Joint learning and pruning of decision forests

Jean-Michel Begon, Arnaud Joly, Pierre Geurts

jm.begon@ulg.ac.be @JmBegon

Systems and Modeling, Deptartment of EE and CS, University of Liege, Belgium

Université de Liège

Motivations

What? Is it possible to build **accurate yet lightweight** decision forests without building the whole model first?

Why? Decision forests are heavy models memory-wise:

- $\propto$ Number of nodes in a tree is (at worst) linear with the size of the data;
- $\propto$ number of required trees grows with the problem complexity.

What for?

- Big data;
- small memory devices;
- better interpretability, less overfitting, faster prediction, ...

How? Joint learning and pruning (JLP)

JLP's foundation

The forest is a linear model in the "forest space" (nodes' indicator function space):

$$\hat{y}(\mathbf{x}) = \frac{1}{T}\sum_{j=1}^{M} w_j z_j(\mathbf{x})$$

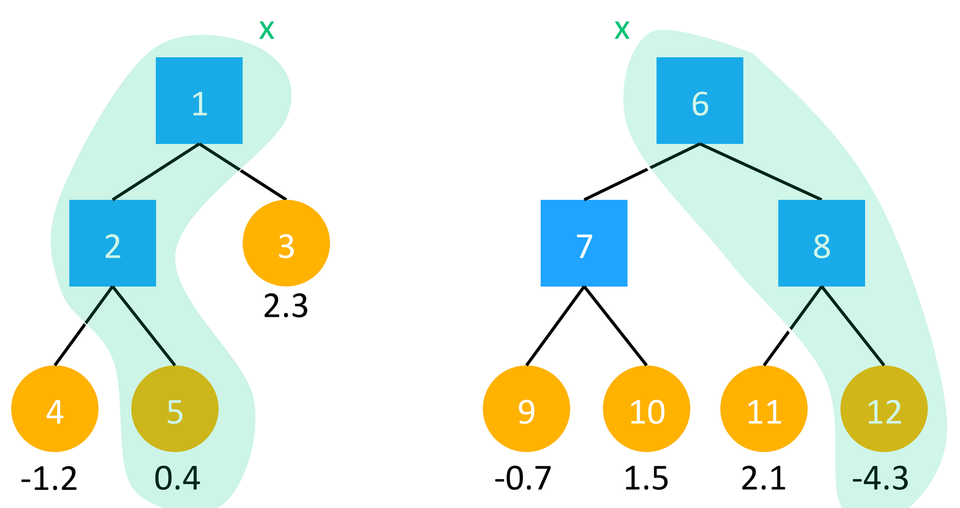
Where

T is the number of trees

M is the total number of nodes

$$z_j(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \text{ reaches node } j \\ 0, & \text{otherwise} \end{cases}$$

$$w_j = \begin{cases} \text{the prediction of leaf } j, \\ 0, \text{ otherwise} \end{cases}$$



$$\mathbf{w}^T = (0\ 0\ 2.3\ -1.2\ 0.4\ 0\ 0\ 0\ -0.7\ 1.5\ 2.1\ -4.3)$$
$$Z(\mathbf{x})^T = (1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1)$$
$$\hat{y}(\mathbf{x}) = \frac{1}{2}\mathbf{w}^T Z(\mathbf{x}) = -1.95$$

JLP iteratively deepens the model in a stagewise fashion by adding the node whose optimal weight reduces the error the most among a pool of candidates.

