Received February 6, 2019, accepted February 17, 2019, date of publication February 21, 2019, date of current version March 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900755

Random Forests for Regression as a Weighted Sum of *k*-Potential Nearest Neighbors

PABLO FERNÁNDEZ-GONZÁLEZ, CONCHA BIELZA, AND PEDRO LARRAÑAGA
Technical University of Madrid, 28660 Madrid, Spain

Corresponding author: Pablo Fernández-González (pablo.fernandezgonz@fi.upm.es)

This work was supported by the Spanish Ministry of Economy and Competitiveness through the Cajal Blue Brain Project (the Spanish partner of the Blue Brain Initiative from EPFL) under C080020-09, and in part by the TIN2016-79684-P Project.

ABSTRACT In this paper, we tackle the problem of random forests for regression expressed as weighted sums of datapoints. We study the theoretical behavior of k-potential nearest neighbors (k-PNNs) under bagging and obtain an upper bound on the weights of a datapoint for random forests with any type of splitting criterion, provided that we use unpruned trees that stop growing only when there are k or less datapoints at their leaves. Moreover, we use the previous bound together with the concept of b-terms (i.e., bootstrap terms) introduced in this paper, to derive the explicit expression of weights for datapoints in a random (k-PNNs) selection setting, a datapoint selection strategy that we also introduce and to build a framework to derive other bagged estimators using a similar procedure. Finally, we derive from our framework the explicit expression of weights of a regression estimate equivalent to a random forest regression estimate with the random splitting criterion and demonstrate its equivalence both theoretically and practically.

INDEX TERMS Random forests, regression, bagging, bootstrap, nearest neighbors, k-potential nearest neighbors.

I. INTRODUCTION

Random forests is a powerful machine learning ensemble method that has achieved state-of-the-art performance in classification and regression tasks. It is computationally fast, produces high accuracy results, has a low parameter count for an ensemble and can handle small sample sizes even with a high number of features. As such, it has earned a wide interest in the research community that spawned a significant amount of papers [5]. It operates by training multiple decision or regression trees each on bootstrapped samples of the data and combining their predictions most typically by voting (classification) or averaging (regression). In the process of building each tree, a randomly selected subset of the total number of features is used at each time the data is split to search for the locally optimal splitting point (also referred to as the cutoff in continuous variables). To determine the optimal splitting point, a splitting criterion is required. In the random forest literature, the two most used splitting criteria for classification are the Gini index and the entropy index. For regression, it is the predicted squared error.

The associate editor coordinating the review of this manuscript and approving it for publication was Md Asaduzzaman.

In this paper, we focus on random forests (RFs) [7] for regression. Initially, we have a training dataset $\mathcal{D}_n = \{(\mathbf{x}_1, y_1), \ldots, \mathbf{x}_n, y_n)\}$ of n i.i.d. samples from a $(d+1)$-dimensional random vector $(\mathbf{X}, Y)$ taking values in $\mathbb{R}^d \times \mathbb{R}$. Our goal is to estimate the regression function $f(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ for any $\mathbf{x} \in \mathbb{R}^d$ using $\mathcal{D}_n$. In doing so, we attempt to minimize the mean squared error $MSE = \mathbb{E}[\hat{f}(\mathbf{x}) - f(\mathbf{x})]^2$, where $\hat{f}(\mathbf{x})$ is the regression function estimate of $f(\mathbf{x})$. In this context, we refer to the random forest regression estimate as $\hat{f}_{RF}(\mathbf{x})$.

While RF desirability has been displayed at a practical level, sound mathematical understanding of the method is still a lacking subject. For the case of RFs with regression trees, the problem stems from the intricate relationships between bagging (Boostrap + AGGregatING) [6], [8] and the splitting criteria together, which renders individual regression trees and conventional statistical analyses insufficient for describing the ensemble. In the direction of mathematical understanding of the model, some early works include [7], that offered a widely known upper bound on the generalization error based on the strength (individual classifier's performance) and correlation (similarity of response of the individual classifiers for given inputs) of the members of the ensemble. More recently, [16] showed that when regression trees are grown without pruning and with a fixed parameter k that regulates the tree growth by stopping whenever there are k or fewer examples in a node, the regression function estimate given by a RF algorithm can be viewed as a weighted sum of datapoints:

$$\hat{f}_{RF_1}(\mathbf{x}_0) = \sum_{i=1}^{n} w_i(\mathbf{x}_0) y_i, \tag{1}$$

where y_i is the response value associated with datapoint $\mathbf{x}_i$, $w_i(\mathbf{x}_0)$ is a weight that scales the contribution of y_i to the final prediction and $\mathbf{x}_0$ is the target datapoint to be predicted. Additionally, an equivalence relationship between RFs and a special type of nearest neighbors ([12], [13]) called k-potential nearest neighbors (k-PNNs) was found out. It was shown that if we omit bootstrapping, the regression function estimate given by RFs can be expressed as

$$\hat{f}_{RF_2}(\mathbf{x}_0) = \sum_{\mathbf{x}_i \in P_k(\mathbf{x}_0|\mathcal{D}_n)} w_i(\mathbf{x}_0) y_i, \tag{2}$$

where $P_k(\mathbf{x}_0|\mathcal{D}_n)$ is a set containing the k-PNN datapoints of $\mathbf{x}_0$ in $\mathcal{D}_n$. In this setting, different splitting criteria determine different $w_i(\mathbf{x}_0)$ values for each datapoint, and different $w_i(\mathbf{x}_0)$ functions. This work established the foundations for a path towards a sound understanding of the model. Another work, [4], extended [16] and achieved consistency results on a regression estimate that uses the 1-PNN, as well as further understanding of the bagging technique when applied to the well-known nearest neighbors algorithm. Both Equation (2) from [16] and the bagging and 1-PNN analyses of [4] have been sources of inspiration for this work.

While Equation (1) shows that the regression function estimate of RFs can be expressed in terms of the weights, an explicit expression for the weights is still unknown for any splitting criterion. Moreover, RFs equipped with non adaptive splitting criteria (i.e., that do not depend on the Y values) such as random splitting, while being studied and widely regarded as a simpler case of RF [10], [14], still lack an explicit expression of these weights. In this direction, while literature concerning bagged regression estimates as weighted sums of datapoints is relatively abundant for some selected regression estimates [9], [17], [19], the general consensus is that the bagged form of a regression estimate cannot be computed analytically for most cases and Monte Carlo simulation must be used instead [18].

An explicit expression for the weights for a given splitting criterion would propose an alternative to the need of training stage for a RF model building algorithm, shifting all computational burden to the estimation of regression values of new examples and completely eliminating trees (effectively overcoming the Monte Carlo computational approach). Additionally, an equivalence between RFs and other more understood models could provide additional insights that could help us understand the unknown theoretical underpinnings of RFs. To the best of our knowledge, these weights are more directly discussed in [4], characterized as "nonnegative Borel measurable functions of" $\mathbf{x}_0$ that sum to 1, but no method for the explicit, analytical expression of these weights can be found in the literature.

Together with this, [4] and [16] analyses left some open questions: In [16], the k-PNN equivalence was discovered, but bootstrapping was discarded as a simplification on the RF models in order to make the analysis affordable. Thus, the question of the k-PNNs relationship with RFs equipped with bootstrapping remained unsolved. In [4], no analysis was performed on the bagged 1-PNN regression estimate and results for $k > 1$ were not considered, both remaining as open problems.

In this paper we propose a framework for the analysis and explicit calculation of the weights corresponding to general RFs, using bootstrapping and different splitting criteria. Effectively answering all previously exposed concerns.

In Section 2 we review in detail the concept of k-PNNs and outline some of its most interesting properties.

In Section 3 we solve the problem of determining the influence of bootstrapping on bagged estimators (including RF) in terms of weights using k-PNNs. We call these weights bootstrapped weights and obtain results for $k = 1$.

