Analyzing the tree-layer structure of Deep Forests

Ludovic Arnould [1] **Claire Boyer** [1] **Erwan Scornet** [2]

Abstract

Random forests on the one hand, and neural networks on the other hand, have met great success in the machine learning community for their predictive performance. Combinations of both have been proposed in the literature, notably leading to the so-called Deep Forests (DF) (Zhou & Feng, 2017). In this paper, our aim is not to benchmark DF performances but to investigate instead their underlying mechanisms. Additionally, DF architecture can be generally simplified into more simple and computationally efficient shallow forests networks. Despite some instability, the latter may outperform standard predictive tree-based methods. We exhibit a theoretical framework in which a shallow tree network is shown to enhance the performance of classical decision trees. In such a setting, we provide tight theoretical lower and upper bounds on its excess risk. These theoretical results show the interest of tree-network architectures for well-structured data provided that the first layer, acting as a data encoder, is rich enough.

1. Introduction

Deep Neural Networks (DNNs) are among the most widely used machine learning algorithms. They are composed of parameterized differentiable non-linear modules trained by gradient-based methods, which rely on the backpropagation procedure. Their performance mainly relies on layer-by-layer processing as well as feature transformation across layers. Training neural networks usually requires complex hyper-parameter tuning (Bergstra et al., 2011) and a huge amount of data. Although DNNs recently achieved great results in many areas, they remain very complex to handle and unstable to input noise (Zheng et al., 2016).

Recently, several attempts have been made to consider networks with non-differentiable modules. Among them the Deep Forest (DF) algorithm (Zhou & Feng, 2017), which uses Random Forests (RF) (Breiman, 2001) as neurons, has received a lot of attention in recent years in various applications such as hyperspectral image processing (Liu et al., 2020), medical imaging (Sun et al., 2020), drug interactions (Su et al., 2019; Zeng et al., 2020) or even fraud detection (Zhang et al., 2019).

Since the DF procedure stacks multiple layers, each one being composed of complex nonparametric RF estimators, the rationale behind the procedure remains quite obscure. However DF methods exhibit impressive performances in practice, suggesting that stacking RFs and extracting features from these estimators at each layer is a promising way to leverage on the RF performance in the neural network framework. The goal of this paper is not an exhaustive empirical study of prediction performances of DF (see Zhou & Feng, 2019) but rather to understand how stacking trees in a network fashion may result in competitive infrastructure.

Related Works. Different manners of stacking trees exist (see Ghods & Cook, 2020, for a general survey on stacking methods), as the Forwarding Thinking Deep Random Forest (FTDRF), proposed by (Miller et al., 2017), for which the proposed network contains trees which directly transmit their output to the next layer (contrary to Deep Forest in which their output is first averaged before being passed to the next layer). A different approach by (Feng et al., 2018) consists in rewriting tree gradient boosting as a simple neural network whose layers can be made arbitrary large depending on the boosting tree structure. The resulting estimator is more simple than DF but does not leverage on the ensemble method properties of random forests.

In order to prevent overfitting and to lighten the model, several ways to simplify DF architecture have been investigated. (Pang et al., 2018) considers RF whose complexity varies through the network, and combines it with a confidence measure to pass high confidence instances directly to the output layer. Other directions towards DF architecture simplification are to play on the nature of the RF involved (Berrouachedi et al., 2019a) (using Extra-Trees instead of Breiman's RF), on the number of RF per layer (Jeong et al., 2020) (implementing layers of many forests with few trees), or even on the number of features passed between two consecutive layers (Su et al., 2019) by relying on an importance

*Equal contribution [1]LPSM, Sorbonne Universite [2]CMAP, Ecole Polytechnique. Correspondence to: Ludovic Arnould <ludovic.arnould@sorbonne-universite.fr>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

measure to process only the most important features at each level. The simplification can also occur once the DF architecture is trained, as in (Kim et al., 2020) selecting in each forest the most important paths to reduce the network time- and memory-complexity. Approaches to increase the approximation capacity of DF have also been proposed by adjoining weights to trees or to forests in each layer (Utkin & Ryabinin, 2017; Utkin & Zhuk, 2020), replacing the forest by more complex estimators (cascade of ExtraTrees) (Berrouachedi et al., 2019b), or by combining several of the previous modifications notably incorporating data preprocessing (Guo et al., 2018). Overall, the related works on DF exclusively represent algorithmic contributions without a formal understanding of the driving mechanisms at work inside the forest cascade.

Contributions. In this paper, we analyze the benefit of combining trees in network architecture both theoretically and numerically. As the performances of DF have already been validated by the literature (see Zhou & Feng, 2019), the main goals of our study are (i) to quantify the potential benefits of DF over RF, and (ii) to understand the mechanisms at work in such complex architectures. We show in particular that much lighter configuration can be on par with DF default configuration, leading to a drastic reduction of the number of parameters in few cases. For most datasets, considering DF with two layers is already an improvement over the basic RF algorithm. However, the performance of the overall method is highly dependent on the structure of the first random forests, which leads to stability issues. By establishing tight lower and upper bounds on the risk, we prove that a shallow tree-network may outperform an individual tree in the specific case of a well-structured dataset if the first encoding tree is rich enough. This is a first step to understand the interest of extracting features from trees, and more generally the benefit of tree networks.

