1. GIF algorithm

Figure 1 illustrates visually the inner loop of the GIF building algorithm: a subset of the candidates nodes is chosen uniformly at random. The contribution of each node is evaluated and the one which reduces the error the most is added to the model. Its children are then built and added to the candidate list.

2. Optimization problem

We are building an additive model by inserting progressively nodes in the forest. At time \( t \), we are trying to find the best node \( j^{(t)} \) from the candidate list \( C_t \) and its associated optimal weight \( w_j^{(t)} \):

\[
 j^{(t)}, w_j^{(t)} = \arg \min_{j \in C_t, w \in \mathbb{R}^K} \sum_{i=1}^{N} L \left( y_i, \hat{y}_i^{(t-1)}(x_i) + wz_j(x_i) \right)
\]

where \( (x_i, y_i)_{i=1}^{N} \) is the learning sample, \( \hat{y}_i^{(t-1)}() \) is the model at time \( t-1 \), \( z_j() \) is the node indicator functions, meaning that it is 1 if its argument reaches node \( j \) and 0 otherwise.

This problem is solved in two steps. First a node \( j \) is selected from \( C_t \) and the corresponding optimal weight, alongside the error reduction, are computed. This is repeated for all nodes and the one achieving the best improvement is selected.

Regression For regression, we used the L2-norm:

\[
 w_j^{(t)} = \arg \min_{w \in \mathbb{R}} \sum_{i=1}^{N} L \left( y_i, \hat{y}_i^{(t-1)}(x_i) + wz_j(x_i) \right)^2
\]

and the solution is given by

\[
 w_j^{(t)} = \frac{1}{|Z_j|} \sum_{i \in Z_j} r_i^{(t-1)}
\]

where \( r_i^{(t-1)} = y_i - \hat{y}_i^{(t-1)}(x_i) \) is the residual at time \( t-1 \) for the \( i \)th training instance and \( Z_j = \{ 1 \leq i \leq N | z_j(x_i) = 1 \} \) is the subset of instances reaching node \( j \).

Classification For classification we used the multi-exponential loss (Zhu et al., 2009). First, we need to encode the labels so that

\[
 y^{(k)}_i = \begin{cases} 1, & \text{if the class of } y_i \text{ is } k \\ -\frac{1}{K-1}, & \text{otherwise} \end{cases}
\]

where \( K \) is the number of classes. Notice that \( \sum_{k=1}^{K} y^{(k)}_i = 0 \). The optimization then becomes

\[
 w_j^{(t)} = \arg \min_{w \in \mathbb{R}^K} \sum_{i=1}^{N} \exp \left( -\frac{1}{K} y^{(t-1)}_i \hat{y}_i^{(t-1)}(x_i) + wz_j(x_i) \right)
\]

Solving for \( \nabla F_j^{(t-1)}(w) = 0 \) yields

\[
 \alpha_j^{(t-1,k)} \phi^{(k)}(w) = \frac{1}{K} \sum_{l=1}^{K} \alpha_j^{(t-1,l)} \phi^{(l)}(w)
\]

for \( 1 \leq k \leq K \), where

\[
 \alpha_j^{(t-1,k)} \triangleq \sum_{i \in Z_j^{(k)}} \exp \left( -\mu^{(t-1)}_i \right)
\]

\[
 \mu^{(t-1)}_i \triangleq \frac{1}{K} \sum_{k=1}^{K} y^{(k)}_i \hat{y}_i^{(t-1,k)}(x_i)
\]

\[
 \phi^{(k)}(w) \triangleq \exp \left( -\frac{1}{K} \psi^{(k)}(w) \right)
\]

\[
 \psi^{(k)}(w) \triangleq -w^{(k)} + \frac{1}{K-1} \sum_{l=1, l \neq k}^{K} w^{(l)}
\]

---

Jean-Michel Begon 1  Arnaud Joly 1  Pierre Geurts 1

1 Department of Electrical Engineering and Computer Science University of Liège, Liège, Belgium. Correspondence to: Jean-Michel Begon <jm.begon@ulg.ac.be>, Pierre Geurts <p.geurts@ulg.ac.be>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).
where \( Z_j^{(k)} = \{ 1 \leq i \leq N | z_{i,j} = 1 \wedge y_i^{(k)} = 1 \} \) is the subset of learning instances of class \( k \) reaching node \( j \). In words, \( \mu_i^{(t-1)} \) is the hyper-margin of instance \( i \) at time \( t-1 \) and \( \alpha_j^{(t-1,k)} \) is the class error of label \( k \) for node \( j \) at time \( t-1 \).