In Section 4 we analyze the addition of splitting criteria to our previous developments. We first derive an upper bound on the final weights for any type of splitting criterion, and follow it by the proposal of a regression estimate called Random k-PNN Selection. We then extend the results of Section 3 for arbitrary k by means of a proposed notation on the bootstrap variations, denominated b-terms. Additionally, we use this notation to derive explicit weights for the Random k-PNN Selection regression estimate. Finally, we introduce a framework to derive bagged estimators for the general case of a splitting criterion and with it, we obtain a regression estimate that corresponds with a RF that uses random splitting criterion and stops at k datapoints in its leaves.

In Section 5 we validate the predictive behavior of both the random k-PNN selection regression estimate and the obtained RF-equivalent regression estimate with some practical experiments, to illustrate the results of our work.

Finally, in Section 6 we summarize our work and present our conclusions.

The Appendix contains the proofs of our results.

II. *k*-POTENTIAL NEAREST NEIGHBORS

Intuitively, a datapoint $\mathbf{x}_i$ in the feature space $\mathbb{R}^d$ is considered a k-potential nearest neighbor (k-PNN) [16] of another, $\mathbf{x}_0$, if the hyperrectangle defined by $\mathbf{x}_i$ and $\mathbf{x}_0$ as opposing vertices ($\mathbf{x}_0$ not included) contains k or less datapoints in the feature space. Formally:

Definition 1: Let $R(\mathbf{x}_0, \mathbf{x}_i)$ denote the set of datapoints contained in the hyperrectangle defined by $\mathbf{x}_0$ and $\mathbf{x}_i$ as opposing vertices ($\mathbf{x}_0$ not included) in the feature space in $\mathcal{D}_n$. Then $\mathbf{x}_i$ is a k-PNN of $\mathbf{x}_0$ in $\mathcal{D}_n$ if and only if $|R(\mathbf{x}_0, \mathbf{x}_i)| \leq k$ ($\mathbf{x}_i \in P_k(\mathbf{x}_0|\mathcal{D}_n)$).

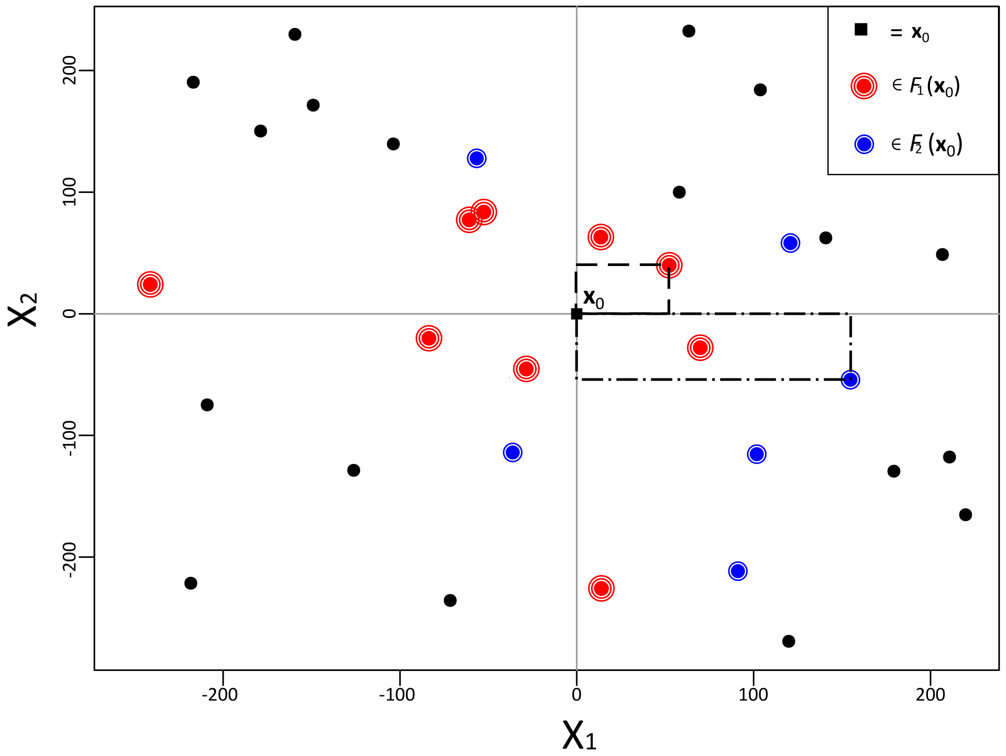


FIGURE 1. $\mathbf{X} = (X_1, X_2)$ **feature space plot where we outline the datapoints in** $F_1(\mathbf{x}_0)$ **(red) and in** $F_2(\mathbf{x}_0)$ **(blue), with** $\mathbf{x}_0 = (0, 0)$**. The number of concentric circles around a datapoint represents the number of times the datapoint is selected as a** k**-PNN in the plot for** $k = 1, 2$**. Notice how the dashed rectangular area does not contain any other datapoint for the datapoints in** $F_1(\mathbf{x}_0)$ **and the dashed dot rectangular area contains just one for the datapoints in** $F_2(\mathbf{x}_0)$**.**

$|.|$ denotes the cardinality of a set and $k \in \mathbb{N}$. Additionally, we now define $F_k(\mathbf{x}_0)$ as the set of datapoints of $\mathcal{D}_n$ that have exactly k datapoints contained in the hyperrectangle that goes from each of them to $\mathbf{x}_0$. That is, $F_k(\mathbf{x}_0) = \{\mathbf{x}_i \in \mathcal{D}_n$ such that $|R(\mathbf{x}_0, \mathbf{x}_i)| = k\}$.

Figure 1 shows an example of the k-PNN points of a point $\mathbf{x}_0$ for two different values of k ($k = 1, 2$). Note that $|P_k(\mathbf{x}_0)|$ (we use $P_k(\mathbf{x}_0)$ instead of $P_k(\mathbf{x}_0|\mathcal{D}_n)$ when the context is clear) can be clearly more than k. For a more precise study of the cardinality of the k-PNNs, the number of the 1-PNNs to be expected for uniform and arbitrary finitely bounded densities in $\mathbb{R}^d$ have been studied in [4] and [16], respectively.

k-PNNs have a number of interesting properties:

k-PNNs correspond to a special case of nearest neighbors where the distance value is defined as the number of datapoints selected by all monotone distances [16]. A monotone distance satisfies the following property: Given datapoints $\mathbf{x}_a$ and $\mathbf{x}_b$ and the hyperrectangle defined by both as opposing vertices, any point $\mathbf{x}_c$ inside the hyperrectangle would be considered "closer" to $\mathbf{x}_a$ or to $\mathbf{x}_b$ than $\mathbf{x}_a$ to $\mathbf{x}_b$, (for example, all p-norm $||.||_p$ distances are monotone distances).

With this we can define the PNN distance between datapoints $\mathbf{x}_0$ and $\mathbf{x}_i$ as a function in $\mathbb{N}$ that outputs the number of datapoints inside the hyperrectangle defined by both as opposing vertices.

The particular case 1-PNN has received special attention and is commonly referred to as the layered nearest neighbors in the literature. It was initially proposed as an example of scale invariant metric in [11]. Biau and Devroye [4] showed that the layered nearest neighbors are closely related to the notions of maximum [2] and dominance [1] in high dimensional spaces. A point $\mathbf{x}_a$ dominates another $\mathbf{x}_b$ if $x_{ai} \geq x_{bi}$ for all $i = 1, \ldots, d$ and a point is a maximum if no point dominates it. The relationship between k-PNNs and dominance is the following: If we consider each quadrant separately and apply absolute value to the coordinates, 1-PNNs or layered nearest neighbors are precisely the points that do not dominate any other point. For arbitrary k, while not explicitly mentioned in [4], the k-PNNs are the points that dominate k or fewer points.

Finally, k-PNNs exhibit a property that links them directly to RFs that grow non-pruned trees stopping at leaves with k or less datapoints. Regression tree cuts at splitting points define hyperrectangular partitions of the feature space, and the number of possible partitions in a RF that include a point $\mathbf{x}_0$ with k or fewer datapoints, is finite and determined by the distribution of the datapoints. As proven in [16] we have that, for a fixed dataset (i.e., without bootstrapping) the datapoints $\mathbf{x}_i$ that can be selected with $|R(\mathbf{x}_0, \mathbf{x}_i)| \leq k$, are the k-PNNs of $\mathbf{x}_0$ ($P_k(\mathbf{x}_0)$), that is, the voting points of a RF (as in Equation (2)).