Agenda. DF are formally described in Section 2. Section 3 is devoted to the numerical study of DF, by evaluating the influence of the number of layers in DF architecture, by showing that shallow sub-models of one or two layers perform the best, and finally by understanding the influence of tree depth in cascade of trees. Section 4 contains the theoretical analysis of the shallow centered tree network. For reproducibility purposes, all codes together with all experimental procedures are to be found in the supplementary materials.

2. Deep Forests

2.1. Description

Deep Forest (Zhou & Feng, 2017) is a hybrid learning procedure in which random forests are used as the elementary components (neurons) of a neural network. Each layer of DF is composed of an assortment of Breiman's forests and Completely-Random Forests (CRF) (Fan et al., 2003) and trained one by one. In a classification setting, each forest of each layer outputs a class probability distribution for any query point x, corresponding to the distribution of the labels in the node containing x. At a given layer, the distributions output by all forests of this layer are concatenated, together with the raw data. This new vector serves as input for the next DF layer. This process is repeated for each layer and the final classification is performed by averaging the forest outputs of the best layer (without raw data) and applying the `argmax` function. The overall architecture is depicted in Figure 1.

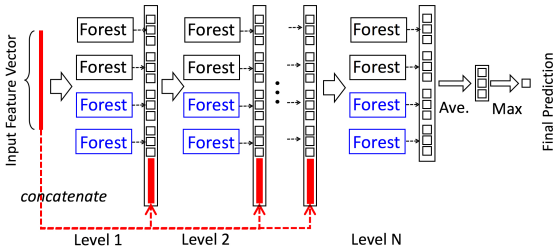


Figure 1. Deep Forest architecture (the scheme is taken from Zhou & Feng (2017)).

2.2. DF hyperparameters

Deep Forests contain an important number of tuning parameters. Apart from the traditional parameters of random forests, DF architecture depends on the number of layers, the number of forests per layer, the type and proportion of random forests to use (Breiman or CRF). In Zhou & Feng (2017), the default configuration is set to 8 forests per layer, 4 CRF and 4 RF, 500 trees per forest (other forest parameters are set to `sk-learn` (Pedregosa et al., 2011) default values), and layers are added until 3 consecutive layers do not show score improvement.

Due to their large number of parameters and the fact that they use a complex algorithm as elementary bricks, DF consist in a potential high-capacity procedure. However, as a direct consequence, the numerous parameters are difficult to estimate (requiring specific tuning of the optimization process) and need to be stored which leads to high prediction time and large memory consumption. Besides, the layered structure of this estimate, and the fact that each neuron is replaced by a powerful learning algorithm makes the whole prediction hard to properly interpret.

As already pointed out, several attempts to lighten the architecture have been conducted. In this paper, we will propose and assess the performance of a lighter DF configuration on tabular datasets.

Remark 1. *DF (Zhou & Feng, 2017) was first designed to classify images. To do so, a pre-processing network called Multi Grained Scanning (MGS) based on convolutions is first applied to the original images. Then the Deep Forest algorithm runs with the newly created features as inputs.*

3. Refined numerical analysis of DF architectures

In order to understand the benefit of using a complex architecture like Deep Forests, we compare different configurations of DF on six datasets in which the output is binary, multi-class or continuous, see Table 1 for description. All classification datasets belong to the UCI repository, the two regression ones are Kaggle datasets (Housing data and Airbnb Berlin 2020)[1]. Note that the Fashion Mnist features are built using the Multi Grained Scanning process from the DF original article (Zhou & Feng, 2017) (see A.3 for the encoding details).

Dataset	Type (Nb of classes)	Train/Val/Test Size	Dim
Adult	Class. (2)	26048/ 6512/ 16281	14
Higgs	Class. (2)	120000/ 28000/ 60000	28
Fashion Mnist	Class (10)	24000/ 6000/ 8000	260
Letter	Class. (26)	12800/ 3200/ 4000	16
Yeast	Class. (10)	830/ 208/ 446	8
Airbnb	Regr.	73044/ 18262/ 39132	13
Housing	Regr.	817/ 205/ 438	61

Table 1. Description of the datasets.

In what follows, we propose a light DF configuration. We show that our light configuration performance is comparable to the performance of the default DF architecture of Zhou & Feng (2017), thus questioning the relevance of deep models. Therefore, we analyze the influence of the number of layers in DF architectures, showing that DF improvements mostly rely on the first layers of the architecture. To gain insights about the quality of the new features created by the first layer, we consider a shallow tree network for which we evaluate the performance as a function of the first-tree depth.