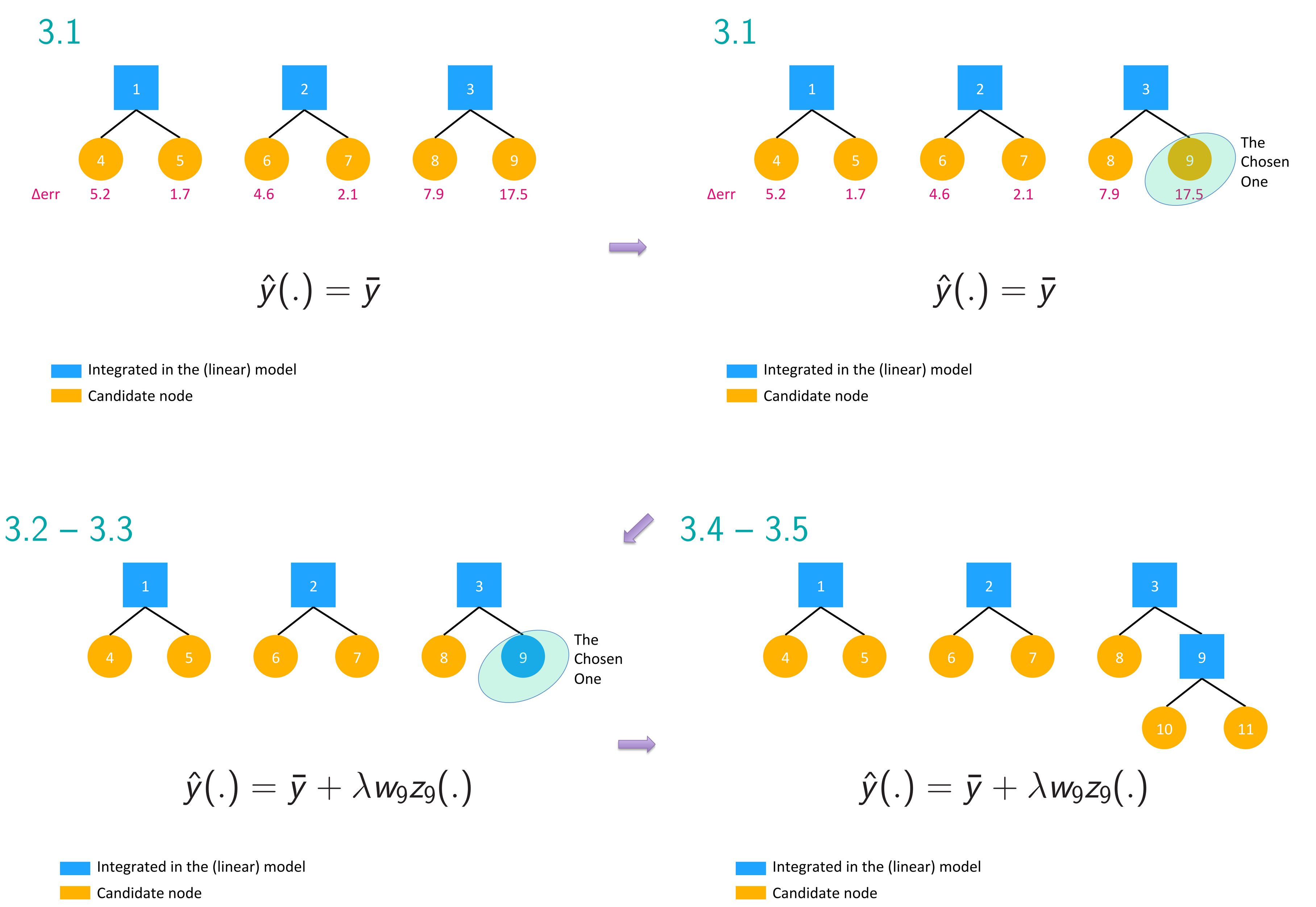
JLP algorithm

Inputs: $D = (x_i, y_i)_{i=1}^N$, the learning set; λ, the learning rate; K, the node budget; $\mathcal{A}$, the tree learning algorithm; T, the number of trees

Output: An ensemble S of K tree nodes with their corresponding weights.

Algorithm:

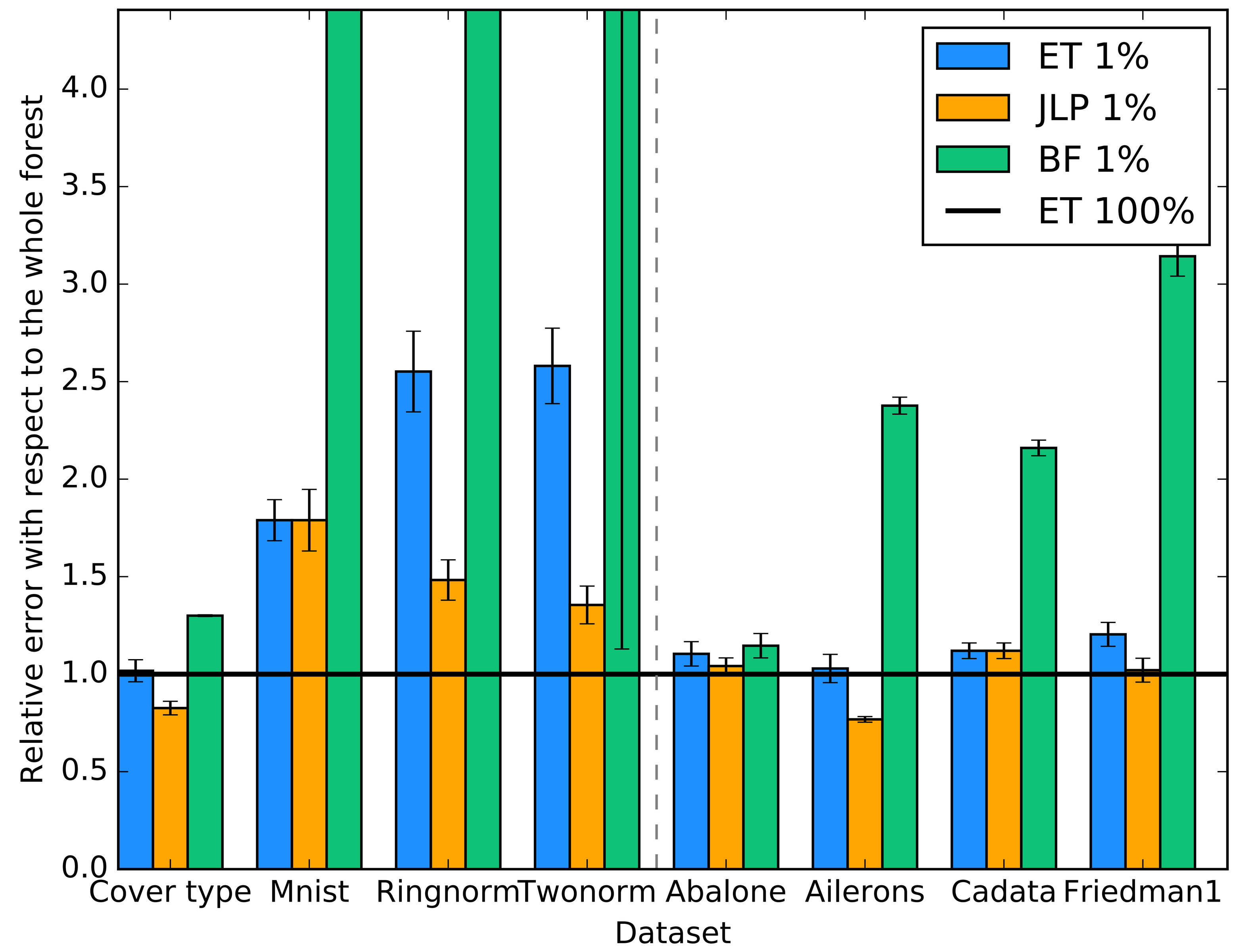
1. $S = \emptyset$; $C = \emptyset$; $\hat{y}^{(0)}(.) = \frac{1}{N}\sum_{i=1}^N y_i$
2. Grow T stumps with $\mathcal{A}$ on D and add both successors of all stumps to C.
3. For $k = 1$ to K:
   - 3.1 Compute:
   $$(j^*, w_j^*) = \underset{j \in C, w \in \mathbb{R}}{\arg\min} \sum_{i=1}^{N} \left(y_i - \left(\hat{y}^{(k-1)}(x_i) + w z_j(x_i)\right)\right)^2$$
   - 3.2 $S = S \cup \{(j^*, w_j^*)\}$; $C = C \setminus \{j^*\}$
   - 3.3 $y^{(k)}(.) = y^{(k-1)}(.) + \lambda w_j^* z_{j^*}(.)$
   - 3.4 Split j^* using $\mathcal{A}$ to obtain children j_l and j_r
   - 3.5 $C = C \cup \{j_l, j_r\}$



JLP versus other prepruning methods

We tested JLP on several standard datasets, starting with $T = 1000$ stumps and a node budget K of 1% of the number of nodes in a forest of 1000 fully-developed extremely randomized trees (ET).

We compared JLP to the whole forest ($ET_{100\%}$), a forest of 10 trees ($ET_{1\%}$) and a best-first approach which grows the trees in parallel, splitting on the leaf which leads to the largest local reduction of the total node impurity, until exhaustion of the 1% budget (BF).



JLP ($\lambda = 10^{-1.5}$) and other methods' relative error with respect to the original forest.

JLP versus a L1-based postpruning method

Several postpruning exists to tackle this problem with the obvious disadvantage of requiring the building of the whole forest.