III. BAGGING AND k-PNN

Biau et al. [3] analyzed the regression estimate resulting from bagging the 1-NN regression estimator. It was shown that the bagged 1-NN takes the form of a weighted NN estimator where each point contributes to the regression estimate of $\mathbf{x}_0$, $\hat{f}^*_{1-NN}(\mathbf{x}_0)$, according to

$$\hat{f}^*_{1-NN}(\mathbf{x}_0) = \sum_{i=1}^{n} v_i(\mathbf{x}_0) y_i, \tag{3}$$

where all $\mathbf{x}_i$ datapoints are here sorted by increasing distance to $\mathbf{x}_0$ in the feature space, the $*$ symbol denotes a bagged estimator and $v_i(\mathbf{x}_0)$ is the probability that the i-th NN of $\mathbf{x}_0$, $\mathbf{x}_i$ in $\mathcal{D}_n$, is the closest neighbor in a bootstrapped dataset. The set of v_i's is in this case a decreasing sequence given by the expression

$$v_i(\mathbf{x}_0) = \left(1 - \frac{i-1}{n}\right)^n - \left(1 - \frac{i}{n}\right)^n. \tag{4}$$

We will refer to the set $V_{NN} = \{v_1(\mathbf{x}_0), \ldots, v_i(\mathbf{x}_0), \ldots, v_n(\mathbf{x}_0)\}$ as the bootstrap weights for the NN regression estimate.

Our interest now lies in understanding how bootstrap weights behave in a similar setting but using the set of k-PNN points instead of the k-NNs. We start by understanding that similarly to the previous case, each point must be weighted by an additional $v_i(\mathbf{x}_0)$ factor where $v_i(\mathbf{x}_0)$ is the probability that $\mathbf{x}_i$ is a k-PNN of $\mathbf{x}_0$ in a bootstrapped dataset ($\mathbf{x}_i$'s are not sorted here). Our bootstrap weights $v_i(\mathbf{x}_0)$ would appear in the bagged version of

$$f_{k-SA}(\mathbf{x}_0) = \sum_{\mathbf{x}_i \in P_k(\mathbf{x}_0)} y_i, \tag{5}$$

that is, $f^*_{k-SA}(\mathbf{x}_0)$.

We will refer to Equation (5) as "select all" point selection strategy, hence the SA subindex. Notice that Equation (5) is not an estimator of $\mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$. The normalized version, $\hat{f}_{k-PNN}(\mathbf{x}_0) = \frac{1}{|P_k(\mathbf{x}_0)|} \sum_{\mathbf{x}_i \in P_k(\mathbf{x}_0)} y_i$ is a regression estimate and is studied in [4] for $k = 1$ as the layered NN estimate.

In order to calculate $\hat{f}^*_{k-SA}(\mathbf{x}_0)$, additional results and definitions are needed, with the final solution, for arbitrary k, provided in section 4. We now continue with the following lemma and the case $k = 1$:

Lemma 2: Let us define the set $Rm(\mathbf{x}_i) = R(\mathbf{x}_0, \mathbf{x}_i) \setminus \{\mathbf{x}_i\}$, where $\mathbf{x}_0$ and $\mathbf{x}_i$ are datapoints and $\mathbf{x}_0$ is our prediction target. Then $\mathbf{x}_i$ is a k-PNN of $\mathbf{x}_0$ for all bootstrap variations such that $|\mathcal{D}^*_j \cap Rm(\mathbf{x}_i)| \leq k - 1$ where $\mathcal{D}^*_j \in B(\mathcal{D}_n)$ and $B(\mathcal{D}_n) = \{\mathcal{D}^*_1, \mathcal{D}^*_2, \ldots \mathcal{D}^*_{n^n}\}$ is the set of all bootstrap variation selections of $\mathcal{D}_n$.

Proof: See Appendix.

Intuitively, Lemma 2 establishes for a datapoint $\mathbf{x}_i$ such that $|R(\mathbf{x}_0, \mathbf{x}_i)| = p$, $p \in \mathbb{N}$ to be a k-PNN with $p > k$, then $p - k$ points of $R(\mathbf{x}_0, \mathbf{x}_i)$ need not to appear in a considered bootstrap variation, or differently said, $\mathbf{x}_i$ is a k-PNN only in the fraction of the bootstrap variations that satisfy the requirement of Lemma 2 (for $p \leq k$, the only difference is that $\mathbf{x}_i$ is already a k-PNN in $\mathcal{D}_n$). Therefore, we only need to be concerned with the bootstrap variations that alter $P_k(\mathbf{x}_0)$.

A. THE 1-PNN CASE

For the remainder of this section and for purposes of simplicity, we will analyze the case of bootstrap weights for 1-PNN. Notice that in this case we need $Rm(\mathbf{x}_i) = \emptyset$ for $\mathbf{x}_i$ to be a 1-PNN.

For purposes of explanation, let us consider a dataset plot (Figure 2) and analyze both the cases of using 1-NN and 1-PNN point selection strategies of $\hat{f}_{1-NN}(\mathbf{x}_0)$ [3] and $f_{k-SA}(\mathbf{x}_0)$ (Equation (5)), respectively. Using 1-NN as our criterion and in a continuous feature space, we can arrange all datapoints in a ranking type hierarchy (from lowest to highest Euclidean distance to $\mathbf{x}_0$), where the point to be selected as 1-NN is always the highest ranked that appears in the bootstrapped variation. In other words, the i-th ranked point will be selected as the 1-NN in the bootstrap variations that do not include the first $i - 1$ ranked points.

For 1-PNNs, distances between points are discrete (PNN distance) and multiple point selections occur in the general case (that is, multiple 1-PNNs for a given $\mathbf{x}_0$ are expected). The result is a seemingly complex hierarchy where some points are linked to certain others by a "+1 PNN distance" relationship that determines the bootstrap requirements for a datapoint to be selected as a 1-PNN (Figure 2 and Figure 3).

We are now prepared for the following theorem:

Theorem 3: Let m be the minimum value of k for which all datapoints are m-PNN. Then f^*_{1-SA} can be written as

$$f^*_{1-SA}(\mathbf{x}_0) = \sum_{i=1}^{m} \left(\sum_{\mathbf{x}_j \in F_i(\mathbf{x}_0)} v_j(\mathbf{x}_0) y_j \right), \tag{6}$$

where the set of bootstrap weights V_{SA} are of the form:

$$v_j(\mathbf{x}_0) = \left(1 - \frac{|R(\mathbf{x}_0, \mathbf{x}_j)| - 1}{n}\right)^n - \left(1 - \frac{|R(\mathbf{x}_0, \mathbf{x}_j)|}{n}\right)^n$$

where $|R(\mathbf{x}_0, \mathbf{x}_j)| = i$.

Proof: See Appendix.

Notice that as expected from multiple point selections, we do not necessarily have $\sum_{i=1}^{n} v_i(\mathbf{x}_0) = 1$, and in most of the cases $\sum_{i=1}^{n} v_i(\mathbf{x}_0) > 1$. Generalization of this theorem for $k > 1$ will be provided in Section 4.

Our achievement here, described in terms of hierarchies for bootstrap variation requirements, is that it makes the bootstrap weights of 1-PNNs accessible for calculation as they are effectively expressed in Theorem 3 and by Lemma 2, in the same way as the resulting V_{NN} from bagging the 1-NN regression estimate. Its importance lies in that it describes the voting points (1-PNN) variations under bootstrapping, thus making it a useful tool for RF analysis.