3.1. Towards DF simplification

Setting. We compare the performances of the following DF architectures on the datasets summarized in Table 1:

(i) the default setting of DF, described in Section 2;

(ii) the best DF architecture obtained by grid-searching over the number of forests per layer, the number of trees per forest and the maximum depth of each tree. The selected architecture is chosen with respect to the performances achieved on validation datasets;

(iii) a new light DF architecture, composed of 2 layers, 2 forests per layer (one RF and one CRF) with only 50 trees of depth 30 trained only once;

(iv) the first layer of the best DF;

(v) the first layer of the light DF;

(vi) a "Flattened best DF as RF" which consists in one RF with as many trees as in the best DF with similar forest parameters (refer to Supplementary Materials A.2 and Table S3 for details);

(vii) a "Flattened light DF as RF" which corresponds to one RF with as many trees as in the light DF with similar forest parameters.

Results. Results are presented in Figures 2 and 3. Each bar plot respectively corresponds to the average accuracy or the average R^2 score over 10 tries for each test dataset; the error bars stand for accuracy or R^2 standard deviation. The description of the resulting best DF architecture for each dataset is given in Table S3 (Supplementary Materials).

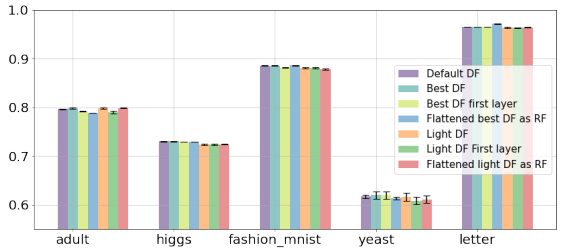


Figure 2. Accuracy of different DF architectures for classification datasets (10 runs per bar plot).

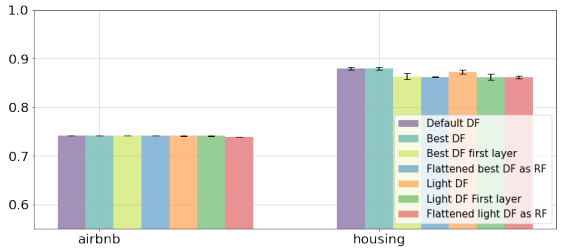


Figure 3. R^2 score of different DF architectures for regression datasets (10 runs per bar plot).

As highlighted in Figure 2, the performance of the light configuration for classification datasets is comparable to the default and the best configurations' one, while being much more computationally efficient: faster to train, faster at prediction, cheaper in terms of memory (see Table S2 in the Supplementary Materials for a comparison of computing time and memory consumption). Moreover, except on the

[1] https://www.kaggle.com/raghavs1003/airbnb-berlin-2020 https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

Letter dataset, the DF performs better than its RF equivalent. The results for the Letter dataset can be explained by the fact that the CRFs within the DF are outperformed by Breiman RFs in this specific case. Overall, for classification tasks, the small performance enhancement of Deep Forests (Default or Best DF) over our light configuration should be assessed in the light of their additional complexity. This questions the usefulness of stacking several layers made of many forests, resulting in a heavy architecture. We further propose an in-depth analysis of the role of each layer to the global DF performance.

3.2. Tracking the best sub-model

Setting. On all the previous datasets, we train a DF architecture by specifying the maximal number p of layers. Unspecified hyper-parameters are set to default value (see Section 2). For each p, we consider the truncated sub-models composed of layer 1, layer 1-2, ..., layer 1-p, where layer 1-p is the original DF with p layers. For each value of p, we consider the previous nested sub-models with $1, 2, \ldots, p$ layers, and compute the predictive accuracy of the best sub-model.

Results. We only display results for the Adult dataset in Figure 4 (all the other datasets show similar results, see Section A.5 of the Supplementary Materials). The score (accuracy or R^2-score) corresponds to the result on the test dataset. We observe that adding layers to the Deep Forest does not significantly change the accuracy score. Even if the variance changes by adding layers, we are not able to detect any pattern, which suggests that the variance of the procedure performance is unstable with respect to the maximal number of layers.

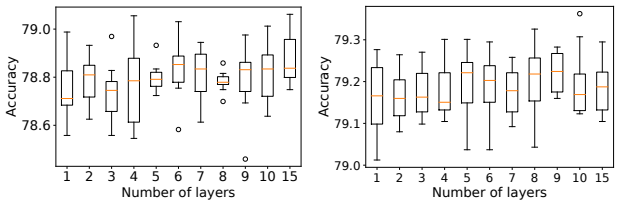


Figure 4. Adult dataset. Boxplots over 10 runs of the accuracy of a DF sub-model with 1 (Breiman) forest by layer (left) or 4 forests (2 Breiman, 2 CRF) by layer (right), depending on the maximal number of layers of the global DF model.

