Learning with Ensembles of Randomized Trees : New Insights

Vincent Pisetta,
Pierre-Emmanuel Jouve, and Djamel A. Zighed

Rithme, 59 bd Vivier Merle 69003 Lyon
Fenics, 59 bd Vivier Merle 69003 Lyon
ERIC Laboratory, 5 avenue Pierre Mendes-France, 69500 Bron
vpisetta@rithme.eu,
pjouve@fenics.com,
abdelkader.zighed@univ-lyon2.fr

Abstract. Ensembles of randomized trees such as Random Forests are among the most popular tools used in machine learning and data mining. Such algorithms work by introducing randomness in the induction of several decision trees before employing a voting scheme to give a prediction for unseen instances. In this paper, randomized trees ensembles are studied in the point of view of the basis functions they induce. We point out a connection with kernel target alignment, a measure of kernel quality, which suggests that randomization is a way to obtain a high alignment, leading to possibly low generalization error. The connection also suggests to post-process ensembles with sophisticated linear separators such as Support Vector Machines (SVM). Interestingly, post-processing gives experimentally better performances than a classical majority voting. We finish by comparing those results to an approximate infinite ensemble classifier very similar to the one introduced by Lin and Li. This methodology also shows strong learning abilities, comparable to ensemble post-processing.

Keywords: Ensemble Learning, Kernel Target Alignment, Randomized Trees Ensembles, Infinite Ensembles.

1 Introduction

Ensemble methods are among the most popular approaches used in statistical learning. This popularity essentially comes from their simplicity and their efficiency in a large variety of real-world problems. Instead of learning a single classifier, ensemble methods first build several base classifiers, usually via a sequential procedure such as Boosting ([13], [15]), or a parallel strategy using randomization processes such as Bagging [2] or Stochastic Discrimination [21], and in a second phase, use a voting scheme to predict the class of unseen instances.

Because of their impressive performances, understanding the mechanisms of ensemble learning algorithms is one of the main priority in the machine learning community. Several theoretical works have connected the Boosting framework

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part III, LNAI 6323, pp. 67–82, 2010.
© Springer-Verlag Berlin Heidelberg 2010

with the very well known SVMs [30] highlighting the margin's maximization properties of both algorithms (see e.g [11] [14] [26]). Another popular theoretical framework comes from [15] who pointed out its connection with forward stagewise modelling leading to several improved Boosting strategies.

Ensembles using randomized processes suffer from a lack of well defined theoretical framework. Probably the most well-known result highlighting the benefits of such a strategy is due to Breiman [5] who showed that the performance of majority voting of an ensemble depends on the correlation between members forming the pool and their individual strength. Other notable works concern the study of the consistency of such algorithms [1].

In this paper, we go a step further [5] by analyzing the basis functions induced by an ensemble using a randomized strategy. As pointed out in [19], most ensemble methods can be seen as approaches looking for a linear separator in a space of basis functions induced by the base learners. In this context, analyzing the space of basis functions of an ensemble is of primary importance to better understand its mechanism. We specifically focus on studying the situation where base learners are decision trees. A lot of empirical studies have shown that this class of classifiers is particularly well-suited for ensemble learning (see e.g [12]).

More precisely, we show a close connection between randomized trees ensembles and Parzen window classifiers. Interestingly, the error of a Parzen window classifier can be bounded from above with a kernel quality measure. This results in a generalization bound for an ensemble of randomized trees and clearly highlights the role of diversity and individual strength on the ensemble performance. Moreover, the connection suggests potential improvements of classical trees ensembles strategies.

Our paper is organized as follows. In section 2, we review some basic elements concerning decision tree induction. We focus on the importance of regularization and we point out a connection between decision trees and Parzen window classifiers. We introduce the notion of kernel target alignment (KTA) [9], a kernel quality measure allowing to bound the error of a Parzen window classifier. Once those base concepts are posed, we will show that an ensemble of randomized trees generates a set of basis functions leading to a kernel which can have a high alignment, depending on the individual strength and correlation between base learners (section 3). Interestingly, the connection shows that increasing the amount of randomization leads to a more regularized classifier.

Based on those results, we present in section 4 two possibilities for improving the performance of a randomized trees ensemble. The first strategy consists in post-processing intensively the comittee using powerful linear separators. The second strategy builds an approximate infinite ensemble classifier and is very similar to the one presented in [23]. That is, instead of selecting a set of interesting basis functions as realized by an ensemble, we will fit a regularized linear separator in the (infinite dimensional) space of basis functions induced by all possible decision trees having a fixed number of terminal nodes. Experiments comparing all those approaches are presented in section 5. Finally in section 6 we conclude.

2 Single Decision Tree Learning

2.1 Decision Tree Induction

We consider the binary classification case specified as follows. We are given access to a set of n labeled examples $S = \{(x_1, y_1), ..., (x_n, y_n)\}$ drawn from some (unknown) distribution P over $X \times \{-1, +1\}$, where X is a d-dimensional abstract instance space composed of features $X_1, ..., X_d$ taking their values in $\Re$. The objective of any classification algorithm is to learn a function $f : X \to \{-1, +1\}$ whose generalization error rate $Pr_{(x,y)\equiv P}[f(x) \neq y]$ is as low as possible. Among the large variety of methods dedicated to this goal, decision trees are very popular thanks to their efficiency, their ability to capture non linear relations between inputs and output and essentially because they are easily interpretable.

