Dynamic Random Forests

Simon Bernard, Sébastien Adam, Laurent Heutte

Dynamic Random Forests

Simon Bernard, Sébastien Adam, Laurent Heutte

University of Rouen, LITIS EA 4108
BP 12 - 76801 Saint-Etienne du Rouvray, France
{*simon.bernard,laurent.heutte,sebastien.adam*}*@univ-rouen.fr*

Abstract

In this paper, we introduce a new Random Forest (RF) induction algorithm called Dynamic Random Forest (DRF) which is based on an adaptative tree induction procedure. The main idea is to guide the tree induction so that each tree will complement as much as possible the existing trees in the ensemble. This is done here through a resampling of the training data, inspired by boosting algorithms, and combined with other randomization processes used in traditional RF methods. The DRF algorithm shows a significant improvement in terms of accuracy compared to the standard static RF induction algorithm.

Keywords: Random Forests, Ensemble of Classifiers, Random Feature Selection, Dynamic Induction.

1. Introduction

The Random Forest (RF) algorithms form a family of classification methods that rely on the combination of several decision trees. The particularity of such Ensembles of Classifiers (EoC) is that their tree-based components are grown from a certain amount of randomness. Based on this idea, RF is defined as a generic principle of randomized ensembles of decision trees. Although this idea has already been exploited during the 90's ([1, 2, 3, 4]), the formal definition and the use of the term "Random Forest", have been introduced in 2001 in a founding paper written by Leo Breiman ([5]). The definition is as following:

Definition 1. A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\boldsymbol{x}, \Theta_k), k = 1, ..., L\}$ where $\{\Theta_k\}$ are independent and

identically distributed random vectors and each tree $h(\boldsymbol{x}, \Theta_k)$ casts a unit vote for the most popular class at input $\boldsymbol{x}$.

The reference RF algorithm, called Breiman's RF in the following, has been introduced by Breiman in [5]. It uses two randomization principles: bagging [1] and Random Feature Selection (RFS). This latter principle introduces randomization in the choice of the splitting test designed for each node of the tree. This choice is usually based on an impurity measure that is used as a criterion to determine the best feature for the partition of the current node into several child nodes. RFS randomly selects for each node a subset of features that will be withdrawn for this choice.

RF have shown to be particularly competitive with state-of-the-art learning methods, such as boosting which is known to be one of the most efficient learning principles ([5, 6]). Since their introduction in 2001, RF have been studied in many ways. For example, some works have proposed to further randomize the tree growing procedure, in comparison with Breiman's RF. One of the most successful methods is the Extra-trees algorithm ([7]) that relies on the Random Feature Selection technique, but with a random choice of the cut point associated to the feature selected for the splitting test. Another example is the PERT algorithm ([8]) that sets the cut point midway between two training instances randomly sampled. Other research works have proposed RF induction techniques based on manipulations of the description space. The Rotation Forest method ([9]) for example, combines several randomization techniques to build a sub-problem dataset which is projected into a new feature space through a Principal Component Analysis (PCA). Finally, efforts have been also brought on new combination operators, by weighting for example the tree votes for each new data point according to an estimation of the ability of each tree to correctly classify similar instances ([10, 11, 12]).

Despite these efforts and even if better performance have sometimes been reported, all these methods have the first drawback that the number of trees has to be set *a priori*, ideally to a very high value in order to obtain reasonably good performance. A second drawback is that some trees may decrease the forest performance as shown in two recent studies ([13, 14]), since trees are arbitrarily (independently) added to the forest. We believe that RF would certainly benefit from adopting a slightly different approach for the forest induction by the way each tree is added to the ensemble. While all RF algorithms add trees independently from one another, we highlight in this

paper the interest of a dynamic induction of the forest where trees are grown by taking into account the sub-forest already built. In that way, only reliable trees are intended to be grown in the forest. Thus, the main contribution of this paper is to offer a new dynamic RF induction algorithm, called Dynamic Random Forest (DRF), that compares favorably to the reference algorithm, *i.e.* Breiman's RF.

The rest of the paper is organized as follows. In section 2 we motivate the interest of designing a dynamic RF induction algorithm and detail our choices for implementing a first variant of DRF. Then, section 3 presents an evaluation of this algorithm and a comparison with Breiman's RF. In section 4, we discuss general pros and cons of DRF over Breiman's RF. Finally, conclusions and suggestions for future work directions are drawn in the last section.