Globally, we observe that the sub-models with one or two layers often lead to the best performance (see Figure 5 for the Adult dataset and Supplementary Materials A.5). When the dataset is small (Letter or Yeast), the sub-model with only one layer (i.e. a standard RF or an aggregation of RFs) is almost always optimal since a single RF with no maximum depth constraint already overfits on most of these

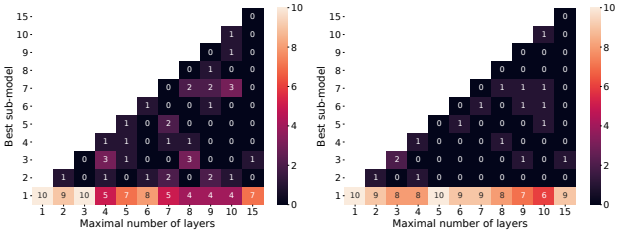


Figure 5. Adult dataset. Heatmap counting the optimal layer index over 10 tries of a default DF with 1 (Breiman) forest per layer (left) or 4 forests (2 Breiman, 2 CRF) per layer (right), with respect to the maximal number of layers. The number corresponding to (n, m) on the x- and y-axes indicates how many times out of 10 the layer m is optimal when running a cascade network with a maximal number n of layers.

datasets. Therefore the second layer, building upon the predictions of the first layer, entails overfitting as well, therefore leading to no improvement of the overall model. Besides, one can explain the predominance of small sub-models by the weak additional flexibility created by each layer: on the one hand, each new feature vector size corresponds to the number of classes times the number of forests which can be small with respect to the number of input features; on the other hand, the different forests within one layer are likely to produce similar probability outputs, especially if the number of trees within each forest is large. The story is a little bit different for the Housing dataset, for which the best submodel is between 2 and 6. As noticed before, this may be the result of the frustratingly simple representation of the new features created at each layer. Eventually, these numerical experiments corroborate the relevance of shallow DF as the light configuration proposed in the previous section.

We note that adding forests in each layer decreases the number of layers needed to achieve a pre-specified performance. This is surprising and is opposed to the common belief that in Deep Neural Networks, adding layers is usually better than adding neurons in each layer.

We can conclude from the empirical results that the first two layers convey the performance enhancement in DF. Contrary to NNs, depth is not an important feature of DFs. The following studies thus focus on two-layer architectures which are deep enough to reproduce the improvement of deeper architectures over single RFs.

3.3. A precise understanding of depth enhancement

In order to finely grasp the influence of tree depth in DF, we study a simplified version: a shallow CART tree network, composed of two layers, with one CART per layer.

Setting. In such an architecture, the first-layer tree is fitted on the training data. For each sample, the first-layer tree outputs a probability distribution (or a value in a regression setting), which is referred to as "encoded data" and given as input to the second-layer tree, with the raw features as well. For instance, considering binary classification data with classes 0 and 1, with raw features (x_1, x_2, x_3), the input of the second-layer tree is a 5-dimensional feature vector $(x_1, x_2, x_3, p_0, p_1)$, with p_0 (resp. p_1) the predicted probabilities by the first-layer tree for the class 0 (resp. 1).

For each dataset of Table 1, we first determine the optimal depth $k^\star$ of a single CART tree via 3-fold cross validation. Then, for a given first-layer tree with a fixed depth, we fit a second-layer tree, allowing its depth to vary. We then compare the resulting shallow tree networks in three different cases: when the (fixed) depth of the first tree is (i) less than $k^\star$, (ii) equal to $k^\star$, and (iii) larger than $k^\star$. We add the optimal single tree performance to the comparison.

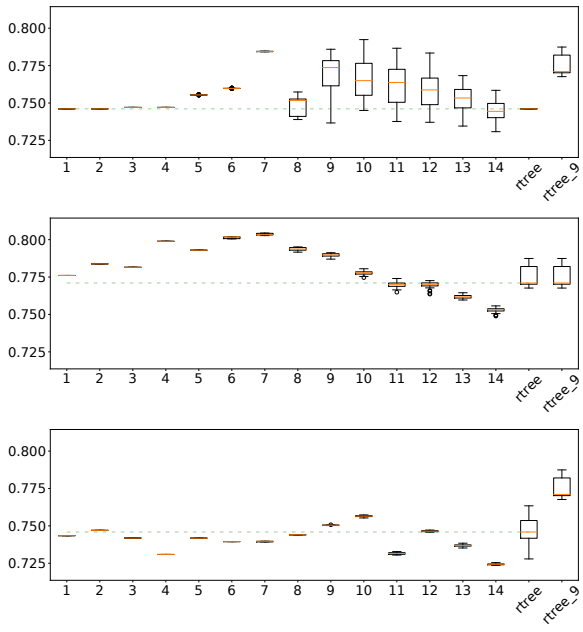


Figure 6. Adult dataset. Accuracy on the test dataset of a two-layer tree architecture w.r.t. the second-layer tree depth, when the first-layer (encoding) tree is of depth 2 (top), 9 (middle), and 15 (bottom). `rtree` is a single tree of respective depth 2 (top), 9 (middle), and 15 (bottom), applied on raw data. For this dataset, the optimal depth of a single tree is 9 and the tree with the optimal depth is depicted as `rtree_9` in each plot. The green dashed line indicates the median score of the `rtree`. All boxplots are obtained by 10 different runs.