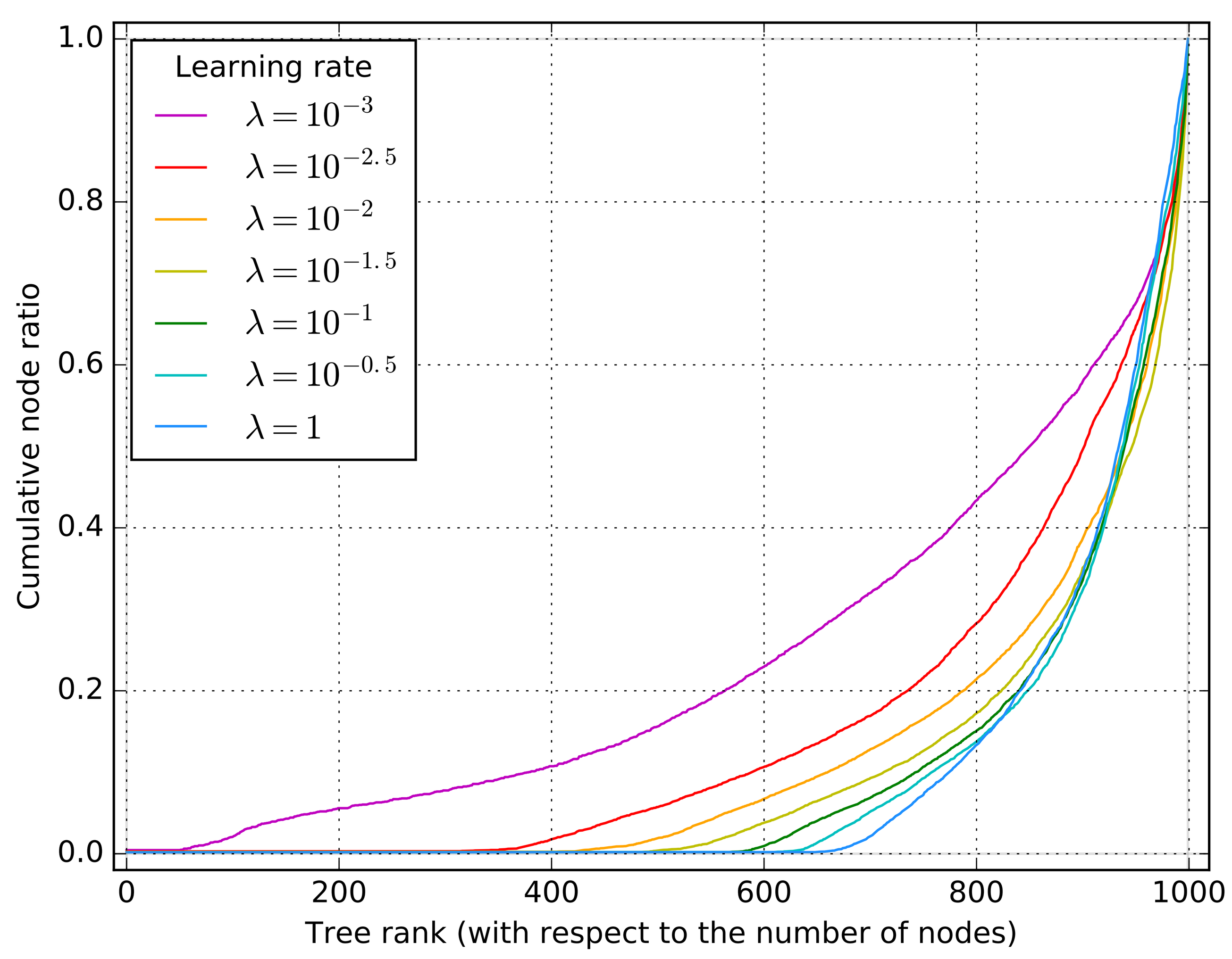
We tested our method against the L1-based compression method (L1P) of Joly et al. (2012) with a budget of 1% of the 1000 thousands trees. Unsurprisingly the latter tends to produce better model at the same node constraint:

Datasets	$ET_{100\%}$	L1P	JLP	λ
Ringnorm	2.9 ± 0.4	3.8 ± 0.4	4.5 ± 0.4	$10^{-1.5}$
Twonorm	3.1 ± 0.1	5.1 ± 0.4	4.5 ± 0.3	$10^{-1.5}$
Ailerons $\times 10^{-8}$	6.9 ± 0.2	4.0 ± 0.0	4.7 ± 0.1	$10^{-0.5}$
Friedman1	4.9 ± 0.2	3.2 ± 0.3	5.0 ± 0.3	$10^{-1.5}$

Error comparison between L1P and JLP.

Influence of the learning rate

Beside controlling the overfitting/underfitting tradreoff, the learning rate has a practical impact on the shape of the forest. Typically, large (resp. small) values of λ favor a more in depth (resp. in breadth) development of the forest:



Influence of the learning rate on the node distribution.