2. Dynamic Random Forest

As stated in definition 1, a classical RF induction procedure grows trees independently from one another. Hence, each new tree is arbitrarily added to the forest. One can therefore wonder if all those trees contribute to the performance improvement. In [14], we have shown that using classical RF induction algorithms, some trees make the ensemble performance decrease, and that a well selected subset of trees can outperform the initial forest. Figure 1 illustrates this statement by showing the evolution of error rates obtained with a sequential tree selection process called SFS (Sequential Forward Search [15]) applied on an existing 300-trees forest built with Breiman's RF. This selection technique starts with an empty set and iteratively adds a classifier according to a given criterion (e.g. the error rate obtained on a validation dataset). At each iteration of the SFS process, each classifier in the pool of candidate classifiers is evaluated and the one that optimizes the performance of the ensemble is kept. The results presented in [14] (part of them illustrated on figure 1 for some datasets) show that there always exists at least one sub-forest that significantly outperforms the initial one, sometimes with ten times less trees. The performance of the RF could therefore be improved by removing from the ensemble well selected trees. The idea with DRF is to avoid the induction of trees that could make the forest performance decrease, by forcing the algorithm to grow only trees that would suit to the ensemble already grown.

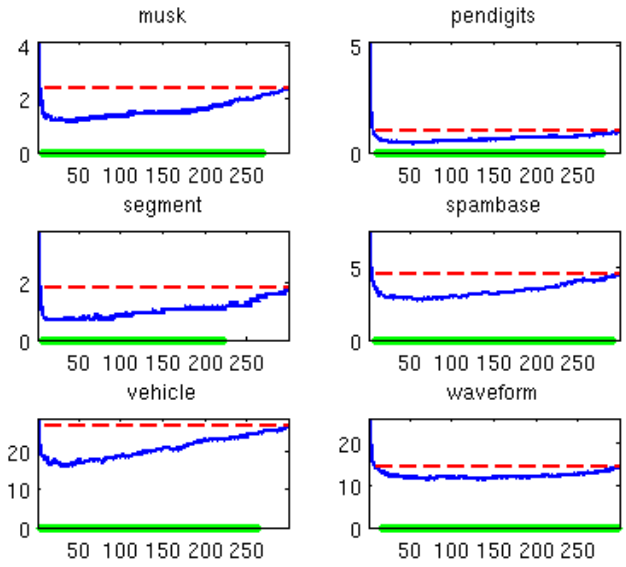

Figure 1: 10-fold cross-validation average error rates (in %) obtained during the SFS selection process on 6 datasets test (excerpt from experimental results given in [14]). The blue curves represent the error rates obtained with SFS according to the number of trees selected in the 300-trees forest the error rate of which is represented by the red dashed line. The straight green line on the x-axis indicates the range of sub-forests that obtained a statistically significant performance improvement compared to the 300-trees forest and according to a McNemar.

We show in this paper that the RF induction could benefit from adopting a dynamic approach, that is to say by making the tree induction dependent of the ensemble under construction. For that purpose, it is necessary to guide the tree induction by bringing to the process some "information" from the sub-forest already built. To do so, the idea behind the Dynamic Random Forest (DRF) algorithm is to perform a resampling of the training data that is halfway between bagging and boosting : proceed first to a random sampling with replacement of N instances, where N stands for the initial number of training instances (bagging), and then reweight the data (some kind of boosting). The reason for this choice is to keep using the two efficient randomization processes (*i.e.* bagging and Random Feature Selection) of Breiman's RF, and to improve RF accuracy by using the adaptative resampling principle of boosting.

Hence, what is performed in DRF is a weighting of each randomly selected training instance according to the predictions given by all the trees already added to the forest. Therefore, to determine the contribution of each training instance for the induction of the next tree, it is necessary to evaluate the ability of the current forest to predict its class. Thus, one possible measure can be a ratio of trees in the existing forest that have predicted the true class. This ratio is defined by:

$$c(\boldsymbol{x}, y) = \frac{1}{\mid h_{oob} \mid} \times \sum_{h_i \in h_{oob}} I(h_i(\boldsymbol{x}) = y) \tag{1}$$

Where $I(.)$ is the indicator function; $\boldsymbol{x}$ is an input data point and y its true class; $h_i(\boldsymbol{x})$ is an equivalent notation for $h(\boldsymbol{x}, \Theta_i)$, which represents the i-th classifier output; and h_{oob} stands for the set of out-of-bag trees of $\boldsymbol{x}$, *i.e.* the trees for which $\boldsymbol{x}$ is an out-of-bag instance. We recall that the out-of-bag instances are obtained with the bagging principle as follows: each tree is grown from a subset of training instances called a bootstrap sample, formed by random draws (with replacement) from the initial training dataset; the complementary subset of instances that have not been used for growing the tree, called out-of-bag instances, can easily be used as estimates of generalization capacities of the ensemble ([5]). The objective of computing $c(\boldsymbol{x}, y)$ by taking into account only the trees for which $\boldsymbol{x}$ belongs to the out-of-bag subset is to limit the risk of overfitting ([16]).

The lower the value of $c(\boldsymbol{x}, y)$, the more the next tree will have to focus on the instance $\boldsymbol{x}$, since it means that it was incorrectly classified by a large number of trees in the current forest. Consequently, the weighting function that will attribute a weight to $\boldsymbol{x}$ has to decrease with respect to $c(\boldsymbol{x}, y)$. For our experiments, we used the following function:

$$W(c(\boldsymbol{x}, y)) = 1 - c(\boldsymbol{x}, y) \tag{2}$$

We will give in the next section results with this linear function $W(c(\boldsymbol{x}, y))$ but other functions could be used to obtain the same effect (see [17]).