Results. Results are displayed in Figure 6 for the Adult dataset only (see Supplementary Materials A.4 for the results on the other datasets). Specifically noticeable in Figure 6 (top), the tree network architecture can introduce performance instability when the second-layer tree grows (e.g. when the latter is successively of depth 7, 8 and 9).

Furthermore, when the encoding tree is not deep enough (top), the second-layer tree improves the accuracy until it approximately reaches the optimal depth $k^\star$. In this case, the second-layer tree compensates for the poor encoding, but cannot improve over a single tree with optimal depth $k^\star$. Conversely, when the encoding tree is more developed than an optimal single tree (bottom) - overfitting regime, the second-layer tree may not lead to any improvement, or worse, may degrade the performance of the first-layer tree. On all datasets, the second-layer tree is observed to always make its first cut over the new features (see Figure 7 and Supplementary Materials). In the case of binary classification, a single cut of the second-layer tree along a new feature yields to gather all the leaves of the first tree, predicted respectively as 0 and 1, into two big leaves, therefore reducing the predictor variance (cf. Figure 6 (middle and bottom)). Furthermore, when considering multi-label classification with n_{classes}, the second-layer tree must cut over at least n_{classes} features to recover the partition of the first tree (see Figure S15). Similarly, in the regression case, the second tree needs to perform a number of splits equal to the number of leaves of the first tree in order to recover the partition of the latter.

Figure 7. Adult dataset. Focus on the first levels of the second-layer tree structure when the first layer tree is of depth 9 (optimal depth). Raw features range from X[0] to X[13], X[14] and X[15] are the features built by the first-layer tree.

In Figure 6 (middle), one observes that with a first-layer tree of optimal depth, the second-layer tree may outperform an optimal single tree, by improving both the average accuracy and its variance. We aim at theoretically quantifying this performance gain in the next section.

4. Theoretical study of a shallow tree network

In this section, we focus on the theoretical analysis of a simplified tree network. Our aim is to exhibit settings in which a tree network outperforms a single tree. Recall that the second layer of a tree network gathers tree leaves of the first layer with similar distributions. For this reason, we believe that a tree network is to be used when the dataset has a very specific structure, in which the same link between the input and the output can be observed in different subareas

of the input space. Such a setting is described in Section 4.2

To make the theoretical analysis possible, we study centered trees (see Definition 1) instead of CART. Indeed, studying the original CART algorithm is still nowadays a real challenge and analyzing stacks of CART seems out-of-reach in short term. As highlighted by the previous empirical analysis, we believe that the results we establish theoretically are shared by DF. All proofs are postponed to the Supplementary Materials.

4.1. The network architecture

We assume to have access to a dataset $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ of i.i.d. copies of the generic pair (X, Y) with X living in $[0,1]^d$ and $Y \in \{0,1\}$ being the label associated to X.

Notations. Given a decision tree, we denote by $L_n(X)$ the leaf of the tree containing X and $N_n(L_n(X))$ the number of data points falling into $L_n(X)$. The prediction of such a tree at point X is given by

$$\hat{r}_n(X) = \frac{1}{N_n(L_n(X))} \sum_{X_i \in L_n(X)} Y_i$$

with the convention $0/0 = 0$, i.e. the prediction for X in a leaf with no observations is arbitrarily set to zero.

A shallow centered tree network. We want to theoretically analyze the benefits of stacking trees. To do so, we focus on two trees in cascade and will try to determine, in particular, the influence of the first (encoding) tree on the performance of the whole tree network. To catch the variance reduction property of tree networks already emphasized in the previous section, we consider a regression setting: let $r(x) = \mathbb{E}[Y|X = x]$ be the regression function and for any function f, its quadratic risk is defined as $R(f) = \mathbb{E}[(f(X) - r(X))^2]$, where the expectation is taken over $(X, Y, \mathcal{D}_n)$.

Definition 1 (Shallow centered tree network)**.** *The shallow tree network consists in two trees in cascade:*

- ***(Encoding layer)*** *The first-layer tree is a* cycling centered tree of depth k. *It is built independently of the data by splitting recursively on each variable, at the center of the cells. The first cut is made along the first coordinate, the second along the second coordinate, etc. The tree construction is stopped when exactly k cuts have been made. For each point X, we extract the empirical mean $\bar{Y}_{L_n(X)}$ of the outputs Y_i falling into the leaf $L_n(X)$ and we pass the new feature $\bar{Y}_{L_n(X)}$ to the next layer, together with the original features X.*
- ***(Output layer)*** *The second-layer tree is a* centered tree of depth k' *for which a cut can be performed at the center of a cell along a raw feature (as done by the encoding tree) or along the new feature $\bar{Y}_{L_n(X)}$. In this latter case, two cells corresponding to $\{\bar{Y}_{L_n(X)} < 1/2\}$ and $\{\bar{Y}_{L_n(X)} \geq 1/2\}$ are created.*

The resulting predictor composed of the two trees in cascade, of respective depth k and k', trained on the data $(X_1, Y_1), \ldots, (X_n, Y_n)$ is denoted by $\hat{r}_{k,k',n}$.

