Joint learning and pruning of decision forests

Jean-Michel Begon, Arnaud Joly, Pierre Geurts

Systems and Modeling, Dept. of EE and CS, University of Liege, Belgium

Benelearn 2016

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 :

- Number of nodes in a tree is (at worst) linear with the size of the data ;
- 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" :

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

Where

T is the number of trees

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

i.e. node j indicator function

M is the total number of nodes

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

JLP : iteratively introduce nodes into the tree, optimizing the split locally but the weight globally.

JLP in a nutshell

1. Initialize the model $\hat{y} \leftarrow \frac{1}{N} \sum_{i=1}^{N} y_i$;
2. grow T stumps and add their children to a candidate list C ;
3. repeat until budget exhaustion :
   i. find the best candidate j^* together with its optimal weight w^* :

$$(j^*, w^*) = \underset{j \in C, w \in \mathbb{R}}{\arg\min} \sum_{i=1}^{N} (y_i - \hat{y}(x_i) + w z_j(x_i))^2 \qquad (2)$$

   ii. add node j^* to the model with its weight w^* tempred by some learning rate λ :

$$\hat{y} \leftarrow \hat{y} + \lambda w_j z_j \qquad (3)$$

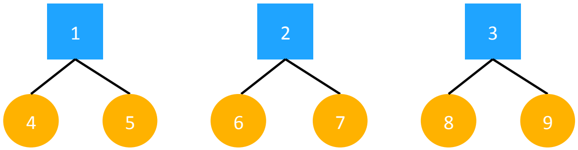
   iii. develop node j^* into its children l and r (if it is possible) and add them to C.

An illustration of JLP — Initialization

Integrated in the (linear) model

$$\hat{y}(.) = \bar{y}$$

An illustration of JLP — Initialization



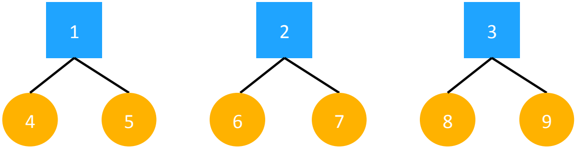
Integrated in the (linear) model

Candidate node

$$\hat{y}(.) = \bar{y}$$

An illustration of JLP — Iterate until there are enough nodes

Loop 1

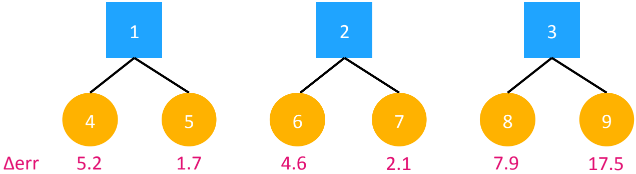


$$(j^*, w^*) = \arg\min_{j \in C, w \in \mathbb{R}} \sum_{i=1}^{N} \left(y_i - \hat{y}(x_i) + w z_j(x_i)\right)^2$$

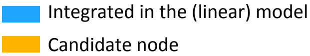
$$\hat{y}(.) = \bar{y}$$

An illustration of JLP — Iterate until there are enough nodes

Loop 1



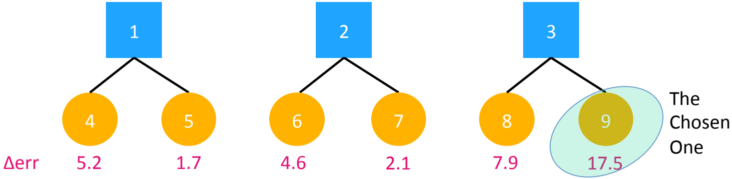
$$(j^*, w^*) = \arg\min_{j \in C, w \in \mathbb{R}} \sum_{i=1}^{N} \left(y_i - \hat{y}(x_i) + w z_j(x_i)\right)^2$$



$$\hat{y}(.) = \bar{y}$$

An illustration of JLP — Iterate until there are enough nodes

Loop 1



$$(j^*, w^*) = \arg\min_{j \in C, w \in \mathbb{R}} \sum_{i=1}^{N} (y_i - \hat{y}(x_i) + w z_j(x_i))^2$$