The DRF process is detailed in Algorithm 1. The same weight (initially equal to $\frac{1}{N}$) is assigned to all the training instances during the initialisation step. Thus, the first tree is induced in a traditional way, *i.e.* by taking into account each instance identically during the tree induction process. Then,

after the first tree has been built, weights are modified so that they are increased for instances wrongly classified by this first tree, and decreased for well classified instances. The second tree is consequently trained from the "re-weighted" training instances, selected in the new bootstrap sample. At the third iteration, these weights are re-calculated thanks to the predictions of the first two trees: maximum weights are given to instances wrongly classified by both the first and the second trees, and minimum weights are given to instances well classified by those two trees. The third tree is trained on the re-weighted data and so on. Note that the weight of a given input data point $\boldsymbol{x}$ is modified at each iteration, only if at least one out-of-bag tree is available for it. In Algorithm 1, this is checked via the $OoBTrees$ function, which is defined for accessing to the ensemble of out-of-bag trees of a given $\boldsymbol{x}$.

As shown in Algorithm 1, DRF are built with trees grown using a Random Tree induction algorithm. Rather than using Breiman's algorithm in which the number of randomly selected features, K, is a parameter of the algorithm that is constant and *a priori* fixed, we use the Random Tree induction algorithm proposed in [18]. In this induction algorithm, the Random Feature Selection technique is still used since it appears to be efficient as shown in [5, 6, 17, 18] but the number of features is randomly selected at each node according to the mean intrinsic information brought by each feature. Hence, given a bootstrap sample made up of randomly sampled training instances from the initial training set, the Random Tree induction procedure is the following one :

- Measure the information gain value of each feature;
- Compute the ratio p of features that exhibit the lowest information gains;
- For each internal node of the tree:
  - if p is superior to a given threshold then pick a random value of K according to the uniform distribution between 1 and M, where M is the number of features.
  - else, pick a random value of K according to $\mathcal{N}(\sqrt{M}, M/50)$.
  - randomly select K features.
  - choose the splitting rule according to an impurity measure computed on class counts.

Algorithm 1 *Dynamic Random Forest*

Require: T the training set $(\boldsymbol{x}_i, y_i)$
Require: N the number of training instances in T
Require: M the number of features
Require: L the number of trees in the forest to be built
Require: $W(c(\boldsymbol{x}, y))$ a weighting function inversely proportional to $c(\boldsymbol{x}, y)$ (see eq. 1)
Ensure: $forest$ the ensemble of trees that compose the forest

for all $\boldsymbol{x}_i \in T$ **do**
  $D_1(\boldsymbol{x}_i) \leftarrow \frac{1}{N}$
end for
for l from 1 to L **do**
  $T_l \leftarrow$ a bootstrap sample, made with randomly sampled (with replacement) training instances from T, according to a uniform distribution
  $T_l \leftarrow T_l$ weighted with D_l
  $tree \leftarrow RandomTree(T_l)$
  $forest \leftarrow forest \bigcup tree$
  $Z \leftarrow 0$
  for all $\boldsymbol{x}_i \in T$ **do**
    if $OoBTrees(\boldsymbol{x}_i) \neq \emptyset$ **then**
      $D_{l+1}(\boldsymbol{x}_i) \leftarrow W(c(\boldsymbol{x}_i, y_i))$
    else
      $D_{l+1}(\boldsymbol{x}_i) \leftarrow D_l(\boldsymbol{x}_i)$
    end if
    $Z \leftarrow Z + D_{l+1}(\boldsymbol{x}_i)$
  end for
  for all $\boldsymbol{x}_i \in T$ **do**
    $D_{l+1}(\boldsymbol{x}_i) \leftarrow \frac{D_{l+1}(\boldsymbol{x}_i)}{Z}$
  end for
end for
return $forest$

Note that in the last step of Random Tree induction procedure, the class counts are weighted by $W(c(\boldsymbol{x}, y))$. These counts initially consider that

training instances are equally important and thus are all associated to equal weights. By modifying these weights, we modify the way each instance will be taken into account in these class counts. The choice of the splitting test will consequently be modified as for the whole tree structure and thus the resulting model. For further explanations of K setting, please refer to Forest-RK proposed by the authors in [18].