The two cascading trees can be seen as two layers of trees, hence the name of the shallow tree network. Note in particular that $\hat{r}_{k,0,n}(X)$ is the prediction given by the first encoding tree only and outputs, as a classical tree, the mean of the Y_i's falling into a leaf containing X.

4.2. Problem setting

Data generation. The data X is assumed to be uniformly distributed over $[0,1]^d$ and $Y \in \{0,1\}$. Let $k^\star$ be a multiple of d and let $p \in (1/2, 1]$. We build a regular partition of the space with *cells* $C_1, \ldots, C_{2^{k^\star}}$ of generic form

$$\prod_{k=1}^{d} \left[\frac{i_k}{2^{k^\star/d}}, \frac{i_k + 1}{2^{k^\star/d}} \right),$$

for $i_1, ..., i_d \in \{0, ..., 2^{k^\star/d} - 1\}$. We arbitrary assign a color (black or white) to each cell, which has a direct influence on the distribution of Y in the cell. More precisely, for x in a given cell C,

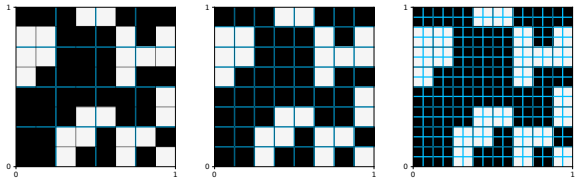
$$\mathbb{P}[Y = 1 | X = x] = \begin{cases} p & \text{if } C \text{ is a black cell,} \\ 1 - p & \text{if } C \text{ is a white one.} \end{cases} \tag{1}$$

We define $\mathcal{B}$ (resp. $\mathcal{W}$) as the union of black (resp. white) cells and $N_{\mathcal{B}} \in \{0, \ldots, 2^{k^\star}\}$ (resp. $N_{\mathcal{W}}$) as the number of black (resp. white) cells. Note that $N_{\mathcal{W}} = 2^{k^\star} - N_{\mathcal{B}}$. The location and the numbers of the black and white cells are arbitrary. This distribution corresponds to a *generalized chessboard* structure. The whole distribution is thus parameterized by $k^\star$ ($2^{k^\star}$ is the total number of cells), p and $N_{\mathcal{B}}$. Examples of this distribution are depicted in Figures 8 and 9 for different configurations and $d = 2$.

Why such a structured setting? The data distribution introduced above is highly structured, which can be seen as a restrictive study setting. However, the generalized chessboard is nothing but a discretized quantification of the regression function r using only 2 values (see Equation (1)). Going further than quantification towards general discretization does not seem appropriate for tree networks. To see this, consider a more general distribution such as

$$\mathbb{P}[Y = 1 | X = x] = P_{ij} \text{ when } x \in C_{ij},$$

where P_{ij} is a random variable drawn uniformly in $[0,1]$.



(a) Depth 4 (b) Depth 6 (c) Depth 8

Figure 8. Arbitrary chessboard data distribution for $k^\star = 6$ and $N_{\mathcal{B}} = 40$ black cells (p is not displayed here). Partition of the (first) encoding tree of depth 4, 6, 8 (from left to right) is displayed in blue. The optimal depth of a single centered tree for this chessboard distribution is 6.

Lemma 1. *Consider the previous setting with $k \geq k^\star$. In the infinite sample setting, the risks of a single tree and a shallow tree network are given by $R(\hat{r}_{k,0,\infty}) = 0$ and*

$$R(\hat{r}_{k,1,\infty}) \geq \frac{1}{48}\left(1 - \frac{8}{2^{k^\star} - 1}\right) + \frac{1}{2^{2^{k^\star}}}\frac{9}{24}.$$

Lemma 1 highlights the fact that a tree network has a positive bias, which is not the case for a single tree. Besides, by letting $k^\star$ tend to infinity (that is the size of the cells tends to zero), the above chessboard distribution boils down to a very generic classification framework. In this latter case, the tree network performs poorly since its risk is lower bounded by $1/48$. In short, when the data distribution is disparate across the feature space, the averaging performed by the second tree leads to a biased regressor. Note that Lemma 1 involves a shallow tree network, performing only one cut on the second layer. But similar conclusions could be drawn for a deeper second-layer tree, until its depth reaches $k^\star$. Indeed, considering $\hat{r}_{k,k^\star,\infty}$ would result in an unbiased regressor, with comparable performances as of a single tree, while being much more complex.

Armed with Lemma 1, we believe that the intrinsic structure of DF and tree networks makes them useful to detect similar patterns spread across the feature space. This makes the generalized chessboard distribution particularly well suited for analyzing such behavior. The risk of a shallow tree network in the infinite sample regime for the generalized chessboard distribution is studied in Lemma 2.