Equation 7 is equivalent to
\[
\alpha_j^{(t-1,k)} \phi_j^{(k)}(w) = \alpha_j^{(t-1,l)} \phi_j^{(l)}(w) \quad 1 \leq k, l \leq K
\] (12)

In keeping with the output representation (Equation 4), we can impose a zero-sum constraint on the prediction to get a unique solution for the \( k \)th component of \( w_j^{(t)} \). If it is imposed at each stage, it means that
\[
\sum_{k=1}^{K} \hat{y}_j^{(t-1,k)} = \sum_{k=1}^{K} \hat{y}_j^{(t,k)} = 0 = \sum_{k=1}^{K} w_j^{(k)} \quad (13)
\]
and this is not impacted by the learning rate.

The corresponding solution is
\[
\phi_j^{(k)}(w) = \exp \left( -\frac{1}{K-1} w_j^{(k)} \right) \quad (14)
\]
\[
\alpha_j^{(t-1,k)} = \sum_{i \in Z_j^{(k)}} \exp \left( -\frac{1}{K-1} \hat{y}_j^{(t-1,k)}(x_i) \right) \quad (15)
\]
\[
w_j^{(t,k)} = \frac{K-1}{K} \sum_{l=1}^{K} \log \frac{\alpha_j^{(t-1,k)}}{\alpha_j^{(t-1,l)}} \quad (16)
\]

3. Equivalence of GIF and the underlying tree

In the case of a single tree (\( T = 1 \)) and a unit learning rate (\( \lambda = 1 \)), both the square loss in regression and the multi-exponential loss in classification produce the same predictions as the underlying tree. This is due to the fact that, when examining the weight to give to node \( j \) at time \( t \), the prediction of time \( t-1 \) relates to the parent node \( \pi_j \) of \( j \). It is thus independent of \( t \) and is also the same for all instances reaching that node.

Consequently, we will adopt the following slight change in notation:
\[
\hat{y}_j = \hat{y}_{\pi_j} + w_j \quad (17)
\]
meaning that the prediction associated to any object reaching node \( j \) is the weight of \( j \) plus the prediction associated to its parent \( \pi_j \). With \( \hat{y}_{\pi_1} = 0 \), the prediction of the root’s pseudo-parent.

3.1. Regression

In regression, the tree prediction \( T\hat{y}_j \) of any leaf \( j \) is the average of the learning set’s outputs reaching that node:
\[
T\hat{y}_j = \frac{1}{|Z_j|} \sum_{i \in Z_j} y_i \quad (18)
\]
The prediction of node \( j \) is
\[
\hat{y}_j = \hat{y}_{\pi_j} + w_j \quad (19)
\]
\[
= \hat{y}_{\pi_j} + \frac{1}{|Z_j|} \sum_{i \in Z_j} (y_i - \hat{y}_{\pi_j}) \quad (20)
\]
\[
= \hat{y}_{\pi_j} + \frac{1}{|Z_j|} \sum_{i \in Z_j} (y_i - \hat{y}_{\pi_j}) \quad (21)
\]
\[
= \frac{1}{|Z_j|} \sum_{i \in Z_j} y_i \quad (22)
\]
The first step is how the additive model is built. The second is the optimal weight value of node \( j \) derived in Equation 3, the third step is due to the fact that the prediction at \( \pi_j \) is constant since there is only one tree.

3.2. Classification

In order to have the same prediction as the underlying tree, we must demonstrate that the probability of being in class \( l \) associated to node \( j \) will be \( \frac{|Z_j^{(l)}|}{|Z_j|} \).