As one can see in Algorithm 1, DRF procedure is inspired by boosting. Nevertheless, two differences should be highlighted. The first one is the use of randomization in the resampling process, *i.e.* performing bagging before boosting. More importantly, the second one concerns the way the weights are adapted: they are not updated according to the last induced tree only, but are entirely re-evaluated according to the whole forest at each iteration. The reason is that, unlike to boosting, we do not want the trees to be more and more specialized in previous prediction errors of the ensemble, but we want each tree to fit as well as possible the forest under construction. Boosting algorithms typically multiply the previous weights at each iteration by an update term that depends on the prediction errors of the last tree ([19]) : the weights progressively evolve to force the new trees to be more specialized in the classification of training instances particularly difficult to predict. This weighting strategy has the advantage to focus quickly on the wrongly classified instances, but has the drawback at the same time to lose the influence of the early trees and to make the ensemble more sensitive to noisy instances or outliers ([4]). On the contrary the key point of DRF as mentioned before is to keep a compromise between compensating previous errors and reinforcing correct predictions. Therefore, it is important to re-evaluate the weights according to the whole forest and not only to update them with the predictions of the last tree. This makes the evolution of the training sample weights slower and smoother than in boosting, thus rendering DRF less sensitive to noisy datasets.

All the choices described in this section define one procedure for the RF induction principle named Dynamic Random Forest. It actually establishes one example of how RF dynamic induction can be performed. However, as we will discuss in the conclusion, it is obvious that several other choices might work to induce accurate RF. Nevertheless, we will show in the following of this paper that this first algorithm of DRF is accurate and allows to significantly improve the traditional static RF induction.

3. Evaluation

To evaluate the DRF algorithm described in the previous section, we have compared it to the two following static RF induction methods: Breiman's reference method and a modified version called Forest-RK [18] that compares favorably with Breiman's RF. This section presents our experimental protocol to do it so, and discusses the results obtained.

3.1. Experimental Protocol

In this evaluation, we have focused on two goals:

1. Evaluate and compare the overall performance of the forests induced with both dynamic and static induction algorithms;
2. Monitor the error rate evolution to analyze the DRF behavior according to the number of trees, in comparison with static-induced RF results.

3.1.1. Datasets

For those experiments, 20 datasets have been selected. They are described in Table 1. The first 17 datasets in this table have been selected from the UCI repository ([20]), because they deal with different machine learning issues in terms of number of classes, number of features and number of samples. Three additional datasets on different handwritten digit recognition problems have been used: (i) the well-known MNIST database ([21]) with a 85 multiresolution density feature set ($1 + 2 \times 2 + 4 \times 4 + 8 \times 8$) built from greyscale mean values as explained in [22]; (ii) Digits and (iii) DigReject both described in [23], on which a 330-feature set has been extracted, made from three state-of-the-art descriptors, *i.e.* a 117-statistical/structural feature set [24], a 128-feature set extracted from the chaincode (contour-based) ([25]), and the same 85-feature set as for MNIST ([22])

3.1.2. Protocol

Each training subset has been used for building 3 different forests, with the following algorithms: Breiman's RF with $K = \sqrt{M}$, Forest-RK and DRF. For each forest, the test error rates have been monitored each time a new tree was added to the ensemble. For that purpose, each dataset has been randomly split in two subsets, with two thirds of the data dedicated to train the forest, and the other third to test it. This splitting procedure has been repeated 10 times so that for each dataset, 10 different random training subsets are at our disposal. We denote these subsets $T_i = (T_{r_i}, T_{s_i})$, with

Datasets	# instances	# features	# classes
Diabetes	768	8	2
Gamma	19020	10	2
Isolet	7797	616	26
Letter	20000	16	26
Madelon	2600	500	2
Mfeat-factors	2000	216	10
Mfeat-fourier	2000	76	10
Mfeat-karhunen	2000	64	10
Mfeat-zernike	2000	47	10
Musk	6597	166	2
OptDigits	5620	64	10
Page-blocks	5473	10	5
Pendigits	10992	16	10
Segment	2310	19	7
Spambase	4610	57	2
Vehicle	946	18	4
Waveform	5000	40	3
Digits	38142	330	10
DigReject	14733	330	2
Mnist	60000	85	10

Table 1: Description of the datasets

$i = [1..10]$ and where T_{r_i} stands for the training subset and T_{s_i} for the test subset. For each T_{r_i}, we have grown 3 forests (Breiman's RF, Forest-RK and DRF), made of 500 trees. Thus for each dataset, $3 \times 10 = 30$ forests of 500 trees have been built.

Usually, fixing the number of trees to 100 is considered to be reasonable for reaching the generalization error convergence with traditional RF algorithms ([5, 26, 14]). In our expeiments, we set this number to 500 since we could not predict *a priori* the test error convergence speed according to the number of trees for this kind of dynamic induction process and could not even predict if it actually converges. Indeed the proof of the generalization error convergence given by Breiman in [5] is based on the hypothesis that the random vectors which define the randomization principle (see definition 1)

are independent and identically distributed. But for the dynamic induction process these conditions are no longer verified, and the convergence is not proved anymore (this aspect is further discussed in section 4). Consequently, it was interesting to let the RF induction algorithm grow a large number of trees, in order to observe and compare convergence speed of the different algorithms tested.

Algorithm 2 sums up the whole experimental protocol. The next subsection presents and discusses the obtained results.