A binary decision tree[1] consists in recursively partitioning the input space X by searching for the transversal cut which optimizes some predefined criterion. The algorithm starts with the root node containing all learning instances and looks for a split of the form $[X_j \in s_{jm} \ ; \ X_j \notin s_{jm}]$ where $s_{jm} = (l_{jm}; u_{jm}]$ represents a set of possible values of X_j defined by a lower and upper limit $l_{jm} < X_j \leq u_{jm}$. Two new nodes are then added to the tree, one containing instances respecting $[X_j \in s_{jm}]$ and the other instances respecting $[X_j \notin s_{jm}]$. The process is then repeated for each subset (instances in a current node) until a stopping criterion is satisfied. In this context, a decision tree can be seen as a feature mapping $\Phi : X \to F$ such that F is the space represented by the nodes of the tree. Each node represents a basis function $b_t(x)$ (i.e a dimension of F) taking the form of a conjunctive rule [16]:

$$b_t(x) = \prod_{j=1}^{d} I(X_j \in s_{jm}) \tag{1}$$

where $I(.)$ is the indicator function of the truth of its argument. We have $b_t(x) = 1$ if the instance x belongs to node t and $b_t(x) = 0$ elsewhere. Once the decision tree has been constructed, one can make a prediction by fitting a linear function f in F :

$$f(x) = \sum_{t=1}^{2T-1} \alpha_t b_t(x) + \alpha_0 \tag{2}$$

where T is the number of terminal nodes of the tree[2]. Classically a basis function associated to a terminal node has a weight α_t equal to 1 or -1 depending on the most frequent class of all examples belonging to it, while both α_0 and basis functions associated to non terminal nodes have a weight of 0.

[1] We restrict our framework to binary decision tree. Let us simply note that more complex structures such as n-ary trees can be implicitly constructed using binary trees (see e.g [19]).

[2] A binary decision tree has a total number of node equal to $2T - 1$ which explains the upper term of the sum.

A lot of different decision trees algorithms have been designed. The main differences between them concern the criterion used to find the "best" split and the strategy employed to stop the induction process. Several experimental studies have shown that the splitting criterion has not a very significant effect on the performance of the decision tree (see e.g [7], [29]). As pointed out in [6], the key of success lies in controlling the tree complexity. Consequently, most efficient decision tree algorithms such as CART [6] or C4.5 [25] particularly focus on tree regularization.

A general method to control model complexity consists in giving a penalty proportional to the model complexity. The goal of the induction process then becomes to find the best tradeoff between performance (small error rate) and complexity, which can be realized by minimizing a criterion of the following form

$$Criterion(f, \lambda) = L(y, f) + \lambda J(f), \quad \lambda \geq 0 \tag{3}$$

where $L(y, f)$ is a measure of the classifier error (a loss function) and $J(f)$ a functional penalty that should be large for complex functions f [19]. This kind of regularization, widely used in machine learning is known as Tikhonov regularization. In this spirit, Breiman [6] proposed to select $L(y, f)$ as the error rate and $J(f)$ as the number of terminal nodes T of the decision tree. Once a value of λ has been selected, [6] develop a tree of maximum depth using Gini index as splitting criterion and, in a second phase, prune the tree in order to minimizes (3) measured on a test set.

2.2 Connection with Parzen Window Classifier and Generalization Error

The classical predictive scheme (using only terminal nodes with a weight of $+1$ or -1) can interestingly be seen as a Parzen window estimator based on the kernel $K(x, x') = \sum_{t=1}^{T} b_t(x) b_t(x')$ with $b_t(x) = 1$ if x belongs to leaf t and 0 elsewhere. Note that the basis functions considered here are only those associated to terminal nodes. Here, K is a very sparse kernel since we have $K(x, x') = 1$ if x and x' are in the same terminal node and 0 if not. The Parzen window estimator consists in labeling x with $f(x) = sign(\sum_{i=1}^{n} y_i K(x, x_i))$ which is strictly equivalent to predict the most frequent label encountered in the terminal node in which x falls.