IV. REGRESSION ESTIMATES AS A WEIGHTED SUM OF k-PNNS

We are now interested in obtaining the final weights, that is, the set of weights W_{RF}, that accounts for both

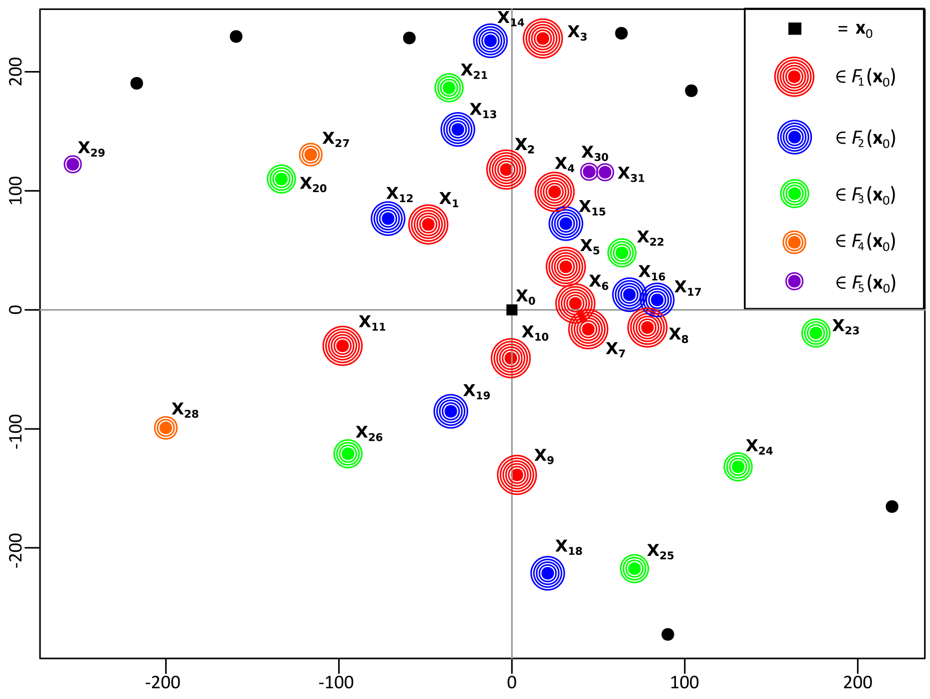


FIGURE 2. Feature space plot showing the outlining of $F_k(\mathbf{x}_0)$ of $\mathbf{x}_0 = (0, 0)$ for values of k from 1 to 5.

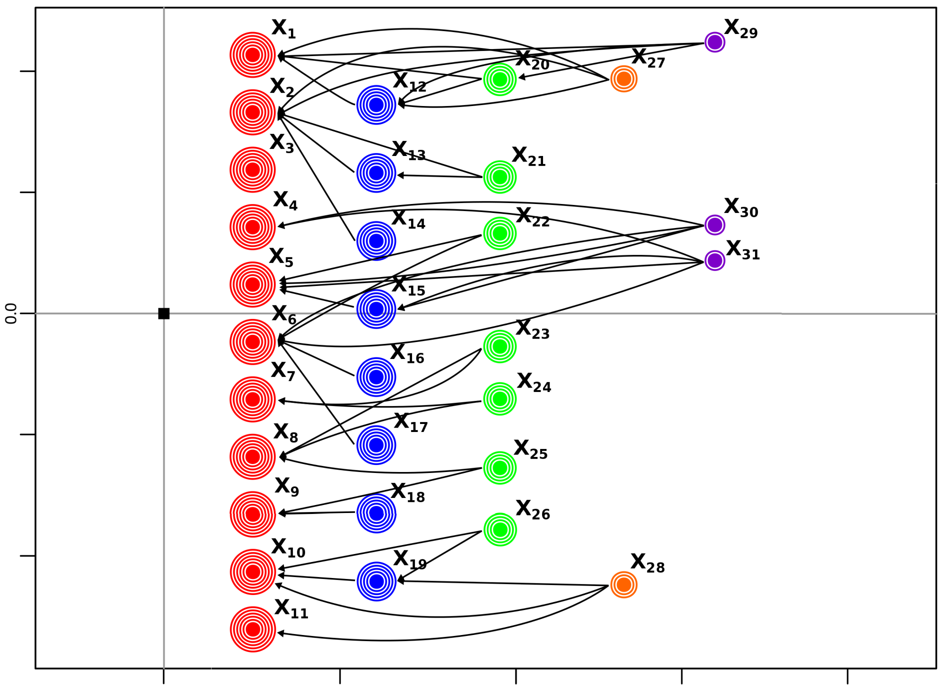


FIGURE 3. Graphical representation of the hierarchical precedence order for 1-PNN for the datapoints in Figure 2. In the plot, original points are arranged in a hierarchy that shows the precedence relationships for the 1-PNN. Similarly, we can see that the $k-1$ points connected by the arrows from another $\mathbf{x}_i$ are the points that need not to be included in a bootstrapped sample for $\mathbf{x}_i$ to be selected as a 1-PNN. For example, $\mathbf{x}_{29}$ will be selected as 1-PNN if and only if $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_{12}$ and $\mathbf{x}_{20}$ are not included in the bootstrapped sample.

bootstrapping and splitting criterion in a RF algorithm, where $w_i(\mathbf{x}_0) \in W_{RF}$ is the probability that $\mathbf{x}_i$ is selected in a RF algorithm.

In [16] it was shown that the splitting criteria can be viewed as weight redistributors for the obtained k-PNNs, corresponding to some particular solutions for the weights in

Equation (2). Also, for a fixed dataset, the splitting criterion can be interpreted as a selector of k points from $P_k(\mathbf{x}_0)$.

We add the following result to the previous considerations by understanding the relationship between bootstrap weights and the final weights:

Lemma 4: Let $\hat{f}_{RF}(\mathbf{x}_0)$ be a RF regression estimate that uses bootstrapping and unpruned trees that stop at $k = 1$ datapoints in the leaves. Let $\mathbf{x}_i$ be a datapoint in $\mathcal{D}_n$ and $\mathbf{x}_0$ the datapoint to predict. Then

$$v_i(\mathbf{x}_0) = \left(1 - \frac{|R(\mathbf{x}_0, \mathbf{x}_i)| - 1}{n}\right)^n - \left(1 - \frac{|R(\mathbf{x}_0, \mathbf{x}_i)|}{n}\right)^n$$

is an upper bound of the final weight $w_i(\mathbf{x}_0)$ (that is, $v_i(\mathbf{x}_0) \geq w_i(\mathbf{x}_0)$).

Proof: See Appendix.

Lemma 4 holds independently of the chosen split. Thus, it also holds for all RFs growing unpruned trees stopping when there is $k = 1$ datapoint in the leaves. Intuitively, we can think of the splitting criterion as a second point selection strategy applied after the selection of the k-PNNs for a given $\mathbf{x}_0$, which causes only up to k k-PNNs to be selected. In the general case, splitting criteria can be viewed as some form of weight shrinking procedure of bootstrap weights V_{SA} (that is, the bootstrap weights for the $f^*_{k-SA}(\mathbf{x}_0)$ regression estimate), that bounds the resulting $w_i(\mathbf{x}_0)$ to $\sum_{i=1}^n w_i(\mathbf{x}_0) = 1$. That is, a normalized regression estimator.

A. ANALYSIS OF POINT SELECTION STRATEGIES USING WEIGHTED B-TERMS

Now let us consider a regression estimate that applies a random selection over the k-PNNs, that is, k random k-PNN points are uniformly selected among the existing k-PNNs for each bootstrapped sample. We have

$$\hat{f}_{RkS}(\mathbf{x}_0) = \frac{1}{k} \sum_{\mathbf{x}_i \in P_k(\mathbf{x}_0)} \mathbb{1}_{[\mathbf{x}_i \in D(P_k(\mathbf{x}_0), k)]} y_i, \tag{7}$$

where $D(P_k(\mathbf{x}_0), k)$ is a set containing k uniform draws without replacement of datapoints from $P_k(\mathbf{x}_0)$. We call this criterion, random k-PNN selection (hence the *RkS* subindex on the regression estimate).

In a setting with a fixed dataset, the probability of selection for each k-PNN is equal to $\frac{1}{|P_k(\mathbf{x}_0)|}$. However, when considering bootstraping, different bootstrap variations have different number of k-PNNs. Classical analysis suggests we write the final weight of a point $\mathbf{x}_j$ under bootstrapping as

$$w_j(\mathbf{x}_0) = \frac{1}{n^n} \sum_{i=1}^{n^n} \mathbb{1}_{[\mathbf{x}_j \in P_k(\mathbf{x}_0|\mathcal{D}_i^*)]} \frac{1}{|P_k(\mathbf{x}_0|\mathcal{D}_i^*)|} \tag{8}$$

A first look at Equation (8) can be regarded as disappointing from a computational perspective, since it seems we are burdened with the need to calculate each individual $|P_k(\mathbf{x}_0|\mathcal{D}_i^*)|$ value.