Algorithm 2 Experimental protocol

Require: L the number of trees in each forest
Require: N the number of training instances
Require: M the number of features
Ensure: $\epsilon_{BRF}[10]$ a 1D table for storing error rates of RF induced with Breiman's algorithm using $K = \sqrt{M}$.
Ensure: $\epsilon_{RK}[10][L]$ a 2D table for storing error rates of RF induced with Forest-RK.
Ensure: $\epsilon_{DRF}[10][L]$ a 2D table for storing error rates of RF induced with DRF.

1: **for** $i \in [1..10]$ **do**
2: Randomly draw without replacement $\frac{2}{3} \times N$ samples from the original dataset to form the training subset T_{r_i}. The remaining samples form the testing subset T_{s_i}. The couple (T_{r_i}, T_{s_i}) is noted T_i.
3: $h_{BRF}[i] \leftarrow BreimanForest(T_{r_i}, \sqrt{M}, L)$
4: $\epsilon_{BRF}[i] \leftarrow$ error rate of $h_{BRF}[i]$ on T_{s_i}
5: $h^0_{RK}[i] \leftarrow \emptyset$, $h^0_{DRF}[i] \leftarrow \emptyset$
6: **for** $l \in [1..L]$ **do**
7: $h^l_{RK}[i] \leftarrow h^{l-1}_{RK}[i] \bigcup RandomTree(T_{r_i})$
8: $\epsilon_{RK}[i][l] \leftarrow$ error rate of $h^l_{RK}[i]$ on T_{s_i}
9: $T^l_{r_i} \leftarrow$ weighted T_{r_i}, as in algorithm 1.
10: $h^l_{DRF}[i] \leftarrow h^{l-1}_{DRF}[i] \bigcup RandomTree(T^l_{r_i})$
11: $\epsilon_{DRF}[i][l] \leftarrow$ error rate of $h^l_{DRF}[i]$ on T_{s_i}
12: **end for**
13: **end for**

3.2. Results

Table 2 presents the average test error rates of the final forests made of 500 trees obtained with each of the 3 algorithms used.

Datasets	BRF	F-RK	DRF	Datasets	BRF	F-RK	DRF
Diabetes	25.25	24.86	**24.59**	Mfeat-zer	21.55	21.50	**21.05**
Digits	2.28	2.21	**2.10**	MNIST	4.97	4.95	**4.61**
DigReject	7.27	7.31	**6.56**	Musk	2.62	2.53	**2.41**
Gamma	11.99	12.09	**11.74**	OptDigits	1.65	1.60	**1.51**
Isolet	5.35	5.38	**4.96**	Page-blo	2.70	2.71	**2.64**
Letter	3.94	4.02	**3.31**	Pendigits	0.91	0.92	**0.87**
Madelon	32.78	**19.22**	22.66	Segment	2.33	2.41	**2.02**
Mfeat-fac	3.37	3.46	**2.89**	Spambase	4.88	5.03	**4.04**
Mfeat-fou	17.61	17.06	**16.83**	Vehicle	25.60	25.50	**25.07**
Mfeat-kar	3.36	3.39	**3.23**	Waveform	14.48	14.48	**14.35**

Table 2: Average error rates (in %) of the RF made of 500 trees, for the 3 RF induction algorithms. Lowest error rates have been marked in bold for each dataset.

One can observe that the DRF algorithm exhibits lower error rates than Breiman's RF and Forest-RK for 19 of the 20 datasets. One can however mention an atypical result for one of the 20 datasets: *madelon*. This phenomenom can be explained by the way this database has been built. This artificial dataset was originally built for a feature selection challenge ([27]) and contains data points grouped in 32 clusters, placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1; from these points, 20 informative features were extracted, while the 480 others are not discriminative at all. We think that this particularity of the feature space could explain the atypical results obtained with DRF and more generally with RF methods, since their performance depend on the intrinsic discriminative power of features chosen in the splitting test. For that reason, RF sometimes fail to filter irrelevant features, as explained for example in [7]. Despite this particular case, those results prove that DRF outperforms Forest-RK and the classical Breiman algorithm for all but one dataset.

The results presented so far where about the "final" random forests made of 500 trees. But one of the main objectives with the design of our dynamic algorithm is to obtain a performance improvement while trees are added to

the forest, faster and in a more straightforward way than with static RF induction. For studying the error variation during the forest construction, we represent in figure 2 the evolution of the test error rates according to the number of trees added throughout the RF induction processes, *i.e.* from the first tree up to the ensemble of 500 trees for both Forest-RK and DRF.