Cristianini et al.[9] have shown that the generalization error GE of the expected Parzen window estimator $f(x) = sign(E_{(x',y')}[y' K(x', x)])$ based on a kernel K is bounded from above with probability $1 - \delta$:

$$GE(f(x)) \leq 1 - \hat{A}\left(K, y^t y\right) + \sqrt{\frac{8}{n} \ln \frac{2}{\delta}} \tag{4}$$

where $\hat{A}\left(K, y^t y\right)$ is called (empirical) kernel target alignment (KTA) and is formally defined on a sample S as :

$$\hat{A}\left(K, y^t y\right) = \frac{\langle K, y^t y \rangle_F}{\sqrt{\langle K, K \rangle_F \langle y^t y, y^t y \rangle_F}} = \frac{\sum_{i,s=1}^{n} y_i y_s K(x_i, x_s)}{n \sqrt{\sum_{i,s=1}^{n} K(x_i, x_s)^2}} \tag{5}$$

where $\langle ., . \rangle_F$ is the Frobenius product. KTA was initially designed to reflect the goodness of a kernel and more generally the goodness of a similarity matrix. As we can see from (5), KTA calculates the sum of similarities between objects belonging to the same class and substracts the sum of similarities between objects belonging to distinct classes. This quantity is then normalized to obtain an indicator varying between -1 and 1.

The connection allows one to use the following regularization scheme. First, we can induce a tree having the purest possible nodes, and then, look for the subtree leading to the kernel that minimizes (4) or equivalently, that maximizes $\hat{A}(K, y^t y)$ on a test sample. Interestingly, this methodology is very similar to the pruning technique of CART which estimates the expected Parzen window error on a test sample and keep the subtree that minimizes a tradeoff between this error and the tree complexity. In KTA, the complexity of the tree is penalized implicitly because the effect of adding splits is reflected in the sparsity of K leading to a lower value of $\langle K, y^t y \rangle_F$.

2.3 Drawbacks of Single Decision Trees

While regularization is an essential feature of the success of decision trees, this is generally not enough to obtain strong learning abilities. The first problem comes from the greedy mechanism used to find a split. In some cases such as the well known XOR problem, it is difficult for the algorithm to find the best split because this implies finding an interaction between two (or more) features in one shot. The second problem is their high variance. It is well-known that small changes in data could lead to drastic changes in the decision tree. This phenomenon comes from their hierarchical structures which implies that an error at split k will be propagated down to all splits below it [19]. This latter problem is reflected in the basis functions induced by a single tree. If the algorithm does a "strong" mistake at a high level, i.e, at the top levels of the tree, it will be impossible for any regularization strategy based on pruning to obtain a good result.

The main question is how one can overcome these problems. An appealing solution lies in increasing the number of basis functions generated by the algorithm. However, in order to be efficient, this process should induce basis functions in a non-hierarchical manner, else the problem of high variance will remain. Trees ensembles are particularly well-suited to realize such a process. Indeed, they work by generating several decision trees which is equivalent to increase the number of generated basis functions in a non-hierarchical way. In the next section, we will see that randomized trees ensembles allow to increase the kernel target alignment previously introduced. Consequently they improve the classification accuracy compared to a single decision tree and act in the same time as regularizers.

In the same spirit, one can choose to select all basis functions generated by all possible decision trees having a predefined number of terminal nodes. While this seems a priori untractable due to the infinite number of possible basis functions, we will see (section 4) that a solution can nevertheless be found by embedding

the basis functions into an appropriate kernel. However, in this case, we should carefully regularize the classifier which will be realized using a classical Tikhonov regularization as in (3).

3 Randomized Trees Ensemble

3.1 Algorithms

One of the most efficient ways to improve the performance of a single decision tree is to construct several distinct decision trees and to aggregate them through a voting scheme. Examples of such procedures are Bagging [2], Random Forests [5], PERT [10] or Extremely Randomized Trees [17]. These algorithms are all particular cases of a more general methodology introducing randomization in the induction process of base learners. The skeleton of these techniques consists in repeating M times the following steps :

- *Step* 1 : Apply a sampling strategy on S to obtain a new sample S_m
- *Step* 2 : Induce a decision tree on S_m by searching recursively for the best split among a random subset of all possible ones[3].

Each individual learner predicts a class for an unseen instance x corresponding to the most frequent class of examples belonging to the terminal node in which x falls. The prediction associated to the ensemble corresponds to the most frequently voted class among the M decision trees. The differences between randomized ensembles algorithms comes from the sampling strategy used in step 1 the randomization process chosen to find a split in step 2. In the case of Bagging, S is sampled iteratively using a bootstrap strategy, while there is no randomization in the split's search. Random Forests work by sampling S via a bootstrap and look for the best split among a random subset of d' features ($d' \leq d$). Extremely randomized trees do not use any sampling scheme but look for the best split among d' cut points randomly chosen on d' features themselves randomly chosen. Because such procedures lead to classifers with a low variance, each individual tree is generally fully grown (i.e, until having the purest possible leaves) in order to reduce bias and consequently the generalization error [19].

The use of an ensemble has the effect to increase the number of generated basis functions compared to a single decision tree. Intuitively, this is interesting because it overcomes the problems due to the hierarchical nature of decision trees and to their greedy split's search. However, such a mechanism will work only if the basis functions are enough diverse and individually correlated to the output y. Indeed, if the trees are identical, the set of generated basis functions will be equivalent to the basis functions generated by a single decision tree leading to an unchanged prediction. The use of randomization procedures is then fully

[3] In [24], the authors define a slightly different way of introducing randomization into decision trees ensembles. Their definition allows to study the "spectrum" of randomization and give interesting insights about the effect of randomization on the performance of decision trees ensembles.

justified. Probably the most well-known theoretical justification is due to [5] who has shown that the generalization error GE of majority voting of an ensemble is bounded from above : $GE \leq \bar{\rho}(1-s^2)/s^2$ where $\bar{\rho}$ is the average correlation between base classifiers and s a function of their strength.