Lemma 2. *Assume that the data follows the generalized chessboard distribution described above with parameter $k^\star$, $N_{\mathcal{B}}$ and p. In the infinite sample regime, the following holds for the shallow tree network $\hat{r}_{k,k',n}$ (Definition 1).*

(i) ***Shallow encoding tree.*** *Let $k < k^\star$. The risk of the shallow tree network is minimal for all configurations of the chessboard if the second-layer tree is of depth $k' \geq k^\star$ and if the $k^\star$ first cuts are performed along raw features only.*

(ii) ***Deep encoding tree.*** *Let $k \geq k^\star$. The risk of the shallow tree network is minimal for all configurations of the chessboard if the second-layer tree is of depth $k' \geq 1$ and if the first cut is performed along the new feature $\bar{Y}_{L_n(X)}$.*

In the infinite sample regime, Lemma 2 shows that the preprocessing is useless when the encoding tree is shallow ($k < k^\star$): the second tree cannot leverage on the partition of the first one and needs to build a finer partition from zero.

Lemma 2 also provides an interesting perspective on the second-layer tree which either acts as a copy of the first-layer tree or can simply be of depth one.

Remark 2. *The results established in Lemma 2 for centered-tree networks also empirically hold for CART ones (see Figures 6,S12,S15,S17,S19,S21: (i) the second-layer CART trees always make their first cut on the new feature and always near 1/2; (ii) if the first-layer CART is biased, then the second-layer tree will not improve the accuracy of the first tree (see Figure 6 (top)); (iii) if the first-layer CART is developed enough, then the second-layer CART acts as a variance reducer (see Figure 6, middle and bottom).*

4.3. Main results

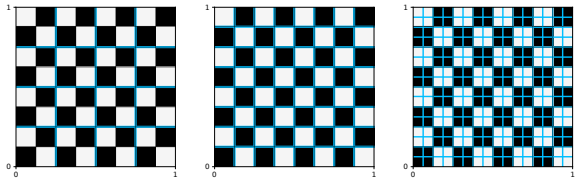
Building on Lemma 1 and 2, we now focus on a shallow network whose second-layer tree is of depth one, and whose first cut is performed along the new feature $\bar{Y}_{L_n(X)}$ at $1/2$. Two main regimes of training can be therefore identified when the first tree is either shallow ($k < k^\star$) or deep ($k \geq k^\star$).

In the first regime ($k < k^\star$), to establish precise non-asymptotics bounds, we study the balanced chessboard distribution (see Figure 9). Such a distribution has been studied in the unsupervised literature, in order to generate distribution for X via copula theory (Ghosh & Henderson, 2002; 2009) or has been mixed with other distribution in the RF framework (Biau et al., 2008). Intuitively, this is a worst-case configuration for centered trees in terms of bias. Indeed, if $k < k^\star$, each leaf contains the same number of black and white cells. Therefore in expectation the mean value of the leaf is $1/2$ which is non informative.

Proposition 3 (Risk of a single tree and a shallow tree network when $k < k^\star$)**.** *Assume that the data is drawn according to a balanced chessboard distribution with parameters $k^\star$, $N_{\mathcal{B}} = 2^{k^\star - 1}$ and $p > 1/2$ (see Figure 9).*

1. *Consider a single tree $\hat{r}_{k,0,n}$ of depth $k \in \mathbb{N}^\star$. We have,*

$$R(\hat{r}_{k,0,n}) \leq \left(p - \frac{1}{2}\right)^2 + \frac{2^k}{2(n+1)} + \frac{(1-2^{-k})^n}{4};$$



(a) Depth 4 (b) Depth 6 (c) Depth 8

Figure 9. Chessboard data distribution for $k^\star = 6$ and $N_{\mathcal{B}} = 2^{k^\star - 1}$. Partition of the (first) encoding tree of depth 4, 6, 8 (from left to right) is displayed in blue. The optimal depth of a single centered tree for this chessboard distribution is 6.

and

$$R(\hat{r}_{k,0,n}) \geq \left(p - \frac{1}{2}\right)^2 + \frac{2^k}{4(n+1)} + \frac{(1-2^{-k})^n}{4}\left(1 - \frac{2^k}{n+1}\right).$$

2. *Consider the shallow tree network $\hat{r}_{k,1,n}$. We have*

$$R(\hat{r}_{k,1,n}) \leq \left(p - \frac{1}{2}\right)^2 + \frac{2^{k/2+3}(p-\frac{1}{2})}{\sqrt{\pi n}} + \frac{7 \cdot 2^{2k+2}}{\pi^2(n+1)}(1+\varepsilon_{k,p}) + \frac{p^2+(1-p)^2}{2}\left(1-2^{-k}\right)^n$$

where $\varepsilon_{k,p} = o(2^{-k/2})$ uniformly in p, and

$$R(\hat{r}_{k,1,n}) \geq \left(p - \frac{1}{2}\right)^2.$$

First, note that our bounds are tight in both cases ($k < k^\star$ and $k \geq k^\star$) since the rates of the upper bounds match that of the lower ones. The first statement in Proposition 3 quantifies the bias of a single tree of depth $k < k^\star$: the term $(p - 1/2)^2$ appears in both the lower and upper bounds, which means that no matter how large the training set is, the risk of the tree does not tend to zero. The shallow tree network suffers from the same bias term as soon as the first-layer tree is not deep enough. Here, the flaws of the first-layer tree transfer to the whole network. In all bounds, the term $(1-2^{-k})^n$ corresponding to the probability of X falling into an empty cell is classic and cannot be eliminated for centered trees, whose splitting strategy is independent of the dataset.

