Globally Induced Forest: A Prepruning Compression Scheme

Jean-Michel Begon, Arnaud Joly, Pierre Geurts

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

ICML 2017
Goal and motivation

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

Why? Random 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?
Build an additive model corresponding to a forest by introducing decision nodes sequentially until a node budget constraint is met.

- Splits in decision nodes are optimized locally.
- Nodes are taken from a candidate list.
- The best node is added . . .
- . . . together with its weight.
GIF algorithm — High level view

Build an additive model corresponding to a forest by introducing decision nodes sequentially until a node budget constraint is met.

- Splits in decision nodes are optimized locally.
- Nodes are taken from a candidate list.
- The best node is added ...
- ... together with its weight.
GIF algorithm — Additive model

The model prediction \( \hat{y}^{(t)}(x) \) at step \( t \) for instance \( x \) is given by:

\[
\hat{y}^{(t)}(x) = \hat{y}^{(t-1)}(x) + \lambda w_t z_t(x) = w_0 + \lambda \sum_{\tau=1}^{t} w_\tau z_\tau(x) \quad (1)
\]

where

- \( w_0 \) is some initial bias.
- \( w_\tau \) is the weight of node \( \tau \) (\( 1 \leq \tau \leq t \)).
- \( \lambda \) is the learning rate.

\( z_\tau(x) = \begin{cases} 1, & \text{if } x \text{ reaches node } \tau \\ 0, & \text{otherwise} \end{cases} \quad (1 \leq \tau \leq t)
\]

\( i.e. \) node \( \tau \) indicator function

For classification, the sum of weights represents the class probability vector (\( i.e. \) the weights are multidimensional).
\[ \hat{y}(x) = w_0 + \lambda(1.4 + 1.2) + \lambda(-0.7 + -1.9) \] (2)
GIF algorithm — High level view

Build an additive model corresponding to a forest by introducing decision nodes sequentially until a node budget constraint is met.

- Splits in decision nodes are optimized locally.
- Nodes are taken from a candidate list.
- The best node is added . . .
- . . . together with its weight.
GIF algorithm — Illustration (regression)

\[ \hat{y}(x) = w_0 + \lambda w_9 z_9(x) \]

- Node belonging to the model
- Hypothetical unpruned trees
- Candidate node
GIF algorithm — Illustration (regression)

\[ \hat{y}(x) = w_0 + \lambda w_9 z_9(x) \]

- Node belonging to the model
- Hypothetical unpruned trees
- Candidate node
- Randomly preselected candidate node
The GIF algorithm — Illustration (regression)

![Diagram of the GIF algorithm]

- **Node belonging to the model**
- **Hypothetical unpruned trees**
- **Candidate node**
- **Randomly preselected candidate node**

Mathematical expression:

\[
\hat{y}(x) = w_0 + \lambda w_9 z_9(x)
\]
GIF algorithm — Illustration (regression)

\[ \hat{y}(x) = w_0 + \lambda w_9 z_9(x) \]

- Node belonging to the model
- Hypothetical unpruned trees
- Candidate node
- Randomly preselected candidate node

Candidate node

\[ \Delta \text{err} = 2.6 \]

Chosen node

\[ \text{Randomly preselected candidate node} \]

\[ \text{Goal} \]

\[ \text{GIF algorithm} \]

\[ \text{Results} \]

\[ \text{Conclusion} \]
GIF algorithm — Illustration (regression)

\[ \hat{y}(x) = w_0 + \lambda w_9 z_9(x) + \lambda w_6 z_6(x) \]

- Node belonging to the model
- Hypothetical unpruned trees
- Candidate node
**GIF algorithm — Illustration (regression)**

The GIF algorithm involves the following steps:

1. **Node belonging to the model**
2. **Hypothetical unpruned trees**
3. **Candidate node**

The equation for the predicted value $\hat{y}(x)$ is:

$$\hat{y}(x) = w_0 + \lambda w_9 z_9(x) + \lambda w_6 z_6(x)$$
GIF algorithm — High level view

Build an additive model corresponding to a forest by introducing decision nodes sequentially until a node budget constraint is met.

- Splits in decision nodes are optimized locally.
- Nodes are taken from a candidate list.
- The best node is added . . .
- . . . together with its weight.

Candidate list
Each time a node is added to the model, the learning instances reaching that node are split according to a local criterion. The resulting children are placed in the candidate list.
Build an additive model corresponding to a forest by introducing decision nodes sequentially until a node budget constraint is met.

- Splits in decision nodes are optimized locally.
- Nodes are taken from a candidate list.
- The best node is added . . .
- . . . together with its weight.
GIF algorithm — Node selection

The best node $j^*$, together with its optimal weight $w_j^*$, are the ones that minimize some loss $L$ over the training set $(x_i, y_i)_{i=1}^{N}$:

$$
(j^*, w_j^*) = \arg\min_{j \in C_t, w \in \mathbb{R}^K} \sum_{i=1}^{N} L \left( y_i, \hat{y}^{(t-1)}(x_i) + wz_j(x_i) \right)
$$  (3)

where $C_t$ is the subsample of candidates.