3.2 Connection with Kernel Target Alignment

We will see that the benefit of using randomized trees ensembles is also reflected through the KTA measured on the kernel induced by the ensemble. Let us assume that each tree of the ensemble is grown until having only pure terminal nodes. Note that this is always possible while there are not two examples with the same initial representation and distinct labels. It is well-known that in classification, the best results are generally obtained by growing each tree until having only pure nodes (see e.g [5]). Interestingly, in this case, the classical majority voting scheme is equivalent to a Parzen window estimator based on the kernel induced by the ensemble $K_{ens}(x,x') = M^{-1}\sum_{m=1}^{M}\sum_{t=1}^{T_m} b_{mt}(x)b_{mt}(x')$. Here, $b_{mt}(x)$ represents the basis function associated to the terminal node t of the tree m. In this context, the performance of the ensemble should be highly dependent on the KTA of K_{ens}. The main question is why introducing randomization in the induction of trees could lead to a higher KTA.

K_{ens} can equivalently be written $K_{ens}(x,x') = M^{-1}\sum_{m=1}^{M} K_m(x,x')$ where $K_m(x,x')$ is the kernel induced by the m^{th} tree in the same manner as in section 2.2. Note that dropping the constant M^{-1} has no effect on the prediction. In this point of view, we see that K_{ens} simply consists in summing several base kernels. In [9], the authors have shown that one can benefit from summing two kernels. Indeed, the alignment of the sum of two kernels K_1 and K_2 with the target is given by :

$$\hat{A}\left(K_1+K_2, y^t y\right) = \frac{\|K_1\|_F}{\|K_1+K_2\|_F}\hat{A}\left(K_1, y^t y\right) + \frac{\|K_2\|_F}{\|K_1+K_2\|_F}\hat{A}\left(K_2, y^t y\right)$$

The alignment of the sum will be high if both kernels have a high individual alignment and if their correlation, i.e, $\hat{A}(K_1,K_2) = \langle K_1,K_2\rangle_F / \|K_1\|_F \|K_2\|_F$ is low. Indeed, if the kernels are identical, we have $\hat{A}\left(K_1+K_2, y^t y\right) = \hat{A}\left(K_1, y^t y\right) = \hat{A}\left(K_2, y^t y\right)$ while if they are different, we have :

$$\frac{\|K_1\|_F}{\|K_1+K_2\|_F} + \frac{\|K_2\|_F}{\|K_1+K_2\|_F} > 1$$

leading to a potentially higher overall alignment. Note that if one uses M kernels, the alignment of sum will be equal to :

$$\hat{A}\left(\sum_{m=1}^{M} K_m, y^t y\right) = \sum_{m=1}^{M} \frac{\|K_m\|_F}{\left\|\sum_{m=1}^{M} K_m\right\|_F}\hat{A}\left(K_m, y^t y\right) \tag{6}$$

The effect of randomization will be to decrease a bit the average individual alignment of the kernels in order to decrease their correlation, i.e, to increase

$\|K_m\|_F / \left\|\sum_{m=1}^{M} K_m\right\|_F$. In most empirical studies on the relationship between ensemble performance and strength-correlation, the main problem is to measure the diversity [22]. Equation (6) clearly highlights the role of each component and a possible way of measuring them. While randomization aims at playing on diversity and individual strength, its exact role is more complex.

The reason comes from the concentration property of the alignment. As underlined in [9], if the kernel function is selected a priori, that is, if one do not learn the kernel, the value of the alignment measured on a sample S is highly concentrated around its expected value. As a consequence, building an ensemble of extremely randomized trees as realized by [17] leads to a kernel that would have quite the same alignment on the training sample and any test sample. However, learning too intensively the kernel, i.e, introducing few randomization in the tree induction will result in a larger difference and will be reflected in a lower expected alignment than one could wish to have. The direct implication is that introducing a high level of randomness leads to a more regularized classifier. This also shows that decreasing the amount of randomization in the induction of decision trees will not necessarily result in a higher individual expected alignment of a decision tree. Interestingly, in its experiments, Breiman [5] observed that increasing the number of features to find the best split did not necessarily lead to higher individual strength. The explanation in terms of alignment concentration may give a clue to these results. Experiments showing all those claims are presented in section 5.

4 Improved Randomized Trees Ensembles

In this section, we present two possible improvements of an ensembles of randomized trees. Here, we describe the theoretical aspects. Experiments will be presented in section 5.

