decision trees. In *Proc. of the 11th AAAI Conference on Artificial Intelligence (AAAI 1993)*, pages 322–327, Washington, DC, July 11–15 1993.

[10] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli. Efficient non-greedy optimization of decision trees. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 28, pages 1720–1728. MIT Press, Cambridge, MA, 2015.

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

[12] L. Rokach and O. Maimon. Top-down induction of decision trees classifiers—a survey. *IEEE Trans. Systems, Man, and Cybernetics Part C—Applications and Reviews*, 35(4):476–487, Nov. 2005.