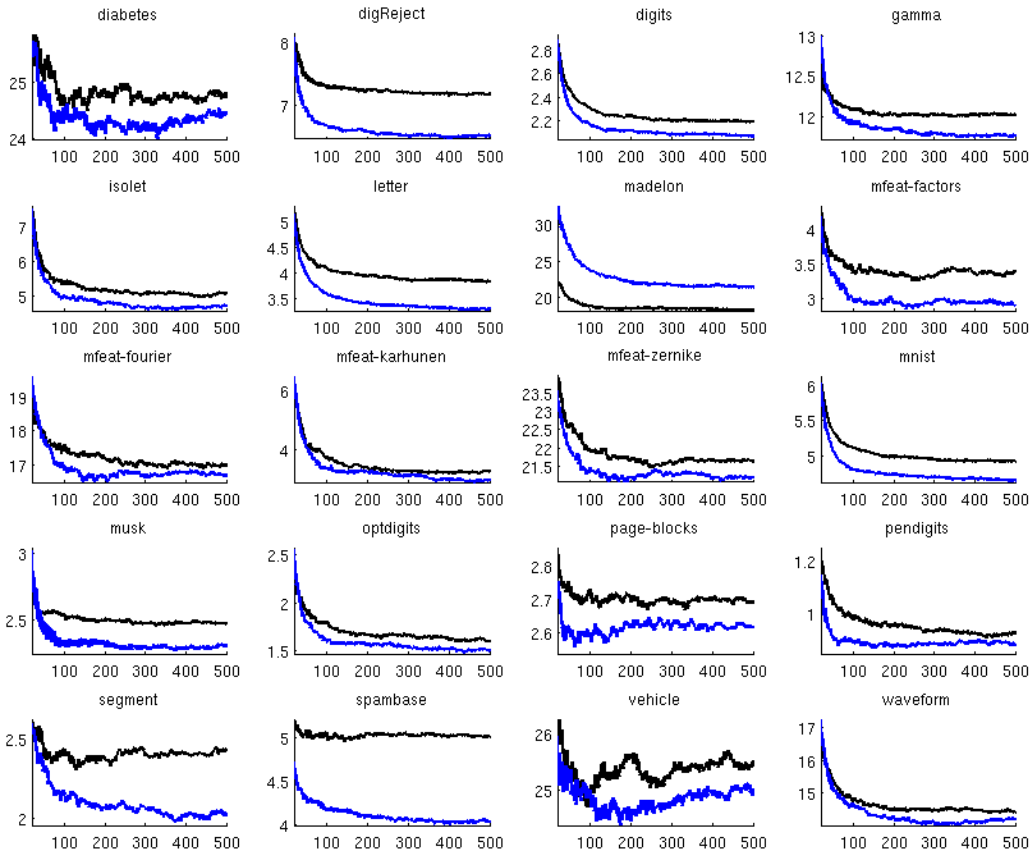

Figure 2: Evolution of the test error rates (in %) according to the number of trees added to the forest. Black curves represent the error rates obtained with Forest-RK and blue curves represent the error rates obtained with DRF

For all the datasets, one can observe that curves have similar trends for both algorithms, even when the convergence is not obvious, as it is the case for the smallest datasets *vehicle* and *diabetes*. Besides, for all the other datasets, *i.e.* those where a convergence seems to be obtained, one can further note

that the convergence is reached for the same number of trees. Hence, one can consider that DRF and Forest-RK have the same convergence properties.

Now, one can observe that for all but one dataset (*madelon*), the DRF error curve is always under the Forest-RK curve. This can be explained by a faster decrease of the DRF curves within the first iterations, i.e. from 1 to about 200 trees.

On can also remark from figure 2 that for most of the datasets, DRF error rates progressively reach an horizontal asymptote in the same way as Forest-RK. This has a practical interest since it means that designing a stopping criterion based on performance will be possible for the DRF induction. This stopping criterion has not been included yet in the DRF process, but it is an important mechanism to design. Besides, it is one of our priority future works. For example, it is possible to use the out-of-bag estimates to foresee the amount of trees above which no more improvement is obtained with new trees, as it is done in [28]. In that way, the number of trees in the final forest could not be fixed a priori as a parameter but could be determined by the out-of-bag error rate evolution.

4. Discussion

The preceding section has shown that Dynamic Random Forest outperforms static algorithms like Breiman's one and Forest-RK thanks to adaptative resampling. However one can wonder if the main advantages of Random Forest algorithms are still kept with DRF methods. In particular, it is well known that RF are fast both in training and decision but also that they do not overfit. In the following, we discuss these issues in the case of DRF. Moreover RF are frequently used to evaluate variable importance and proximities for feature selection issues. We show that DRF can be used also for that purpose.