Here we present an analysis framework that exploits the hierarchy of the PNNs and combinatorics regarding the bootstrap variations to arrive at a better calculation scenario. We introduce now the concept of b-term (bootstrap term).

Definition 5: A b-term $b_i = \mathbf{x}_a \ldots \mathbf{x}_b \ldots \neg\mathbf{x}_c \ldots \neg\mathbf{x}_d$ written as a list of datapoints of $\mathcal{D}_n$, denotes the proportion of bootstrap variation selections of $\mathcal{D}_n$ that include the datapoints $\mathbf{x}_a, \ldots, \mathbf{x}_b$ and do not include ($\neg$) datapoints $\mathbf{x}_c, \ldots, \mathbf{x}_d$.

Also, we define $S(b_i)$ as a function that outputs the numerical value associated with a b-term b_i. To further understand b-terms and function $S(.)$, we present here some of their properties (proofs of these properties are not included for the sake of brevity):

1) *Commutativity:* Writing order is commutative. That is, $b_i = \mathbf{x}_a \ldots \mathbf{x}_b \neg\mathbf{x}_c \ldots \neg\mathbf{x}_d = \mathbf{x}_a \ldots \neg\mathbf{x}_c \ldots \neg\mathbf{x}_d \ldots \mathbf{x}_b$.
2) *Reduction by contradiction:* For a b-term of the form $b_i = \mathbf{x}_a \ldots \mathbf{x}_b \neg\mathbf{x}_a \ldots \neg\mathbf{x}_d$ we have $S(b_i) = 0$. This can be interpreted as "no bootstrap variation selection can include and not include a point" ($\mathbf{x}_a$ in the example).
3) *Reduction by default:* For an "empty" b-term b_i we have $S(b_i) = 1$. That is, without restrictions (empty b-term) all bootstrap variations are included.
4) *Equivalence class:* Let us define $E_{[l_p,l_m]}$ as the set of all possible b-terms in $\mathcal{D}_n$ that have $l_p \in \mathbb{N}$ included datapoints restrictions and $l_m \in \mathbb{N}$ non-included datapoints restrictions. Then for all $b_i, b_j \in E_{[l_p,l_m]}$ we have $S(b_i) = S(b_j)$.
5) *Sum:* We define the sum of two b-terms b_i, b_j as $b_i + b_j$ and $S(b_i + b_j) = S(b_i) + S(b_j)$.
6) *Subtraction:* Similarly, we define the subtraction of two b-terms b_i, b_j as $b_i - b_j$ and $S(b_i - b_j) = S(b_i) - S(b_j)$.
7) *Concatenation:* We define the concatenation of b-terms b_i, b_j as $b_t = b_i b_j$. That is, another b-term containing all b_i and b_j datapoint restrictions.
8) *Concatenation of the sum:* $(b_i + b_j)(b_a + b_b) = b_i b_a + b_i b_b + b_j b_a + b_j b_b$. That is, the concatenation of the sum works in the fashion of a classical product operation.
9) *Reduction by sum:* For the sum of b-terms, restrictions of different type over the same datapoint can be canceled. That is, $\mathbf{x}_a \mathbf{x}_d b_i + \neg\mathbf{x}_a \mathbf{x}_d b_i = \mathbf{x}_d b_i$; since $(\mathbf{x}_a + \neg\mathbf{x}_a)$ covers all possible cases for datapoint $\mathbf{x}_a$.
10) *Reduction by subtraction:* Similar rules apply for defining and using the subtraction of b-terms. That is, $\mathbf{x}_d b_i - \neg\mathbf{x}_a \mathbf{x}_d b_i = \mathbf{x}_a \mathbf{x}_d b_i$.
11) *Reduction by redundancy:* Redundancy is canceled in b-terms. That is, $b_j = \mathbf{x}_a \mathbf{x}_a b_i = \mathbf{x}_a b_i$.
12) *Constant extraction:* Constants multiplying b-terms can be computed outside the $S(.)$ function. That is, $S(A b_i) = A S(b_i), A \in \mathbb{Z}$.

We can now use b-terms to write the bootstrap weights of Equation (4) (bagged 1-NN with datapoints sorted by increasing Euclidean distance) as

$$v_i(\mathbf{x}_0) = S(\neg\mathbf{x}_1 \neg\mathbf{x}_2 \ldots \neg\mathbf{x}_{i-1} \mathbf{x}_i),$$

and accounting for the decomposability showed in property 9, and property 6, we can rewrite

$$v_i(\mathbf{x}_0) = S(\neg\mathbf{x}_1 \ldots \neg\mathbf{x}_{i-1}) - S(\neg\mathbf{x}_1 \ldots \neg\mathbf{x}_{i-1} \neg\mathbf{x}_i),$$

The following lemma can now be introduced:

Lemma 6: The numerical value of a b-term $b_i = \overbrace{\mathbf{x}_a \ldots \mathbf{x}_b}^{l_p} \overbrace{\neg\mathbf{x}_c \ldots \neg\mathbf{x}_d}^{l_m} \in E_{[l_p,l_m]}$ *can be calculated as*

$$S(b_i) = \sum_{i=0}^{l_p} \binom{l_p}{i} (-1)^i \left(1 - \frac{i + l_m}{n}\right)^n \tag{9}$$

Proof: See Appendix.

It turns out that this notation allows us to express the final weights in a more accessible way than the direct computation of Equation (8). To illustrate this, let us break down Equation (8) into its pieces: We can see that $v_j(\mathbf{x}_0) = \frac{1}{n^n} \sum_{i=1}^{n^n} \mathbb{1}_{[\mathbf{x}_j \in P_k(\mathbf{x}_0|\mathcal{D}_i^*)]}$ (as these are all the cases where $\mathbf{x}_j$ is a k-PNN) and then see that $\frac{1}{|P_k(\mathbf{x}_0|\mathcal{D}_i^*)|}$ models the inclusion of the random k-PNN selection for each bootstrapping case. However, by Lemma 2, it is clear that this expression computes many unnecessary cases (as it iterates over all possible bootstrap variations without regard for changes in $P_k(\mathbf{x}_0|\mathcal{D}_i^*)$). Since b-terms cover subsets of the total bootstrap variation cases (those who satisfy the b-term), it is possible to cover the set of bootstrap variations for which $\mathbf{x}_j$ is a k-PNN using b-terms or sums of b-terms. For example, let us consider in isolation datapoints $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_{12}$ of Figure 2, with $\mathbf{x}_0$ as our prediction target and $k = 1$. Clearly, $\mathbf{x}_1, \mathbf{x}_2 \in F_1(\mathbf{x}_0)$ and $\mathbf{x}_{12} \in F_2(\mathbf{x}_0)$ since $\mathbf{x}_1$ is in the way. Then, we can simply write the bootstrap weights of $\mathbf{x}_{12}$ using b-terms as: $v_{12}(\mathbf{x}_0) = \neg\mathbf{x}_1\mathbf{x}_{12}$. Now, we need to account for the inclusion of the random 1-PNN selection using the b-terms notation. It turns out that by the property of reduction by sum, it is possible to expand a b-term into the different cases where the random 1-PNN selection takes different values for selecting a given datapoint. Continuing with our example, it is possible to do $\neg\mathbf{x}_1\mathbf{x}_{12} = \neg\mathbf{x}_1\mathbf{x}_{12}\mathbf{x}_2 + \neg\mathbf{x}_1\mathbf{x}_{12}\neg\mathbf{x}_2$ which effectively accounts for the cases where $\mathbf{x}_2$ is present and absent. Then, the final weight of $\mathbf{x}_{12}$ can be expressed as: $w_{12}(\mathbf{x}_0) = \frac{1}{2}S(\neg\mathbf{x}_1\mathbf{x}_{12}\mathbf{x}_2) + S(\neg\mathbf{x}_1\mathbf{x}_{12}\neg\mathbf{x}_2)$. This shows how weighted sums of b-terms can be used to express the final weights W_{R1S}.

