Joint learning and pruning of decision forests

Jean-Michel Begon JM.BEGON@ULG.AC.BE
Arnaud Joly A.JOLY@ULG.AC.BE
Pierre Geurts P.GEURTS@ULG.AC.BE
Systems and Modeling, Dept. of EE and CS, University of Liege, Belgium

Keywords: Random forest, (pre-)pruning, model compression, node budget, memory constraint

1. Motivation

Decision forests, such as Random forest (Breiman, 2001) and Extremely randomized trees (Geurts et al., 2006), are state-of-the-art supervised learning methods. Unfortunately, the size of the trees increases (at worst linearly) with the size of the training set, leading to substantial memory footprints when storing the forests. Several post-processing methods have been proposed to tackle this problem (Meinshausen et al., 2009; Joly et al., 2012; De Vleeschouwer et al., 2015; Ren et al., 2015). Although very effective, they require building the whole forest first and then solving a high-dimensional optimization problem to (post-)prune it. In this research, we would like to investigate whether a similar compression effect could be achieved directly during tree growing, without ever requiring the development of the whole forest. For that purpose, we propose a joint learning and pruning (JLP) method which iteratively and greedily deepens the trees by optimizing *globally* the sequence of nodes to develop, while still choosing *locally* the splitting variables and cut points at all tree nodes based on the standard score criterion. The algorithm is presented in Section 2 and some results and comparisons with more naive and post-pruning methods are reported in Section 3.

2. Joint learning and pruning

An ensemble of T trees can be expressed as a linear model in the "forest-space", a binary M-dimensional space where M is the total number of nodes in the whole forest (Joly et al., 2012). For instance, in regression, the prediction of the forest at some point x can be expressed as:

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

where the indicator function $z_j(x)$ is 1 if x reaches node j and 0 otherwise, and w_j is the prediction at a node j if j is a leaf and 0 otherwise.