Proposition S8 in the Supplementary Materials extends the previous result to the case of a random chessboard, in which each cell has a probability of being black or white. The same phenomenon is observed: the bias of the first layer tree is not reduced, even in the infinite sample regime.

In the second regime ($k \geq k^\star$), the tree network may improve over a single tree as shown in Proposition 4.

Proposition 4 (Risk of a single tree and a shallow tree network when $k \geq k^\star$)**.** *Consider a generalized chessboard with parameters $k^\star$, $N_{\mathcal{B}}$ and $p > 1/2$.*

1. *Consider a single tree $\hat{r}_{k,0,n}$ of depth $k \in \mathbb{N}^\star$. We have*

$$R(\hat{r}_{k,0,n}) \leq \frac{2^k p(1-p)}{n+1} + \left(p^2 + (1-p)^2\right)\frac{(1-2^{-k})^n}{2},$$

and

$$R(\hat{r}_{k,0,n}) \geq \frac{2^{k-1}p(1-p)}{n+1} + \left(p^2 + (1-p)^2 - \frac{2^k p(1-p)}{n+1}\right)\frac{(1-2^{-k})^n}{2}.$$

2. *Consider the shallow tree network $\hat{r}_{k,1,n}$. Letting*

$$\bar{p}^2_{\mathcal{B}} = \left(\frac{N_{\mathcal{B}}}{2^{k^\star}}p^2 + \frac{2^{k^\star} - N_{\mathcal{B}}}{2^{k^\star}}(1-p)^2\right)(1-2^{-k})^n,$$

we have

$$R(\hat{r}_{k,1,n}) \leq 2 \cdot \frac{p(1-p)}{n+1} + \frac{2^{k+1}\varepsilon_{n,k,p}}{n} + \bar{p}^2_{\mathcal{B}},$$

where $\varepsilon_{n,k,p} = n(1 - \frac{1-e^{-2(p-\frac{1}{2})^2}}{2^k})^n$, and for all $n \geq 2^{k+1}(k+1)$,

$$R(\hat{r}_{k,1,n}) \geq \frac{2p(1-p)}{n} - \frac{2^{k+3}(1-\rho_{k,p})^n}{n} + \bar{p}^2_{\mathcal{B}},$$

where $0 < \rho_{k,p} < 1$ depends only on p and k.

Proposition 4 shows that there exists a benefit from using this network when the first-layer tree is deep enough. In this case, the risk of the shallow tree network is $O(1/n)$ whereas that of a single tree is $O(2^k/n)$. In presence of complex and highly structured data (large $k^\star$ and similar distribution in different areas of the input space), the shallow tree network benefits from a variance reduction phenomenon by a factor 2^k. These theoretical bounds are numerically assessed in the Supplementary Materials (see Figures S35 to S40) showing their tightness for a particular choice of the chessboard configuration.

Finally, note that although the dimension d does not explicitly appear in our bounds, it is closely related to $k^\star$. Indeed, in high dimensions, modelling the regression function requires a finer partition, hence a direct relation of the form $k^\star \gg d$. Therefore, obtaining an unbiased estimator with a reduced variance as in Proposition 3 is more stringent in high dimensions, since it requires to choose $k \geq k^\star \gg d$.

5. Conclusion

In this paper, we study both numerically and theoretically DF and its elementary components. We show that stacking layers of trees (and forests) may improve the predictive performance of the algorithm. However, most of the improvements rely on the first DF-layers. We show that the performance of a shallow tree network (composed of single CART) depends on the depth of the first-layer tree. When the first-layer tree is deep enough, the second-layer tree may build upon the new features created by the first tree by acting as a variance reducer.

To quantify this phenomenon, we propose a first theoretical analysis of a shallow tree network (composed of centered trees). Our study exhibits the crucial role of the first (encoding) layer: if the first-layer tree is biased, then the entire shallow network inherits this bias, otherwise the second-layer tree acts as a good variance reducer. One should note that this variance reduction cannot be obtained by averaging many trees, as in RF structure: the variance of an averaging of centered trees with depth k is of the same order as one of these individual trees (Biau, 2012; Klusowski, 2018), whereas two trees in cascade (the first one of depth k and the second of depth 1) may lead to a variance reduction by a 2^k factor. This highlights the benefit of tree-layer architectures over standard ensemble methods. We thus believe that this first theoretical study of this shallow tree network paves the way of the mathematical understanding of DF.

First-layer trees, and more generally the first layers in DF architecture, can be seen as data-driven encoders. More precisely, the first layers in DF create an automatic embedding of the data, building on the specific conditional relation between the output and the inputs, therefore potentially improving the performance of the overall structure. Since preprocessing is nowadays an important part of all machine learning pipelines, we believe that our analysis is interesting beyond the framework of DF.