Formally, we define for $b_i \in E_{[l_p,l_m]}$:

$$P_{R1S}(\mathbf{x}_i, b_i) = L_{R1S}(\mathbf{x}_i, b_i)S(b_i) \tag{10}$$

where

$$L_{R1S}(\mathbf{x}_i, b_i) = \frac{1}{l_p}.$$

and for which following property is verified:

$$P_{R1S}(\mathbf{x}_i, b_i + b_j) = P_{R1S}(\mathbf{x}_i, b_i) + P_{R1S}(\mathbf{x}_i, b_j).$$

Using the function $P_{R1S}(\mathbf{x}_i, b_i)$, it is possible to pair each b-term in isolation or in a sum of b-terms with the weight obtained by $L_{R1S}(\mathbf{x}_i, b_i)$. Thus, final weights W_{R1S} of any datapoint can be expressed using this function as long as the necessary b-terms are known.

Notice that in here, it was possible to define the random 1-PNN selector as a function $L_{R1S}(.,.)$ requiring only the local information of the b-term to output its corresponding value.

The $S(b_i)$ function, on the other hand, accounted for the k-PNN hierarchy and the selectability and bootstrap variations all together. In this sense, b-terms can be looked at as bootstrap variations themselves and hence our target is to find all relevant bootstrap variations for the calculation of $w_i(\mathbf{x}_0)$.

B. RANDOM *k*-PNN SELECTION REGRESSION ESTIMATE

Here we solve the problem of calculating the expression of the bagged version of the regression estimate in Equation (7), that is $\hat{f}^*_{RkS}(\mathbf{x}_0)$, and the explicit form of its final weights W_{RkS}. Considering our work so far, all that remains open is to find regularized way to write the expression that we obtain by expanding the b-terms of $v_i(\mathbf{x}_0)$ using the reduction by sum property until all datapoints are considered. For this, we add to the previous work on b-terms the following definition:

Definition 7: We define the restricted concatenation operator

$$[\mathbf{x}_a\mathbf{x}_b \ldots \neg\mathbf{x}_c\neg\mathbf{x}_d, \ldots, \mathbf{x}_f\mathbf{x}_g \ldots \neg\mathbf{x}_h\neg\mathbf{x}_i] \\ (\mathbf{x}_j\mathbf{x}_k \ldots \neg\mathbf{x}_m\neg\mathbf{x}_n \ldots)$$

as a special type of concatenation operator which specifies in brackets $[., \ldots, .]$ *to which other b-terms the expression* $(\mathbf{x}_j\mathbf{x}_k \ldots \neg\mathbf{x}_m\neg\mathbf{x}_n \ldots)$ *is concatenated.*

For Definition 7, let us consider the example

$$(\mathbf{x}_1 + \neg\mathbf{x}_1)(\mathbf{x}_2 + \neg\mathbf{x}_2)([\mathbf{x}_1\neg\mathbf{x}_2, \neg\mathbf{x}_1\neg\mathbf{x}_2](\mathbf{x}_3 + \neg\mathbf{x}_3)).$$

This results in:

$$\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_1\neg\mathbf{x}_2(\mathbf{x}_3 + \neg\mathbf{x}_3) + \neg\mathbf{x}_1\mathbf{x}_2 + \neg\mathbf{x}_1\neg\mathbf{x}_2(\mathbf{x}_3 + \neg\mathbf{x}_3),$$

where only the b-terms $\mathbf{x}_1\neg\mathbf{x}_2$ and $\neg\mathbf{x}_1\neg\mathbf{x}_2$ are concatenated with $(\mathbf{x}_3 + \neg\mathbf{x}_3)$.

Then, the following theorem holds:

Theorem 8: Let us consider a datapoint $\mathbf{x}_0$ *as our prediction target. Weights* W_{R1S} *for the* $\hat{f}^*_{R1S}(\mathbf{x}_0)$ *regression estimate have the form*

$$w_i(\mathbf{x}_0) = P_{R1S}(\mathbf{x}_i, z_i(\mathbf{x}_0))$$

where

$$z_i(\mathbf{x}_0) = (\mathbf{x}_i)(Req_1(\mathbf{x}_i)) \prod_{\mathbf{x}_j \in Ind(\mathbf{x}_i)} [Req_1(\mathbf{x}_j)](\mathbf{x}_j + \neg\mathbf{x}_j),$$

$Req_1(\mathbf{x}_i) = \{\neg\mathbf{x}_a\neg\mathbf{x}_b \ldots \neg\mathbf{x}_s\}$ *with* $Rm(\mathbf{x}_i) = \{\mathbf{x}_a, \mathbf{x}_b \ldots, \mathbf{x}_s\}$ *and* $Ind(\mathbf{x}_i) = \mathcal{D}_n \setminus R(\mathbf{x}_0, \mathbf{x}_i)$ *as the complementary set of points of* $R(\mathbf{x}_0, \mathbf{x}_i)$ *w.r.t. the data* $\mathcal{D}_n$.

Proof: See Appendix (Proof of Theorem 10).

The expression of $z_i(\mathbf{x}_0)$ shows "a sum expansion of $v_i(\mathbf{x}_0)$, where each added datapoint is restricted to be concatenated to the b-terms that can be expressed as $b_t Req_1(.)$ (that is, the b-terms that contain their $Req_1(.)$ set)". This guarantees that we only consider the inclusion or non inclusion of the datapoint in the subset of cases where it is relevant for the final weight calculation, while ignored otherwise.

Our goal is now to generalize the previous results for arbitrary k. With the b-terms notation, this turned out to be a natural step forward. We start with the introduction of the following lemma, that generalizes Lemma 4 for arbitrary k

Lemma 9: Let $\hat{f}^*_{RF}(\mathbf{x}_0)$ be a RF regression estimate that uses bootstrapping, unpruned trees and stops at arbitrary $k \in \mathbb{N}$ datapoints in the leaves. Let $\mathbf{x}_0$ be our prediction target and $\mathbf{x}_i$ another datapoint. Then

$$v_i(\mathbf{x}_0) = S\left(\mathbf{x}_i\left(\sum_{c \in Req_k(\mathbf{x}_i)} r_{ic}\right)\right)$$

is an upper bound of $w_i(\mathbf{x}_0)$ (that is, $v_i(\mathbf{x}_0) \geq w_i(\mathbf{x}_0)$), where $Req_k(\mathbf{x}_i) = \{r_{i1}, r_{i2}, \ldots, r_{ih}\}$, $h \in \mathbb{N}$, is defined as the set of b-terms listing all possible bootstrap variations where a subset of the datapoints in $Rm(\mathbf{x}_i)$ allows for $\mathbf{x}_i$ to be selectable as a k-PNN.

Proof: See Appendix (Proof of Theorem 10).

For Theorem 8, we first notice that Equation (10) does not need to change to account for the b-terms weights in the $k > 1$ case, since for a b-term $b_i \in E_{[l_p,l_m]}$ the corresponding weight in a random k-PNN selection would be $L_{RkS}(\mathbf{x}_i, b_i) = \left(\frac{1}{k}\right)\left(\frac{k}{l_p}\right) = \frac{1}{l_p}$. We can then define

$$P_{RkS} = P_{R1S}$$

and write:

Theorem 10: Let us consider a datapoint $\mathbf{x}_0$ as our prediction target. Weights W_{RkS} for the $\hat{f}^*_{RkS}(\mathbf{x}_0)$ regression estimate with arbitrary k have the form

$$w_i(\mathbf{x}_0) = P_{RkS}(\mathbf{x}_i, z_i(\mathbf{x}_0))$$

where

$$z_i(\mathbf{x}_0) = (\mathbf{x}_i)\left(\sum_{c \in Req_k(\mathbf{x}_i)} r_{ic}\right) \prod_{\mathbf{x}_j \in Ind(\mathbf{x}_i)} [Req_k(\mathbf{x}_j)](\mathbf{x}_j + \neg\mathbf{x}_j)$$

Proof: See Appendix.

