### Joint learning and pruning of decision forests

Jean-Michel Begon, Arnaud Joly, Pierre Geurts

jm.begon@ulg.ac.be  @JmBegon

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

#### 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” (nodes’ indicator function space):

\[
\hat{y}(x) = \frac{1}{T} \sum_{j=1}^{M} w_j g(x)
\]

Where

- \( T \) is the number of trees
- \( M \) is the total number of nodes

\[
g(x) = \begin{cases} 1, & \text{if } x \text{ reaches node } j, \\ 0, & \text{otherwise} \end{cases}
\]

\[
w_i = \begin{cases} \text{the prediction of leaf } j_i, & \text{if } i \text{ is a leaf node,} \\ 0, & \text{otherwise} \end{cases}
\]

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:** \( D = (x, y) \)\(^\text{N} \), the learning set; \( \lambda \), the learning rate; \( K \), the node budget; \( A \), the tree learning algorithm; \( T \), the number of trees

**Output:** An ensemble of \( S \) of \( K \) tree nodes with their corresponding weights.

**Algorithm:**

1. \( S = \emptyset \); \( A = \emptyset \); \( \hat{y}(0)(x) = \frac{1}{N} \sum_{i=1}^{N} y(x) \)
2. Grow \( T \) stumps with \( A \) on \( D \) and add both successors of all stumps to \( C \).
3. For \( k = 1 \) to \( K \):
   3.1 Compute:
      \[
      (j^*, w^*_j) = \arg \min_{j \in C; w_j \geq 0} \sum_{i=1}^{N} \left( y_i - \left( \hat{y}^{(k-1)}(x_i) + w_j g_j(x_i) \right) \right)^2
      \]
   3.2 \( S = S \cup \{ (j^*, w^*_j) \} \); \( C = C \setminus \{ j^* \} \)
   3.3 \( \hat{y}^{(k)}(x) = \hat{y}^{(k-1)}(x) + \lambda w^*_j g_j(x) \)
   3.4 Split \( j^* \) using \( A \) 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).

![Graph showing relative error comparison between L1P and JLP](image)

**JLP (\( \lambda = 10^{-1}\)) 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:

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ET(_{100%})</th>
<th>L1P</th>
<th>JLP</th>
<th>( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ringnorm</td>
<td>2.9 ± 0.4</td>
<td>3.8 ± 0.4</td>
<td>4.5 ± 0.4</td>
<td>10(^{-15})</td>
</tr>
<tr>
<td>Twonorm</td>
<td>3.1 ± 0.1</td>
<td>5.1 ± 0.4</td>
<td>4.5 ± 0.3</td>
<td>10(^{-15})</td>
</tr>
<tr>
<td>Ailersons \times 10(^{-8})</td>
<td>6.9 ± 0.2</td>
<td>4.0 ± 0.0</td>
<td>4.7 ± 0.1</td>
<td>10(^{-15})</td>
</tr>
<tr>
<td>Friedman11</td>
<td>4.9 ± 0.2</td>
<td>3.2 ± 0.3</td>
<td>5.0 ± 0.3</td>
<td>10(^{-15})</td>
</tr>
</tbody>
</table>

Error comparison between L1P and JLP.

### Influence of the learning rate

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