Under the zero-sum constraint, we have
\[
\exp \left( -\frac{1}{K-1} \hat{y}_j^{(l)} \right) = \frac{1}{c_j} \alpha_j^{(l)} \quad (23)
\]
\[
= \frac{1}{c_j} \sum_{i \in Z_j^{(l)}} \exp \left( -\frac{1}{K-1} \hat{y}_j^{(l)} \right) \quad (24)
\]
\[
= \frac{1}{c_j} |Z_j^{(l)}| \exp \left( -\frac{1}{K-1} \hat{y}_j^{(l)} \right) \quad (25)
\]
\[
\exp \left( -\frac{1}{K-1} \hat{y}_j^{(l)} \right) = \exp \left( -\frac{1}{K-1} \hat{y}_j^{(l)} \right) \exp \left( -\frac{1}{K-1} w_j^{(l)} \right) \quad (26)
\]
\[
= \frac{1}{c_j} |Z_j^{(l)}| \quad (27)
\]
\[
P_j(l) = \frac{\exp \left( -\frac{1}{K-1} \hat{y}_j^{(l)} \right)}{\sum_{k=1}^{K} \exp \left( -\frac{1}{K-1} \hat{y}_j^{(k)} \right)} = \frac{|Z_j^{(l)}|}{|Z_j|} \quad (28)
\]
where \( c_j = \left( \prod_{k=1}^{K} \alpha_j^{(k)} \right)^{\frac{1}{K}} \) is a constant. The first equality is a consequence of the value of \( w_j^{(l)} \) (Equation 16). The
second is a due to the definition of \( \alpha^{(l)} \) (Equation 15). The third is a consequence of having a single tree: the prediction of the parent is the same for all instances.

Notice that, in both regression and classification, the equivalence also holds for an internal node: the prediction is the one the tree would have yielded if that node had been a leaf.

4. Datasets

Table 1 sums up the main characteristics of the datasets we used. Abalone, CT slice, California data housing (Caldat), Musk2, Vowel and Letter come from the UCI Machine Learning Repository (Blake & Merz, 1998). Ringnorm, Twonorm and Waveform are described in (Breiman et al., 1998). Hwang F5 comes from the DELVE repository 1. The noise parameter of the Friedman1 dataset (Friedman, 1991) has been set to 1. Hastie is described in (Friedman et al., 2001). Out of the 500 features of Madelon (Guyon et al., 2004), 20 are informative and 50 are redundant; the others are noise. Mnist8vs9 is the Mnist dataset (LeCun et al., 1998) of which only the 8 and 9 digits have been kept. Binary versions of the Mnist, Letter and Vowel datasets have been created as well by grouping the first half and second half classes together.

| Dataset          | \( N \) | \(|TS| \) | \( p \) | \# Classes |
|------------------|---------|---------|--------|-----------|
| Friedman1        | 300     | 2000    | 10     | -         |
| Abalone          | 2506    | 1671    | 10     | -         |
| CT slice         | 2000    | 51500   | 385    | -         |
| Hwang F5         | 2000    | 11600   | 2      | -         |
| Cadata           | 12384   | 8256    | 8      | -         |
| Ringnorm         | 300     | 7100    | 20     | 2         |
| Twonorm          | 300     | 7100    | 10     | 2         |
| Hastie           | 2000    | 10000   | 10     | 2         |
| Musk2            | 2000    | 4598    | 166    | 2         |
| Madelon          | 2200    | 2200    | 500    | 2         |
| Mnist8vs9        | 11800   | 1983    | 784    | 2         |
| Waveform         | 3500    | 1500    | 40     | 3         |
| Vowel            | 495     | 495     | 10     | 11        |
| Mnist            | 50000   | 10000   | 784    | 10        |
| Letter           | 16000   | 4000    | 8      | 26        |

5. Comparison with local baseline algorithms

We have tested three deepening algorithm for decision forest relying on non-global metrics, meaning that the choice of the best candidate is not made according to how well the forest, as a whole, performs. These algorithms share that the final model is exactly a sub-forest of the un-pruned forest; contrary to GIF, no internal weights are fitted and the predictions of at the leaves are the usual tree predictions.

Breadth first deepening This variant consist in adding the nodes level after level, from left to right, producing a heaped forest. As a consequence, all trees have the same (order of) height, implying that the forest can be quite wide but usually shallow.

Random deepening This variant consist in first choosing a tree and then choosing one of its leaves to transform to a decision nodes. Both choices are made uniformly at random so that the trees are expected to have approximately the same number of nodes. The depth, however, might vary significantly.

Best first deepening This variant consist in choosing, among all leaves which could be turned into an internal node, the one which reduces its local impurity the most. Let \( N_c \), \( N_l \) and \( N_r \) be the number of instances reaching the candidate node, candidate left child and candidate right child respectively. Let also \( I_c \), \( I_l \) and \( I_r \) be the impurity (gini index in classification, variance in regression) of the instances reaching the candidate node, candidate left child and candidate right child respectively. Then, for \( N \) learning instances, the local impurity reduction is defined as:

\[
\Delta I_c \triangleq \frac{N_c}{N} \left( I_c - \left( \frac{N_l}{N_c} I_l + \frac{N_r}{N_c} I_r \right) \right)
\]

Since the fraction of learning instances reaching the candidate is accounted for in the reduction of impurity, this approach will naturally favor higher nodes in the trees.