We finally have:

Theorem 11: Let us consider a datapoint $\mathbf{x}_0$ as our prediction target. The $\hat{f}^*_{RkS}(\mathbf{x}_0)$ regression estimate has the form

$$\hat{f}^*_{RkS}(\mathbf{x}_0) = \sum_{i=1}^{n} w_i(\mathbf{x}_0) y_i \qquad (11)$$

where $w_i(\mathbf{x}_0)$'s are regarded as in the form of Theorem 10.

Proof: See Appendix.

By proving Theorem 11 we have succeeded in our original objective of finding a more direct and accessible approach to compute the weights that in Equation (8). Also, with this theorem we have solved an open problem in [4], since for the random k-PNN selection, the case $k = 1$ corresponds to the final weights W of the bagged layered NN regression estimate detailed in that paper, that is,

$$\hat{f}^*_{R1S}(\mathbf{x}_0) = \hat{f}^*_{1-PNN}(\mathbf{x}_0).$$

For the general case (since $L_{RkS} = L_{R1S}$), we also have

$$\hat{f}^*_{RkS}(\mathbf{x}_0) = \hat{f}^*_{k-PNN}(\mathbf{x}_0).$$

Finally and as a completing remark, we can now express the generalized bootstrap weights for f^*_{k-SA} as:

Theorem 12: Let m be the minimum value of k for which all datapoints are m-PNN. The bagged version of f_{k-SA} is of the form

$$f^*_{k-SA}(\mathbf{x}_0) = \sum_{i=1}^{m}\left(\sum_{\mathbf{x}_j \in F_i(\mathbf{x}_0)} v_j(\mathbf{x}_0) y_j\right), \qquad (12)$$

where $v_j(\mathbf{x}_0)$'s are regarded as in the form of Lemma 9.

Proof: See Appendix (Proof of Theorem 10).

C. BAGGED ESTIMATORS FRAMEWORK

From Equation (10), it is not difficult to imagine that other regression estimates may adjust to this model with a different $P_{.}(\mathbf{x}_i, b_i)$ function. The $P_{.}(\mathbf{x}_i, b_i)$ function general form is, for an estimator $\hat{f}_{.}(\mathbf{x}_0)$ in the calculation of the weight of $\mathbf{x}_i$:

$$P_{.}(\mathbf{x}_i, b_i) = L_{.}(\mathbf{x}_i, b_i) S(b_i). \qquad (13)$$

We have seen how the weight calculation obtained in Theorem 10 accounts for all the bootstrap cases of interest within the k-PNNs, thus, the different ways in which we can specify the $L_{.}(\mathbf{x}_i, b_i)$ function correspond to the different regression estimates. As an example, let us define

$$L_{1-NN}(\mathbf{x}_i, b_i) = \begin{cases} 1 & \text{if } b_i = \neg\mathbf{x}_1\neg\mathbf{x}_2\ldots\neg\mathbf{x}_{i-1}\mathbf{x}_i b_j \\ 0 & \text{otherwise} \end{cases} \qquad (14)$$

where here, datapoints are sorted by increasing Euclidean distance, ($\mathbf{x}_1$ being the closest to $\mathbf{x}_0$ and $\mathbf{x}_n$ the furthest away) as the $L_{.}(.,.)$ function that produces a bagged NN estimate for $k = 1$.

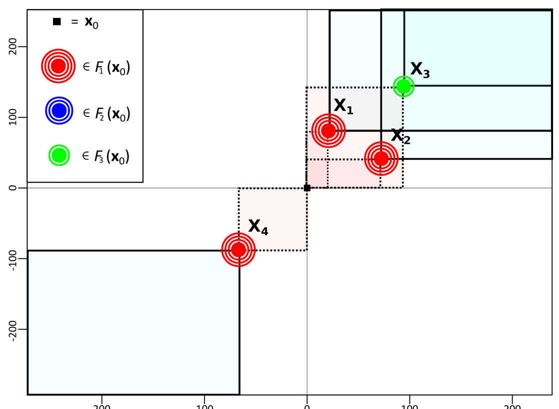


FIGURE 4. Feature space plot showing four datapoints and $F_k(\mathbf{x}_0)$ of $\mathbf{x}_0 = (0, 0)$ for values of k from 1 to 3.

We will show now how we can use, in this case, the $P_{1-NN}(\mathbf{x}_i, b_i)$ function, to deduce the weights of Equation (4). We use the simple dataset of Figure 4.

For the dataset of Figure 4 and $k = 1$ we can write the b-terms sum expansion per datapoint as follows:

$$\begin{aligned} z_1(\mathbf{x}_0) &= \mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4 + \mathbf{x}_1\neg\mathbf{x}_2\neg\mathbf{x}_4 \\ z_2(\mathbf{x}_0) &= \mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \neg\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4 + \neg\mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4 \\ z_3(\mathbf{x}_0) &= \neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4 + \neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_3\neg\mathbf{x}_4 \\ z_4(\mathbf{x}_0) &= \mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_4 + \neg\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 \\ &\quad + \neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4 + \neg\mathbf{x}_1\neg\mathbf{x}_2\neg\mathbf{x}_3\mathbf{x}_4 \end{aligned}$$

where here, the b-term sums were obtained using $z_i(\mathbf{x}_0)$ of Theorem 8. For this example, we have the Euclidean distance sorting of the datapoints as $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_3$ (thus, the $L_{1-NN}(\mathbf{x}_0)$ function will compute them as $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$, respectively). If we now apply $P_{1-NN}(\mathbf{x}_i, b_i)$ we obtain:

$$\begin{aligned} w_1(\mathbf{x}_0) &= S(\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4 \\ &\quad + \mathbf{x}_1\neg\mathbf{x}_2\neg\mathbf{x}_4) \\ w_2(\mathbf{x}_0 &= S(\neg\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \neg\mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4) \\ w_3(\mathbf{x}_0) &= S(\neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_3\neg\mathbf{x}_4) \\ w_4(\mathbf{x}_0) &= S(\neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4 + \neg\mathbf{x}_1\neg\mathbf{x}_2\neg\mathbf{x}_3\mathbf{x}_4) \end{aligned}$$

Now using the properties of the b-terms we can do:

$$\begin{aligned} w_1(\mathbf{x}_0) &= S(\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_4 + \mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4 \\ &\quad + \mathbf{x}_1\neg\mathbf{x}_2\neg\mathbf{x}_4) \\ &= S(\mathbf{x}_1\mathbf{x}_4 + \mathbf{x}_1\neg\mathbf{x}_4) \\ &= S(\mathbf{x}_1) \\ w_2(\mathbf{x}_0) &= S(\neg\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4 + \neg\mathbf{x}_1\mathbf{x}_2\neg\mathbf{x}_4) \\ &= S(\neg\mathbf{x}_1\mathbf{x}_2) \\ w_4(\mathbf{x}_0 &= S(\neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4 + \neg\mathbf{x}_1\neg\mathbf{x}_2\neg\mathbf{x}_3\mathbf{x}_4) \\ &= S(\neg\mathbf{x}_1\neg\mathbf{x}_2\mathbf{x}_4) \end{aligned}$$

which effectively yields the weights of the bagged 1-NN as were known in [3]. Using our framework and operations on the b-terms, we were able to deduce the form of the weights of the bagged estimator and reduce it to its known-form, requiring only one b-term per datapoint.

We argue here that any point selection strategy within the set of the k-PNNs can be adapted to this format, and by applying the b-terms properties to the result of the $P_{\cdot}(.,.)$ function, we can observe how the interplay between the b-terms and the $L.(.,.)$ function gives rise to the bagged version of many known regression estimates (for example, for all regression estimates using p-norm $||.||_p$ distances as point selectors, their bagged versions can be easily derived in a similar way than with the Euclidean distance in the 1-NN example). This includes the well-known predictive square error splitting criterion, considering that for an adaptive splitting criterion, decisions are made using an additional set of values Y.

D. RANDOM FOREST WITH RANDOM SPLIT REGRESSION ESTIMATE

Now, we are looking to attain one of the main goals of this work: inducing the L_{RF} function that would allow us to build a regression estimate that outputs similar predictions to those obtained by the RF algorithm with random split. Using this framework, we reformulated the problem of inducing the RF estimator by traditional means to that of finding an expression for $P(\mathbf{x}_i|b_i)$ for any b_i in $\mathcal{D}_n$.