4.1 Post-processing

Globally, randomized trees ensembles can be seen as powerful kernel constructors because they aim at increasing KTA through the introduction of randomization. While the alignment is directly connected to Parzen window estimator, [9] have shown experimentally that maximizing KTA is also a good strategy before employing more complex learners such as SVMs. Because randomized trees ensembles directly act on the kernel target alignment, it seems interesting to post-process them using a more complex learner than a simple Parzen window estimator. That is, instead of simply giving the same weight to all basis functions induced by the ensemble, one can learn "optimal weights" with an appropriate learning strategy. In this case however, we are no more protected against over-fitting because of the lack of links between the new learner and

KTA and should consequently employ a specific regularization. A possible way consists in searching a vector of weights $\hat{\alpha}$ such that [16]:

$$\{\hat{\alpha}\}_0^{|B|} = \arg \min_{\{\alpha\}_0^{|B|}} \sum_{i=1}^{n} L\left(y_i, \alpha_0 + \sum_{t=1}^{|B|} \alpha_t b_t(x_i)\right) + \lambda \sum_{t=1}^{|B|} |\alpha_t|^p \tag{7}$$

where B is the set of basis functions induced by the ensemble of decision trees[4]. Different parameterizations of $L(.)$ and p lead to classical statistical learners. For example, choosing $L(y_i, f(x_i)) = max(0, 1 - y_i f(x_i))$ (hinge loss), and $p = 2$ consists in solving the SVM optimization program [30] in the space of basis functions constructed by the tree, while choosing $L(.)$ as the hinge loss and $p = 1$ is equivalent to solve the LPBoost problem [11]. The choice of $L(.)$ is mainly dependent on the type of learning problem we are facing. Typically, in a regression setting, $L(.)$ is chosen as the square-loss function while in classification, we will tend to choose the hinge loss. The choice of regularization is a harder task since its effect is not yet fully understood. A well known difference is that constraining the coefficients in L_1 norm (i.e, $p = 1$) leads to sparser results than using the L_2 norm, i.e, most α_i will tend to be equal to 0 [27]. Note that the set of basis functions B can be chosen to be the set of basis functions associated to terminal nodes or the set of basis functions associated to all nodes (terminal and non terminal).

4.2 Generating an Infinite Set of Basis Functions

In case one works with a Tikhonov regularizer, an appealing strategy consists in considering not only basis functions induced by a finite ensemble, but all basis functions that could be induced by any decision tree and let a regularized learner as in (7) finding an optimal solution. The main problem here is that the program (7) will have infinitely many variables and finding a solution seems a priori untractable. However, as we will see, there are some possibilities to overcome this problem.

Consider the optimization problem as stated in (7). Choosing $p = 2$ and $L(y, f(x)) = max(0, 1 - yf(x))$ leads to the well-known SVM optimization problem. Most practical SVM implementations work on the dual formulation of (7)

$$\min_{\beta \in \Re^n} \frac{1}{2} \sum_{i=1}^{n} \sum_{s=1}^{n} \beta_i \beta_s y_i y_s K(x_i, x_s) - \sum_{i=1}^{n} \beta_i \tag{8}$$

$$s.t \quad 0 \leq \beta_i \leq 2/\lambda$$

$$\sum_{i=1}^{n} \beta_i y_i = 0$$

[4] As pointed out by a reviewer, the optimization problem presented in (7) can be seen as a particular case of Stacking [31]. The main difference is that instead of using classifiers as new features, we use decision trees' nodes.

K is a kernel function defined as $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$ where $\Phi(x)$ is obtained from the feature mapping $\Phi : X \rightarrow F$ where F is assumed to be a Hilbert space equipped with the inner product $\langle ., . \rangle$ [28]. The dual form has the great advantage to enable solving the SVM problem even if Φ maps examples into an infinite dimensional space. Indeed, we see from (8) that K is sufficient to find the optimal solution. This also means that the mapping Φ need not to be known if we are sure it exists, which is automatically achieved if the Gram matrix $\tilde{K}_{i,s} = K(x_i, x_s)$ of dimension $n \times n$ is always symmetric and semi-positive definite [28].

The use of an SVM makes possible solving (7) for infinitely many variables and consequently, to solve the SVM in the space represented by all possible basis functions induced by any decision tree. To do so, one must find a kernel function which embodies those basis functions. In section 3, we have seen that the kernel corresponding to the basis functions induced by an ensemble of decision trees is equal to the proportion of decision trees letting two examples x and x' sharing the same terminal node. The main difference here is that instead of simply counting the number of decision trees letting x and x' reaching the same leave, we must calculate the probability that x and x' end in the same terminal node considering all possible decision trees.