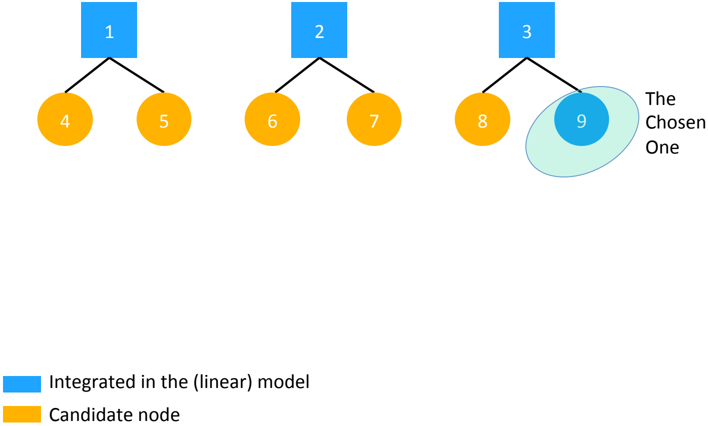
Integrated in the (linear) model

Candidate node

$$\hat{y}(.) = \bar{y}$$

An illustration of JLP — Iterate until there are enough nodes

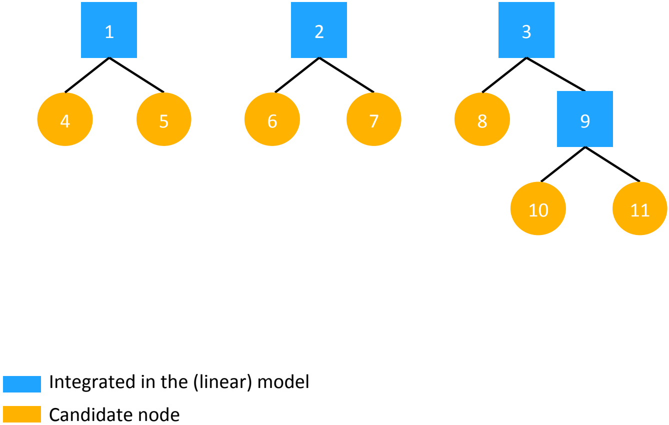
Loop 1



$$\hat{y}(.) = \bar{y} + \lambda w_9 z_9(.)$$

An illustration of JLP — Iterate until there are enough nodes

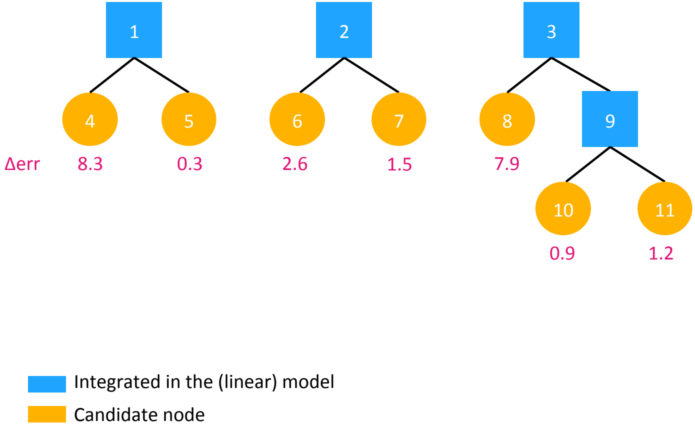
Loop 1



$$\hat{y}(.) = \bar{y} + \lambda w_9 z_9(.)$$

An illustration of JLP — Iterate until there are enough nodes

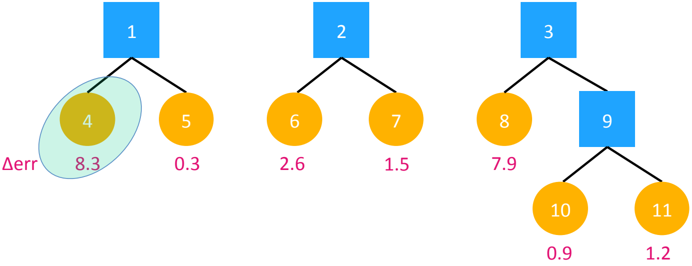
Loop 2



$$\hat{y}(.) = \bar{y} + \lambda w_9 z_9(.)$$