Experiment We conducted the same experiment as for GIF: the three algorithms were tested on ten folds with different learning sample/testing sample splits and were subjected to the 1% and 10% node constraints. We started with a pool of \( T = 1000 \) roots and no restriction was imposed regarding the depth. All of the \( m = p \) features were examined in regression and \( m = \sqrt{p} \) in classification, as suggested in (Geurts et al., 2006). Table 2 holds the average mean square error for the five regression problems and Table 3 holds the average misclassification rate for the classification problems.

Regression The trend is quite clear: both at 1% and 10%, the breadth first algorithm is the best and the best first is (largely) the worst. There are two instances where the local baselines are able to beat GIF: on Abalone and Hwang F5 at 10%. Interestingly, these are the same cases on which GIF was beaten by a small forest of Extremely randomized trees. The 10% Hwang F5 case aside, the local baselines always underperform the smaller fully-developed forest. Overall, such variants do not seem adequate for regression.
Classification In classification, the breadth first and random baselines tend to perform similarly, one beating the other on some problems. Once again, the best first approach seems to be lagging behind on some datasets. At 10%, the local baselines cannot rival with the other methods. Only on Waveform are they able to reach the other performances—a setting where all methods seems to produce close results. At 1%, the breadth first and/or the random methods surpass the ET at 10% on Twonorm, Hastie, Madelon and Waveform. Those datasets correspond to cases where ET was under-performing significantly compared to GIF. All in all, the local baselines are never able to beat GIF, even in the multiclass setting, which is particularly unfavorable to pruning. Once again, the conclusion is against the purely local baselines.

We believed the poor performances of the baselines are due to the building mechanism of traditional ensemble methods. Although the trees are built independently and with randomization, there remains an important redundancy between them, which is especially unfavorable to pruning. A global approach is better able to avoid redundancy and can thus better exploit the node budget. This would also explain why the best first variant performs worst in both regression and classification: it is prone at picking redundant nodes, why the breadth first variant performs worst in both regression and classification. It would also explain why the best first variant performs worst in both regression and classification. It is prone at picking redundant nodes, why the best first variant performs worst in both regression and classification.

### Table 2. Average mean square error for local baselines at 1% and 10% budgets ($T = 1000$, $m = p$).

<table>
<thead>
<tr>
<th>DATASET</th>
<th>BREADTH FIRST 10%</th>
<th>RANDOM 10%</th>
<th>BEST FIRST 10%</th>
<th>BREADTH FIRST 1%</th>
<th>RANDOM 1%</th>
<th>BEST FIRST 1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRIEDMAN</td>
<td>6.02 ± 0.28</td>
<td>6.80 ± 0.34</td>
<td>15.00 ± 0.39</td>
<td>11.73 ± 0.46</td>
<td>12.52 ± 0.47</td>
<td>15.29 ± 0.42</td>
</tr>
<tr>
<td>ABALONE</td>
<td>4.72 ± 0.23</td>
<td>4.77 ± 0.23</td>
<td>6.82 ± 0.33</td>
<td>5.42 ± 0.27</td>
<td>5.55 ± 0.27</td>
<td>6.82 ± 0.33</td>
</tr>
<tr>
<td>CT SLICE</td>
<td>30.39 ± 1.90</td>
<td>36.19 ± 1.84</td>
<td>310.87 ± 4.79</td>
<td>82.19 ± 2.41</td>
<td>97.24 ± 1.90</td>
<td>313.84 ± 4.64</td>
</tr>
<tr>
<td>HWANG F5</td>
<td>6.73 ± 0.07</td>
<td>6.83 ± 0.06</td>
<td>56.57 ± 6.03</td>
<td>8.52 ± 0.24</td>
<td>13.17 ± 0.44</td>
<td>56.60 ± 6.07</td>
</tr>
<tr>
<td>CADATA</td>
<td>29.24 ± 0.73</td>
<td>31.08 ± 0.74</td>
<td>75.23 ± 0.95</td>
<td>43.40 ± 1.18</td>
<td>47.47 ± 1.02</td>
<td>75.48 ± 0.95</td>
</tr>
</tbody>
</table>

### Table 3. Error rate (%) for local baselines at 1% and 10% budgets ($T = 1000$, $m = p$). The six first datasets are binary classification. The last three are multiclass. The three in the middle are their binary versions.