Table 1. Joint learning and pruning (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 the left and right
   successors of all stumps to $C$.
3. For $k = 1$ to $K$:
   (a) Compute:
       $(j^*, w_j^*) = \arg\min_{j \in C, w \in \mathbb{R}} \sum_{i=1}^{N} \left(y_i - \left(\hat{y}^{(k-1)}(x_i) + w z_j(x_i)\right)\right)^2$
   (b) $S = S \cup \{(j^*, w_j^*)\}$; $C = C \setminus \{j^*\}$;
       $y^{(k)}(.) = y^{(k-1)}(.) + \lambda w_j^* z_{j^*}(.)$
   (c) Split $j^*$ using $\mathcal{A}$ to obtain children $j_l$ and $j_r$
   (d) $C = C \cup \{j_l, j_r\}$
```

The proposed joint learning and pruning (JLP) algorithm is described in Table 1. Starting from a constant model (the output average), it builds an additive model in the form of (1) by incrementally adding new node indicator functions while growing the forest. At each step, the node in a candidate list C that contributes the most to a decrease of the global error is selected (3.a) and introduced in the model via its indicator function z_{j^*} and its optimal weight w_j^* tempered by some learning rate λ (3.b). This node is then split locally according to the reference tree growing strategy $\mathcal{A}$ (3.c) and replaced by its two children in the candidate list (3.d). The process is stopped when the node budget K is reached. Implementation details are skipped for the sake of space but step 3.a is trivial and efficient to solve thanks to the hierarchical partitioning of the trees.

The learning rate λ is introduced to avoid overfitting. It has a direct impact on the way the forest is developed. Typically, large (resp. small) values of λ favor a more in depth (resp. in breadth) development of the

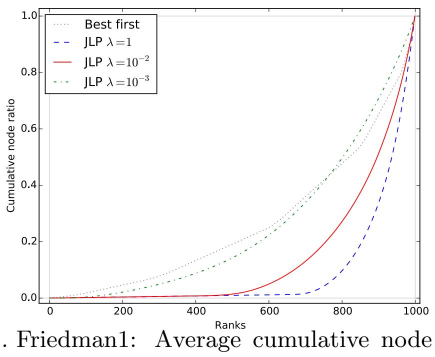

Figure 1. Friedman1: Average cumulative node distribution for BF and JLP with 1000 initial stumps and a budget of 10% compared to 1000 fully developed trees.

forest (see Figure 1).

Note that JLP is very similar to a least-square boosting algorithm (Hastie et al., 2009), where the set of base learners would be composed of node indicator functions and would be expanded at each iteration.

3. Results

We tested JLP on ten standard datasets[1], starting with $T = 1000$ stumps and a node budget K of 1% of the number of nodes in a forest of 1000 fully-developed trees. Table 2 compares JLP with a default learning rate of $10^{-1.5}$ against an uncompressed forest of 1000 trees ($ET_{100\%}$) and two baselines, $\mathrm{BF}_{1\%}$ and $\mathrm{ET}_{1\%}$, that both also divide the number of nodes by 100 wrt $ET_{100\%}$. BF, for best-first, grows the trees in parallel, splitting on the leaf which leads locally to the most important reduction of the total node impurity, until exhaustion of the budget. $ET_{1\%}$ simply grows a forest with only 10 fully-developed trees. Compared to JPL, Figure 1 shows that BF typically grows the forest more in breadth, while by construction $ET_{1\%}$ spends all its budget on growing deep trees. All methods use Extremely randomised trees with default setting for tree growing (Geurts et al., 2006).

JLP is clearly superior to BF on all problems and it is only inferior to $ET_{1\%}$ on Hill Valley. On Hwang F5, Ailerons and Cover type, JLP even surpasses the original forest of 1000 trees. Note that $\lambda = 10^{-1.5}$ is not necessarily optimal. For instance, on Abalone and Hwang F5, JLP reaches an error of 4.7 and 7.0×10^{-2} respectively for $\lambda = 10^{-2}$.

In Table 3, we also compare our approach to the L1-based post-pruning method (L1P) of Joly et al. (2012) that refits the linear model of the forest-space in (1) using the LASSO (Tibshirani, 1996). As expected, since L1P starts from the whole forests, it performs better than JLP with the same node budget. This improvement however comes with an important increase of the memory footprints and computing times (results not shown).

[1]Mnist has been binarized into zero-to-four and five-to-nine digits. Covertype has been binarized in pines (classes 2 and 3) versus non-pines.

Table 2. Average errors under memory constraint. Misclassification rate, in percent, for classification problems (datasets one to five). Mean square error for regression (datasets six to ten).

Datasets		$\mathrm{ET}_{100\%}$	$\mathrm{ET}_{1\%}$
Cover type		23.0 ± 0.2	23.4 ± 1.3
Hill Valley		41.8 ± 1.3	**42.4 ± 1.6**
Mnist		1.9 ± 0.2	**3.4 ± 0.2**
Ringnorm		2.9 ± 0.4	7.4 ± 0.6
Twonorm		3.1 ± 0.1	8.0 ± 0.6
Abalone		4.8 ± 0.2	5.3 ± 0.3
Ailerons	$\times 10^{-8}$	6.9 ± 0.2	7.1 ± 0.5
Cadata	$\times 10^{-1}$	2.5 ± 0.1	**2.8 ± 0.1**
Friedman1		4.9 ± 0.2	5.9 ± 0.3
Hwang F5	$\times 10^{-2}$	8.2 ± 0.1	8.7 ± 0.1
Datasets		$\mathrm{BF}_{1\%}$	$\mathrm{JLP}_{1\%,\lambda=10^{-1.5}}$
Cover type		29.9 ± 0.1	**19.0 ± 0.8**
Hill Valley		50.6 ± 1.4	43.5 ± 1.7
Mnist		12.5 ± 0.3	**3.4 ± 0.3**
Ringnorm		31.4 ± 18.0	**4.5 ± 0.4**
Twonorm		20.4 ± 16.9	**4.5 ± 0.3**
Abalone		5.5 ± 0.3	**5.0 ± 0.2**
Ailerons		16.4 ± 0.3	**5.3 ± 0.1**
Cadata		5.4 ± 0.1	**2.8 ± 0.1**
Friedman1		15.4 ± 0.5	**5.0 ± 0.3**
Hwang F5		15.2 ± 1.2	**7.6 ± 0.1**

Table 3. L1-based compression (initial forest of 1000 fully-developed trees compress to 1% of its size) compared to JLP with a budget of 1%.

Datasets	L1P	JLP	λ
Hill Valley	42.2 ± 2.1	42.8 ± 1.1	10^{-1}
Ringnorm	3.8 ± 0.4	4.5 ± 0.4	$10^{-1.5}$
Twonorm	5.1 ± 0.4	4.5 ± 0.3	$10^{-1.5}$
Ailerons $\times 10^{-8}$	4.0 ± 0.0	4.7 ± 0.1	$10^{-0.5}$
Friedman1	3.2 ± 0.3	5.0 ± 0.3	$10^{-1.5}$

4. Conclusion and future work

We have developed an algorithm which jointly learns and prunes tree-based ensemble models (JLP). The investigation so far indicates that JLP performs quite well under heavy memory constraints. Although it is slightly inferior to post-compression methods, JLP alleviates the need for building the forest first and allows for easier memory control.

A more systematic study of all the hyper-parameters has yet to be conducted. In addition to extending to multi-class problems, JLP could also be adapted to allow refitting of chosen nodes in order to produce lighter, and perhaps better, models.

Acknowledgments

Arnaud Joly is a recipient of a “aspirant” grant from the FNRS (Belgium) and acknowledges its financial support. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11.

References

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

De Vleeschouwer, C., Legrand, A., Jacques, L., & Hebert, M. (2015). Mitigating memory requirements for random trees/ferns. *Image Processing (ICIP), 2015 IEEE International Conference on* (pp. 227–231).

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*, 3–42.

Hastie, T. J., Tibshirani, R. J., & Friedman, J. H. (2009). *The elements of statistical learning : data mining, inference, and prediction.* Springer series in statistics. New York: Springer. Autres impressions : 2011 (corr.), 2013 (7e corr.).

Joly, A., Schnitzler, F., Geurts, P., & Wehenkel, L. (2012). L1-based compression of random forest models. *20th European Symposium on Artificial Neural Networks.*

Meinshausen, N., et al. (2009). Forest garrote. *Electronic Journal of Statistics*, *3*, 1288–1304.

Ren, S., Cao, X., Wei, Y., & Sun, J. (2015). Global refinement of random forest. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 723–730).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.