An illustration of JLP — Iterate until there are enough nodes

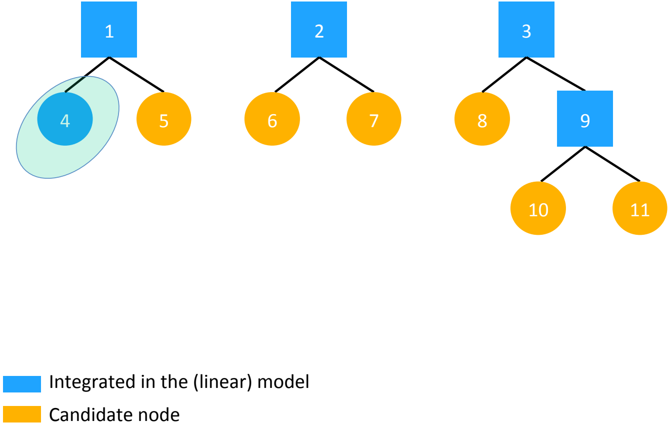
Loop 2



$$\hat{y}(.) = \bar{y} + \lambda w_9 z_9(.)$$

An illustration of JLP — Iterate until there are enough nodes

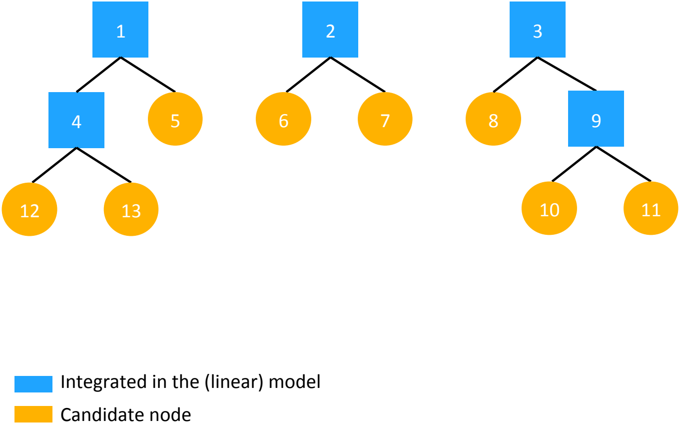
Loop 2



$$\hat{y}(.) = \bar{y} + \lambda w_9 z_9(.) + \lambda w_4 z_4(.)$$

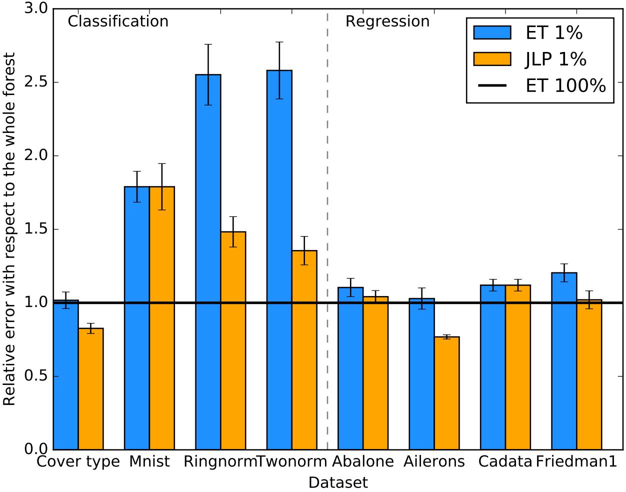
An illustration of JLP — Iterate until there are enough nodes

Loop 2



$$\hat{y}(.) = \bar{y} + \lambda w_9 z_9(.) + \lambda w_4 z_4(.)$$

Results



JLP ($\lambda = 10^{-1.5}$) performance on several datasets.