<table>
<thead>
<tr>
<th>DATASET</th>
<th>BREADTH FIRST 10%</th>
<th>RANDOM 10%</th>
<th>BEST FIRST 10%</th>
<th>BREADTH FIRST 1%</th>
<th>RANDOM 1%</th>
<th>BEST FIRST 1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KINGNORM</td>
<td>4.25 ± 1.24</td>
<td>4.08 ± 1.12</td>
<td>8.38 ± 6.94</td>
<td>8.94 ± 7.45</td>
<td>8.53 ± 7.04</td>
<td>8.94 ± 7.41</td>
</tr>
<tr>
<td>TWONORM</td>
<td>3.51 ± 0.26</td>
<td>3.53 ± 0.30</td>
<td>5.59 ± 1.85</td>
<td>5.91 ± 3.03</td>
<td>6.52 ± 4.28</td>
<td>7.28 ± 4.34</td>
</tr>
<tr>
<td>MUSK2</td>
<td>7.01 ± 0.40</td>
<td>7.63 ± 0.43</td>
<td>15.42 ± 0.23</td>
<td>15.42 ± 0.23</td>
<td>14.29 ± 0.23</td>
<td>15.42 ± 0.23</td>
</tr>
<tr>
<td>MADELON</td>
<td>11.68 ± 0.67</td>
<td>11.92 ± 0.65</td>
<td>19.12 ± 1.94</td>
<td>16.26 ± 0.97</td>
<td>16.70 ± 1.07</td>
<td>20.14 ± 2.41</td>
</tr>
<tr>
<td>MIST8VS9</td>
<td>2.20 ± 0.38</td>
<td>2.37 ± 0.39</td>
<td>6.17 ± 0.73</td>
<td>4.53 ± 0.48</td>
<td>4.84 ± 0.51</td>
<td>6.67 ± 0.69</td>
</tr>
<tr>
<td>BIN. VOWEL</td>
<td>8.99 ± 1.96</td>
<td>8.85 ± 2.03</td>
<td>16.37 ± 3.02</td>
<td>18.73 ± 3.08</td>
<td>19.90 ± 3.71</td>
<td>21.80 ± 4.38</td>
</tr>
<tr>
<td>BIN. MNIST</td>
<td>4.46 ± 0.25</td>
<td>4.91 ± 0.27</td>
<td>21.71 ± 0.30</td>
<td>10.09 ± 0.25</td>
<td>11.78 ± 0.32</td>
<td>22.50 ± 0.35</td>
</tr>
<tr>
<td>BIN. LETTER</td>
<td>5.91 ± 0.43</td>
<td>5.71 ± 0.40</td>
<td>26.16 ± 0.86</td>
<td>17.91 ± 0.77</td>
<td>18.05 ± 0.78</td>
<td>26.19 ± 0.88</td>
</tr>
<tr>
<td>WAVEFORM</td>
<td>14.74 ± 0.63</td>
<td>14.83 ± 0.76</td>
<td>20.25 ± 2.22</td>
<td>16.75 ± 1.26</td>
<td>17.13 ± 1.25</td>
<td>20.45 ± 2.21</td>
</tr>
<tr>
<td>VOWEL</td>
<td>14.26 ± 2.41</td>
<td>13.21 ± 2.33</td>
<td>41.49 ± 5.45</td>
<td>42.40 ± 4.33</td>
<td>40.28 ± 4.62</td>
<td>50.44 ± 5.81</td>
</tr>
<tr>
<td>MNIST</td>
<td>4.63 ± 0.27</td>
<td>4.96 ± 0.26</td>
<td>28.54 ± 0.59</td>
<td>8.60 ± 0.35</td>
<td>9.76 ± 0.31</td>
<td>29.72 ± 0.61</td>
</tr>
<tr>
<td>LETTER</td>
<td>7.06 ± 0.29</td>
<td>6.39 ± 0.20</td>
<td>36.92 ± 1.80</td>
<td>22.11 ± 0.59</td>
<td>20.90 ± 0.55</td>
<td>37.27 ± 1.78</td>
</tr>
</tbody>
</table>

**References**

Blake, Catherine and Merz, Christopher J. {UCI} repository of machine learning databases. 1998.


Figure 1. Illustration of the GIF regression building algorithm ($T = 3, CW = 3$)