This problem is solved in two steps:

1. for a candidate $j$, compute the best weight $w_j^*$:

$$
w_j^* = \arg\min_{w \in \mathbb{R}^K} \sum_{i=1}^{N} L \left( y_i, \hat{y}^{(t-1)}(x_i) + wz_j(x_i) \right)
$$  (4)

2. select the best candidate with exhaustive search:

$$
j^* = \arg\min_{j \in C_t} \sum_{i=1}^{N} L \left( y_i, \hat{y}^{(t-1)}(x_i) + w_j^* z_j(x_i) \right)
$$  (5)
Candidate list
Since all root nodes see all the learning examples, they would produce the same loss reduction. To increase diversity, we grow $T$ stumps and use the leaves to form the initial candidate list.

Initial bias
The initial bias $w_0$ is the best constant that fits the training set

$$w_0 = \arg \min_{y \in \mathbb{R}^K} \sum_{i=1}^{N} L(y_i, y)$$  \hfill (6)
Results — Experimental setting

1. Grow a forest of a thousand fully-developed Extremely randomized trees ($ET_{100\%}$) and count the number of nodes $M$.
2. Compare how different methods fare (in average over ten runs) under a constraint of 1% and 10% of that budget.

- $GIF_{x\%}$ grow the forest of a thousand trees until the node budget is met with the GIF algorithm.
- $RAND_{x\%}$ grow a forest of a thousand trees randomly.
- $ET_{x\%}$ grow only $10x$ fully-developed trees.

We used the following values for the hyper-parameters:

- Number of trees: 1000
- Candidate window size: 1
- Learning rate: $10^{-1.5}$
- Splitting algorithm: Extremely randomized trees (ET)
Results — Candidate window size

- Friedman1
- CT slice
- Twonorm
- Musk2

Candidate window size
+ 1  ▶ 10  ● 100  ■ 1000
Results — Regression

1% budget

- CT Slice
- Abalone
- Hwang F5
- Friedman1
- Cadata

10% budget

- ET
- GIF
- RAND

Relative average mean square error to ET_{100%}.
Results — Binary classification

1% budget

- Ringnorm
- Twonorm
- Hastie
- Musk2
- Madelon
- Mnist 8vs9

10% budget

- ET
- GIF
- RAND

Relative average misclassification rate to ET_{100%}.
Results — Multi-class problems

<table>
<thead>
<tr>
<th>Waveform</th>
<th>Mnist</th>
<th>Vowel</th>
<th>Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Relative average misclassification rate to ET_{100\%}.

- **ET_{x\%}**
- **GIF_{x\%}**
- **RAND_{x\%}**
Conclusion and future works

Performances

- GIF allows for lightweight yet accurate forests.
- Global optimization usually helps.
- Surprisingly, optimizing the shapes hurts.

TODOs

- Handle multiclass problems better.
- In depth comparison with Boosting methods.
GIF algorithm

1. **Input**: $D = (x_i, y_i)_{i=1}^N$, the learning set with $x_i \in \mathbb{R}^P$ and $y_i \in \mathbb{R}^K$; $\mathcal{A}$, the tree learning algorithm; $L$, the loss function; $B$, the node budget; $T$, the number of trees; $CW$, the candidate window size; $\lambda$, the learning rate.

2. **Output**: An ensemble $S$ of $B$ tree nodes with their corresponding weights.

3. **Algorithm**:

4. $S = \emptyset$; $C = \emptyset$; $t = 1$

5. $\hat{y}^{(0)}(.) = \arg\min_{y \in \mathbb{R}^K} \sum_{i=1}^N L(y_i, y)$

6. Grow $T$ stumps with $\mathcal{A}$ on $D$ and add the left and right successors of all stumps to $C$.

7. **repeat**

8. $C_t$ is a subset of size $\min\{CW, |C|\}$ of $C$ chosen uniformly at random.

9. Compute :

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

10. $S = S \cup \{(j^*, w^*_j)\}$; $C = C \setminus \{j^*\}$; $y^{(t)}(.) = y^{(t-1)}(.) + \lambda w^*_j z^*_j(.)$

11. Split $j^*$ using $\mathcal{A}$ to obtain children $j_l$ and $j_r$

12. $C = C \cup \{j_l, j_r\}$; $t = t + 1$

13. **until** budget $B$ is met
**GIF algorithm — Regularization**

**Node split**
Although the weight are optimized globally, the splitting elements of a node are still determined locally.

**Learning rate**
A learning rate $\lambda$ was introduced to prevent overfitting:

$$y^{(t)}(.) = y^{(t-1)}(.) + \lambda w^*_j z^*_j(.)$$  \hspace{1cm} (7)

**Candidate window**
Only a uniformly drawn subset of candidates are examined at each iterations.

- This also serves to speed up computations
Result — Error rates dropout with iteration

Friedman1: average test set error with respect to the budget $B \ (CW = +\infty$, $m = \sqrt{10}, \ T = 1000).$
Friedman1: cumulative node distribution with respect to the size-ranks
\( CW = \infty, \ m = \sqrt{10}, \ T = 1000, \ B = 10\% \)
Datasets

Table: Characteristics of the datasets. \( N \) is the learning sample size, TS stands for testing set, and \( p \) is the number of features.

| 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         |
| Madelon     | 2200   | 2200   | 500   | 2         |
| Madelon     | 2200   | 2200   | 500   | 2         |
| Madelon     | 2200   | 2200   | 500   | 2         |
| Madelon     | 2200   | 2200   | 500   | 2         |
| Waveform    | 3500   | 1500   | 40    | 3         |
| Vowel       | 495    | 495    | 10    | 11        |
| Mnist       | 50000  | 10000  | 784   | 10        |
| Letter      | 16000  | 4000   | 8     | 26        |