4.1. Overfitting and computation time

One of the main losses of DRF, in comparison with Breiman's RF, is that two of its main mathematical properties are not proved anymore. Indeed, the proof of the convergence of RF and the computation of the bound of the generalization error defined in [5] depend on the pre-required condition that trees are grown independently. In the DRF process, it is not the case and consequently those properties are not proved anymore. Without the pre-required condition of independency of trees, it is no more obvious

to mathematically demonstrate that the generalization error convergence is always reached while trees are added to the ensemble, as it is proved with Breiman's RF. This is an important difference with Breiman's RF since, as a consequence, DRF are not proved to avoid overfitting while trees are added to the ensemble. Moreover, these new trees incorporate knowledge to better fit some training instances. Even if the criterion is computed on Out-of-Bag (OoB) error, which is considered to be a good estimate of the generalization error ([16]), one can wonder if the risk of overfitting is not nevertheless increased. DRF is inspired by boosting by resampling, for which overfitting has been largely discussed during the past fifteen years ([29, 30]). Formal definition of boosting implies that it is strongly sensitive to overfitting, since it is based on making each new component classifier better fit the training instances. However, one of the main suprising results with boosting is that it rarely overfits in practice. This is supposed to be due to the averaging of component classifier outputs ([30, 31]). The only overfitting situations that have been observed in our experiments occur on the datasets for which the ratio between the number of samples and the number of features is very small (*diabetes* and *vehicle*). This phenomenon is also frequently observed with boosting, as shown in [32]. Except these few overfitting situations, we believe that, as for boosting, DRF should rarely overfit. Moreover, although it would certainly need further investigations, we think that overfitting could be prevented with DRF thanks to a stopping criterion based on OoB error estimates : as OoB error rates are shown to give a reasonably good estimation of generalization error ([16]), it can be used to stop the induction of the forest, avoiding it to be based on the training error rate evolution.

As far as computation time is concerned, RF is also known to be very fast in learning and in prediction. It is an important property since it is often used in applications for which computation times are critical (like for example in applications on video tracking [33, 34]). In both Breiman's and DRF forests, time to predict a given input data point is proportional to the number of trees. As far as the learning stage is concerned, it is obvious that DRF takes longer to learn a single tree than in Breiman's RF, since additional computations are required for the weighting calculation. However, once the weights are computed, the tree growing in DRF is the same as a traditional random tree growing. Thus, the additional time for learning a DRF is proportional to its number of trees. As mentioned previously, one of the main advantage of DRF over Breiman's RF is to grow an accurate RF classifier with less trees.

Consequently, for reaching comparable accuracies with a RF, DRF will need to grow fewer trees than Breiman's RF.

4.2. Variable importance and proximities

RF methods have became popular during the last years because of their efficiency, but also thanks to the powerful tools they offer ([5, 10, 7]). Most of those tools are also available in DRF. For example, RF can be used to compute a variable importance measure, often set up for feature selection purposes ([35, 36, 37]). This measure is obtained once the whole forest has been grown, by first measuring the OoB error, and by then permuting values of the variable of interest in OoB instances and re-measuring the OoB error. The difference between the two values gives an evaluation of the variable importance. This process is completely preserved in DRF, since the only difference with Breiman's RF is in the learning process. Once built, DRF still allows to compute variable importance, without the dynamic process affecting the results.

In the same way, proximities evaluation matrix is a popular tool of Breiman's RF that can be used with DRF. It allows to give a $N * N$ matrix (N being the number of training instances) in which the element in line k and column l gives an evaluation of the similarity between instances k and l. It can also be used to compute the proximity of a new input point x to each of the training instances used to grow the forest ([10]). It is obtained via implicit distances between data points underlined by the tree partitioning. This mechanism is still enabled with DRF, since, even if the dynamic resampling of DRF affects the tree growing, the resulting structure is still a correct partitioning of the input space. Thus, nodes (and particularly leaves) of the trees still underline proximities between instances belonging to them.

5. Conclusions

In this paper, a new method has been proposed for inducing Random Forest (RF) classifiers, named Dynamic Random Forest (DRF). It is based on a sequential procedure that builds an ensemble of random trees by making each of them dependent on the previous ones. This dynamic aspect of the DRF algorithm is inspired by the adaptative resampling process of boosting. The DRF algorithm exploits the same idea and combines it with the randomization processes used in "classical" RF induction algorithms.

The evaluation of a first variant of DRF has been led by comparing it to the Breiman reference algorithm, and a modified version named Forest-RK. Firstly, it shows that the DRF algorithm outperforms the static RF induction algorithms in 95% of the cases on the databases used in these experiments. Secondly, it shows that the DRF process allows to decrease the test error rate in a faster and more straightforward way than in static algorithms.

This later statement is supposed to be due to the adaptative resampling which makes the tree induction take into account the joint objectives of preserving the prediction confidence for well classified instances and of compensating prediction errors made by existing trees. The guidance method which has been chosen for that purpose has been inspired by boosting and seems quite natural to be used for this task. Nevertheless, one can imagine several other processes to have a tree induction be guided by the chosen criterion. For example, one can imagine applying the global idea of boosting to the features instead of or in complement to the training instances, in order to bias the choice of the splitting test. Our future works will concern the implementation of such a feature boosting-like process and a stopping criterion.

References

[1] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[2] T. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.

[3] Y. Amit, D. Geman, Shape quantization and recognition with randomized trees, Neural Computation 9 (1996) 1545–1588.

[4] T. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization, Machine Learning 40 (1998) 139–157.

[5] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.