Interestingly, Breiman [4] has given an implicit answer to this question. Consider the following assumptions are met : 1) the space of features is entirely bounded, i.e, $X \subseteq (L_1, R_1) \times (L_2, R_2) \times ...(L_d, R_d)$ where $L_j \in \Re$ and $R_j \in \Re$, $1 \leq j \leq d$, are respectively the lowest and highest bounds of feature X_j ; 2) each split of each tree is selected at random, i.e, a feature is first randomly selected, and a split point is then randomly choosen along this feature according to an uniform distribution, 3) the probability measure P is uniform on the space. Then Breiman [4] showed that if one builds an infinity of decision trees having each T terminal nodes, the proportion of trees letting x and x' sharing the same terminal node is approximately :

$$K_{inf}(x, x') \approx \exp^{-\gamma \|x - x\|_1} \tag{9}$$

where $\gamma = log(T)/d$, where d the dimensionality of the initial feature space. Interestingly, this is the very well-known Laplacian kernel. A recent and very interesting paper of Lin and Li has pointed a similar result [23]. Their demonstration shows a strict equality between the kernel and the set of basis functions instead of just an approximation as in (9). However, the kernel is derived in a bit different perspective and the relation between the number of terminal nodes and kernel's sharpness (i.e, γ) is harder to capture. For these reasons, we will keep the framework based on [4]. With an appropriate γ, one could solve the SVM problem using the kernel K_{inf} to obtain an approximate infinite ensemble classifer working in the (infinite dimensional) space H_T embedding all basis functions induced by any tree with T terminal nodes.

Note that the basis functions embedded in K_{inf} corresponds to basis functions associated to terminal nodes of the trees. It is not evident a priori to choose a good γ. One could proceeds by evaluating the SVM with several values of

γ on a test sample or by cross-validation. However, another interesting way of proceeding consists in summing several Laplacian kernels in the following manner

$$K_{Sinf}(x, x') = \sum_{q=2}^{Q} \exp^{-\gamma_q \|x-x\|_1} \tag{10}$$

where $\gamma_q = log(q)/d$. It is well-known that summing two kernels embdedding respectively H_1 and H_2 leads to a kernel embedding $H_1 \cup H_2$ (see e.g [23]). As a consequence, solving an SVM program using the kernel K_{Sinf} in equation (8) leads approximately to search a linear separator in the space represented by all basis functions induced by any tree having a number of leaves from 2 to Q.

5 Experiments

5.1 KTA and Random Forests

To illustrate the link between KTA and the performance of a randomized trees ensemble, we have run the following experiments on the *Heart* and *Ionosphere* datasets coming from the UCI repository. *Heart* has 13 features and 270 examples, while *Ionosphere* has 351 examples and 34 features. We have randomly splitted each dataset into two subsets of equal size, one used as a training sample and the other as a test sample. One Random Forest of 100 trees [5] has been run for several possible d' (d' is the number of features evaluated to find the best split) on the training set. More specifically, we used $d' = \{1, 3, 5, ...33\}$ for *Ionosphere* and $d' = \{1, 2, 3, ..., 13\}$ for *Heart*. Each tree of a Forest was grown until having the purest possible terminal nodes. For each Random Forest, we have calculated its KTA and its error rate on the test sample as well as the KTA on the training sample. Fig. 1 shows the results averaged over 10 runs.

On the *Heart* dataset, we can note an interesting correlation between KTA and the forest's performance. Indeed, the lowest KTA values are met for $d' > 7$ which corresponds to the poorest performances of the ensemble. The maximal alignment is achieved for $d' = 5$ which is also the best performance of the ensemble. We clearly see that the higher the level of randomization, the lower the difference between KTA on the training sample and test sample. The results for *Ionosphere* are less conclusive. Indeed, the test error does not vary too much with d' as well as KTA. We can however note that the higher the level of randomization, the lower the difference between KTA on the learning sample and the test sample as expected.

Of course, more extensive experiments should be carried out to clearly test the relation between KTA and the error rate of a randomized trees ensemble. However, we believe that the framework presented here provides promising perspectives and must be further analyzed.

5.2 Comparison of Randomized Trees Ensembles

In this section, we compare the performances of the methodologies presented previously. Our benchmark consists of 10 real-world datasets coming from the

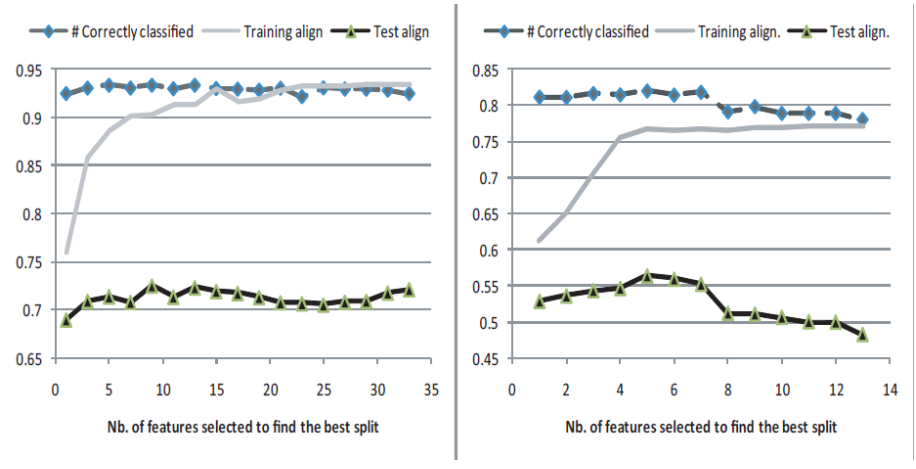

Fig. 1. KTA vs. performance of an ensemble. Left : Ionosphere dataset (KTA was multiplied by a factor 1.5 for better lisibility. Right : Heart dataset - KTA was multiplied by a factor 2.

UCI repository, three synthetic datasets (Twonorm, Threenorm and Ringnorm) coming from [3] and three datasets coming from the NIPS 2003 Feature challenge selection [18].

Four classifiers have been tested : a Random Forest of 100 trees [5] (**RF**), the same Random Forest whose basis funtions **associated to terminal nodes** are post-processed according to equation (7) with $p = 2$ and a hinge loss (**RF-L**), the same Random Forest whose basis functions **associated to all the nodes** are post-processed according to equation (7) with $p = 2$ and a hinge loss (**RF-N+L**) and finally an SVM trained with a kernel defined in (10) (**SVM-K**). We have chosen $p = 2$ and hinge loss for post-processed ensembles in order to have a fair comparison with the SVM trained with (10). The error rates of each method are averaged over 3 trials of 10-fold cross-validation.

All feature elements of all datasets were scaled to $[0, 1]$ even for RF. For RF, RF-L and RF-N+L, the trees forming the ensemble were grown until having the purest possible leaves. The parameters of all algorithms were searched as to minimize the error estimated by a 5-fold cross-validation on the training set as suggested in [20]. That is, for RF, RF-L and RF-N+L, d' was chosen within $\left\{1, \sqrt{d}, d\right\}$ which are references in Random Forests [5], and for RF-L, RF-N+L and SVM-K, the regularization parameter λ was searched within the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. Finally SVM-K was trained using 3 possible sums of 10 Laplacian kernels. The first sum was chosen to embedd decision trees of depth 1 to 10, the second trees of depth 4 to 13 and the third trees of depth 7 to 16. In binary trees, the relation between depth k and number of terminal nodes T is $T = 2^k$. Random Forests have been induced using our own implementation, while LIBSVM [8] has been used as SVM soft-margin solver. Results are shown in Table 1.

Table 1. Performances of 4 classifiers on several datasets. Results in bold are significantly better than others according to one-sided Student t-test at level 0.05.

Dataset	n	d	RF	RF-L	RF-N+L	SVM-K
Sonar	208	60	0.846	**0.894**	**0.894**	**0.905**
Breast	569	30	0.963	0.974	0.972	0.975
Ionosphere	351	34	0.940	0.940	0.940	0.945
Pima	768	8	0.754	**0.782**	**0.775**	**0.772**
Musk	476	166	0.917	0.927	0.948	**0.979**
Heart	270	13	0.807	**0.839**	**0.845**	**0.856**
Vote	435	16	0.960	0.958	0.958	0.940
Australian	690	14	0.801	**0.853**	**0.866**	**0.858**
Spambase	4601	57	0.969	**0.982**	**0.986**	**0.989**
Tic-Tac-Toe	958	9	0.991	1	1	0.996
Twonorm	300	20	0.973	0.971	0.971	0.972
Threenorm	300	20	0.836	0.832	0.832	0.832
Ringnorm	300	20	0.956	**0.972**	**0.972**	**0.974**
Dexter	600	20000	0.905	**0.938**	**0.939**	0.897
Arcene	200	10000	0.774	**0.836**	**0.836**	**0.829**
Gisette	7000	5000	0.968	0.978	0.978	0.961

The results show several interesting things. First, post-processing an ensemble of decision trees gives quite systematically lower error rates than non post processed ones. Secondly, RF-L and Rf-N+L share almost indistinguishable performances suggesting that the basis functions induced by terminal nodes are sufficient to learn well.

On both UCI and synthetic datasets, the performances of post-processed ensembles (RF-L and RF-N+L) and SVM-K are very close. Indeed, except on the Musk dataset, there are no statistical differences between both approaches. However, on the high-dimensional datasets coming from the NIPS feature selection challenge, post-processed ensembles tend to outperform SVM-K. This suggests that ensemble of randomized trees are well-suited for high-dimensional data.

Using an infinite ensemble has the advantage of better space covering resulting in a very smooth decision boundary. In the case of a finite ensemble induced by randomized trees strategies, the lack of smoothness seems to be compensated by the search of a subset of interesting basis functions. This can be seen as a feature selection operating directly in the infinite feature space of basis functions induced by all possible decision trees. This feature selection may be an explanation to the strong performance of randomized trees ensembles in very high dimensional data.

The use of a finite ensemble has also the great advantage to give interpretable results. Indeed, each basis function induced by a decision tree can be seen as a rule (see section 2). Post-processing will give a weight to each basis function (i.e, each rule) highlighting the most important ones. Note that one who focus on interpretability should normalize each weight by the support of the corresponding rule (the number of covered examples by the rule) in order to not give too much importance on rules covering a lot of examples (see eg. [16]).