[6] R. Banfield, L. Hall, K. Bowyer, D. Bhadoria, W.P.Kegelmeyer, S. Eschrich, A comparison of ensemble creation techniques, In The Fifth International Conference on Multiple Classifier Systems (2004) 223–232.

[7] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 36 (1) (2006) 3–42.

[8] A. Cutler, G. Zhao, Pert - perfect random tree ensembles, Computing Science and Statistics 33.

[9] J. Rodriguez, L. Kuncheva, C. Alonso, Rotation forest : A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10) (2006) 1619–1630.

[10] M. Robnik-Sikonja, Improving random forests, European Conference on Machine Learning, LNAI 3210, Springer, Berlin (2004) 359–370.

[11] A. Tsymbal, M. Pechenizkiy, P. Cunningham, Dynamic integration with random forests, The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2006) 801–808.

[12] E. Tripoliti, D. Fotiadis, M. Argyropoulou, An automated supervised method for the diagnosis of alzheimers disease based on fmri data using weighted voting schemes, 2008 IEEE International Workshop on Imaging Systems and Techniques (2008) 340–345.

[13] S. Bernard, L. Heutte, S. Adam, Influence of hyperparameters on random forests accuracy, Workshop onf Multiple Classifier Systems (2009) 171–180.

[14] S. Bernard, L. Heutte, S. Adam, On the selection of decision trees in random forests, International Joint Conference on Neural Network (2009) 302–307.

[15] H. Hao, C. Liu, H. Sako, Comparison of genetic algorithm and sequential search methods for classifier subset selection., Seventh International Conference on Document Analysis and Recognition 2 (2003) 765–769.

[16] L. Breiman, Out-of-bag estimation, Technical Report, Department of Statistics, University of California, Berkeley.

[17] S. Bernard, Random forests : From the study of behaviors to dynamic induction, Ph.D. Thesis (in french), University of Rouen, France.

[18] S. Bernard, L. Heutte, S. Adam, Forest-RK: A new random forest induction method, in: D.-S. Huang, D. C. W. II, D. S. Levine, K.-H. Jo (Eds.), Proceedings of the International Conference on Intelligent Computing (ICIC'08), Vol. 5227 of Lecture Notes in Computer Science, Springer, 2008, pp. 430–437.

[19] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, International Conference on Machine Learning (1996) 148–156.

[20] A. Frank, A. Asuncion, UCI machine learning repository (2010).
URL `http://archive.ics.uci.edu/ml`

[21] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[22] S. Bernard, L. Heutte, S. Adam, Using random forests for handwritten digit recognition, International Conference on Document Analysis and Recognition (2007) 1043–1047.

[23] C. Chatelain, L. Heutte, T. Paquet, A two-stage outlier rejection strategy for numerical field extraction in handwritten documents, International Conference on Pattern Recognition, Honk Kong, China 3 (2006) 224–227.

[24] L. Heutte, T. Paquet, J. Moreau, Y. Lecourtier, C. Olivier, A structural/statistical feature based vector for handwritten character recognition, Pattern Recognition Letters 19 (7) (1998) 629–641.

[25] F. Kimura, S. Tsuruoka, Y. Miyake, M. Shridhar, A lexicon directed algorithm for recognition of unconstrained handwritten words, IEICE Transactions on Information and System E77-D (7) (1994) 785–793.

[26] P. Latinne, O. Debeir, C. Decaestecker, Limiting the number of trees in random forests, 2nd International Workshop on Multiple Classifier Systems (2001) 178–187.

[27] I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh, Feature Extraction: Foundations and Applications, Vol. 207, Studies in Fuzziness and Soft Computing, 2004.

[28] R. Banfield, L. Hall, K. Bowyer, W. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2006) 173–180.

[29] R. Meir, G. Rätsch, An introduction to boosting and leveraging, Advanced lectures on machine learning (2003) 118–183.

[30] D. Mease, A. Wyner, Evidence contrary to the statistical view of boosting.

[31] L. Breiman, Arcing classifiers, The Annals of Statistics 26 (3) (1998) 801–849.

[32] L. Wolf, I. Martin, Robust boosting for learning from few examples, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR, 2005, pp. 359–364.

[33] A. Wang, G. Wan, Z. Cheng, S. Li, An incremental extremely random forest classifier for online learning and tracking (2009) 1433–1436.

[34] J. Gall, A. Yao, N. Razavi, L. V. Gool, V. Lempitsky, Hough forests for object detection, tracking, and action recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (11) (2011) 2188–2202.

[35] C. Strobl, A. L. Boulesteix, A. Zeileis, T. Hothorn, Bias in random forest variable importance measures: Illustrations, sources and a solution, BMC Bioinformatics 8 (25) (2007) .

[36] A. Verikas, A. Gelzinis, M. Bacauskiene, Mining data with random forests: A survey and results of new tests, Pattern Recognition 44 (2) (2011) 330–349.

[37] K. K. Nicodemus, On the stability and ranking of predictors from random forest variable importance measures, Briefings in Bioinformatics 12 (4) (2011) 369–373.