6 Conclusion

We analyzed randomized trees ensembles through the basis functions they induce. We pointed out a connection with the kernel target alignment and Parzen window classifiers. The connection can be used to bound the generalization error of an ensemble and gives some insights about the performance of randomized trees ensembles. Experiments realized in this paper showed that it seems to have an empirical relationship between KTA and the ensemble performance. This connection highlights the role of classifiers diversity as well as individual strength. We also showed that increasing the amount of randomization has the effect to better regularize the ensemble. We should however be careful when analyzing the relation between level of randomization and strength-diversity tradeoff. Indeed, increasing the amount of randomization does not necessarily imply increasing the diversity or decreasing the strength and vice-versa. Open questions and interesting future aspects are : 1) how one can find another ensemble strategy acting more intensively on KTA, 2) realizing more experiments to deeper test the relation between KTA and the performance of an ensemble, 3) if it is possible to generalize the KTA framework to other ensemble approaches such as Boosting algorithms.

We have also suggested two possible improvements of classical randomized ensembles strategies. The first one consists in post-processing the ensemble with powerful linear separators. In our experiments, post-processing always led to better performance than a classical majority voting. Another alternative consists in taking into account the set of all possible basis functions induced by any decision tree having a specified number of leaves. This is possible thanks to the use of an appropriate kernel embedding those basis functions. Experimentally, both "improvements" gives very similar results. Possible future works here are : 1) should we benefit of using another post-processing strategy which uses a penalty on the L_1 norm of regressors. The main advantage could lie in the sparsity property of such regularizers leading in more interpretable results. 2) Breiman [4] showed that in a space of d dimensions, there exists a set of weights $w_1, ..., w_\infty$ which applied to all possible decision trees having $d + 1$ terminal nodes converges to the Bayes rate. Interestingly, the approach presented in section 4, as well as the one of [23] enters completely in this framework and gives perhaps a possible way of finding such weights. A critical step here would be to study the consistency of SVMs when used with Laplacian kernels.

References

1. Biau, G., Devroye, L., Lugosi, G.: Consistency of random forests and other averaging classifiers. Journal of Machine Learning Research 9, 2015–2033 (2008)
2. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
3. Breiman, L.: Bias, variance and arcing classifiers (1996), `http://www.sasenterpriseminer.com/documents/arcing.pdf`

4. Breiman, L.: Some infinity theory for predictor ensembles (2000), http://www.stat.berkeley.edu
5. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
6. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks (1984)
7. Buntine, W., Niblett, T.: A further comparison of splitting rules for decision tree induction. Machine Learning 8, 75–85 (1992)
8. Chang, C.-C., Lin, C.-J.: Libsvm: A library for support vector machines (2001), Software available at, http://www.csie.ntu.edu.tw/cjlin/libsvm
9. Cristianini, N., Kandola, J., Elisseeff, A., Shawe-Taylor, J.: On kernel-target alignment. In: Holmes, D., Jain, L. (eds.) Innovations in Machine Learning: Theory and Application, pp. 205–255 (2006)
10. Cutler, A., Zhao, G.: Pert - perfect random tree ensembles. Computer Science and Statistics (2001)
11. Demiriz, A., Bennett, K., Shawe-Taylor, J.: Linear programming boosting via column generation. Machine Learning 46, 225–254 (2002)
12. Dietterich, T.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40, 139–157 (2000)
13. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: ICML, pp. 148–156 (1996)
14. Freund, Y., Schapire, R.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence 14, 771–780 (1999)
15. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The Annals of Statistics 38, 95–118 (2000)
16. Friedman, J., Popescu, B.: Predictive learning via rule ensembles. The Annals of Applied Statistics 2, 916–954 (2008)
17. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 63, 3–42 (2006)
18. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result analysis of the nips 2003 feature selection challenge. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 545–552. MIT Press, Cambridge (2005)
19. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: Data Mining, Inference and Prediction. Springer, Heidelberg (2009)
20. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A practical guide to support vector classification (2003), http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
21. Kleinberg, E.: On the algorithmic implementation of stochastic discrimination. IEEE Trans. Pattern Anal. Mach. Intell. 22, 473–490 (2000)
22. Kuncheva, L., Whitaker, C.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51, 181–207 (2003)
23. Lin, H.-T., Li, L.: Support vector machinery for infinite ensemble learning. Journal of Machine Learning Research 9, 941–973 (2008)
24. Liu, F.-T., Ting, K.-M., Yu, Y., Zhou, Z.-H.: Spectrum of Variable-Random Trees. Journal of Artifical Intelligence Research 32, 355–384 (2008)
25. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)

26. Ratsch, G., Onoda, T., Muller, K.-R.: Soft margins for adaboost. Machine Learning 42, 287–320 (2001)
27. Rosset, S., Zhu, J., Hastie, T.: Boosting as a regularized path to a maximum margin classifier. Journal of Machine Learning Research 5, 941–973 (2004)
28. Scholkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2001)
29. Utgoff, P., Clouse, J.: A kolmogorov-smirnov metric for decision tree induction (1996), http://www.ics.uci.edu/~mlearn/MLRepository.html
30. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
31. Wolpert, D.: Stacked Generalization. Neural Networks 5, 241–259 (1992)