We found a recursive procedure that allows us to calculate $L_{RF}(\mathbf{x}_i, b_i)$, with $b_i \in E_{[l_p,l_m]}$ and for arbitrary k as:

$$L_{RF}(\mathbf{x}_i, b_i) = \begin{cases} G_{RF}(\mathbf{x}_i, b_i) & \text{if } b_i \text{ includes } \mathbf{x}_i,\ b_i \in E_{[l_p,l_m]} \\ & \quad \text{and } l_p > k \\ \dfrac{1}{l_p} & \text{if } b_i \text{ includes } \mathbf{x}_i,\ b_i \in E_{[l_p,l_m]} \\ & \quad \text{and } l_p \leq k \\ 0 & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} G_{RF}&(\mathbf{x}_i, b_i) \\ &= \frac{1}{d}\sum_{l=1}^{d}\left(\sum_{k=1}^{l_p}\frac{I(k+1,l,b_i,\mathcal{D}_n) - I(k,l,b_i,\mathcal{D}_n)}{I_s(l,b_i,\mathcal{D}_n)}\right. \\ &\qquad \left.\times L_{RF}(\mathbf{x}_i, C(\mathbf{x}_i,k,l,b_i,\mathcal{D}_n))\right), \end{aligned}$$

$I(k, l, b_i, \mathcal{D}_n)$ outputs the value in the l-th feature/column of the k-th datapoint in a sorted sample (from lowest to highest values) of the datapoints listed in $\mathbf{x}_0 b_i$ that appear in $\mathcal{D}_n \cup \mathbf{x}_0$. $I_s(l, b_i, \mathcal{D}_n)$ outputs the range of values of the l-th feature/column for the datapoints listed in $\mathbf{x}_0 b_i$ that appear in $\mathcal{D}_n \cup \mathbf{x}_0$. $C(\mathbf{x}_i, k, l, b_i, \mathcal{D}_n)$ outputs a b-term b_s that contains a subset of the listed datapoints of b_i. b-term b_s is defined as follows: Let us consider the sorted sample of $\mathcal{D}_n$ datapoints listed in b_i by their values in the l-th feature/column. We can then divide b_i into two b-terms b_{s1} and b_{s2} by splitting the sorted sample at the k-th position. Then we can define b_{s1} to be the b-term that lists the datapoints that in the sorted sample appeared before the k-th position, and b_{s2} containing the rest so that $b_{s1}b_{s2} = b_i$. Finally, we define b_s as $b_s = b_{s1}$ if the interval covered by the l-th feature values of the datapoints listed in b_{s1} includes the l-th feature value of $\mathbf{x}_0$. If it doesn't, we define it as $b_s = b_{s2}$.

Intuitively, $L_{RF}(\mathbf{x}_i, b_i)$, accounts for all possible cases that the classical RF algorithm with random split may produce. For $G_{RF}(\mathbf{x}_i, b_i)$, modeling the random subspace method implies d possible choices of coordinate. Each choice weighted by $\frac{1}{d}$ (Notice that in this type of RF, this is always the case regardless of how we tune this parameter). Then, for $b_i \in E_{[l_p,l_m]}$, l_p splits are possible ($l_p - 1$ provided by the l_p datapoints listed in b_i and 1 provided by $\mathbf{x}_0$) per coordinate. Each split is weighted by its probability of occurrence on the selected coordinate. After choosing the split, two mutually exclusive subsets of datapoints are created. Then, the one not containing $\mathbf{x}_0$ is discarded, while the other is selected. Repeating this process recursively for the selected subset of datapoints as a new b_i produces $G_{RF}(\mathbf{x}_i, b_i)$. The second case of $L_{RF}(\mathbf{x}_i, b_i)$ accounts for the stopping criteria, which can be plugged directly, and the third case accounts for the b-terms that do not contribute to the final weight of $\mathbf{x}_i$.

With this, we are ready to present the following theorem:

Theorem 13: Let us consider a datapoint $\mathbf{x}_0$ *as our prediction target. The* $\hat{f}_{RF}(\mathbf{x}_0)$ *regression estimate has the form*

$$\hat{f}_{RF}(\mathbf{x}_0) = \sum_{i=1}^{n} w_i(\mathbf{x}_0) y_i \tag{15}$$

where $w_i(\mathbf{x}_0)$*'s have the form*

$$w_i(\mathbf{x}_0) = P_{RF}(\mathbf{x}_i, z_i(\mathbf{x}_0))$$

with

$$P_{RF}(\mathbf{x}_i, z_i(\mathbf{x}_0)) = L_{RF}(\mathbf{x}_i, z_i(\mathbf{x}_0)) S(z_i(\mathbf{x}_0))$$

and $z_i(\mathbf{x}_0)$ *is regarded as in the form of Theorem 10.*

Proof of this result is considered trivial after the proofs of Theorem 10 and Theorem 11.

This regression estimate, as we will show in Section 5, offers similar predictions to those of a RF in all cases and problems.

V. TOWARDS PRACTICAL IMPLEMENTATION AND RF EQUIVALENCE

After Section 4, we are provided with the means to rewrite bagged estimators that select points in the k-PNN set as sums of weighted b-terms. Additionally, we have succeeded in finding the explicit expression for these weights in the cases of a RF with random splitting criterion and our proposed $\hat{f}_{RkS}(\mathbf{x}_0)$ regression estimate. In this section, we seek to validate our findings at a practical level.

We started by implementing $\hat{f}_{RkS}(\mathbf{x}_0)$ with an algorithm that closely follows Theorem 11. In order to do so, we first noticed that function $S(.)$ as shown in Equation (9) displays a clear exponential growth with respect to l_p (in the binomial coefficient) and n (in the denominator inside the summation, as it is n^n) in computational complexity. In order to alleviate the complexity of both variables we use an easy workaround as any expression of the type $e(i) = \left(1 + \frac{i}{n}\right)^n$ satisfies $\lim_{n\to\infty} e(i) = e^i$. We can take advantage of this simply by approximating Equation (9) with

$$S(b_i) \approx \sum_{i=0}^{l_p} \binom{l_p}{i} (-1)^i e^{-i-l_m}, \tag{16}$$

offering an approximate result ($\sum_{i=1}^{n} w_i(\mathbf{x}_0) \leq 1$) that improves its accuracy the higher n becomes. We now do

$$\begin{aligned} S(b_i) &\approx \frac{1}{e^{l_m}} \sum_{i=0}^{l_p} \binom{l_p}{i} (-1)^i e^{-i} \\ &= \frac{1}{e^{l_m}} \sum_{i=0}^{l_p} \binom{l_p}{i} \left(\frac{-1}{e}\right)^i \\ &= \frac{1}{e^{l_m}} \left(\frac{e-1}{e}\right)^{l_p} \end{aligned}$$

which shows much clearly the relationship between bootstrapping and the b-terms. Also, this implies that final weights W are, when $n \to \infty$, a sum of weighted exponential functions.

As for the number of b-terms to be computed, we can see that the b-terms expansion grows exponentially in the worst case scenario: Examining $z_i(\mathbf{x}_0)$ in Theorem 10, the number of b-terms doubles at each iteration of the product in the subset of b-terms allowed by the $[Req_k(.)]$ set of the datapoint to be computed. This is still an improvement with respect to classical analysis and Equation (8) for the complete computation of weights (from $\mathcal{O}(n^n)$ to $\mathcal{O}(2^n)$) and in some cases, as we have seen in subsection 4.1, the choice of the $L_.(.,.)$ can reduce final complexity to a sub-exponential form. For our testing, however, we limited ourselves to small sample sizes and $k = 1$.

In setting up our experimental environment, we use two datasets from UCI data repository (`Bike` and `Concrete`, with 5 and 6 variables, respectively) and six datasets from the R package mlbench (`Ozone`, `Boston Housing`, `Friedman1` with sd = 1, `Friedman2` with sd =125 and `Friedman3` with sd = 0.1, with 12,14,11,5 and 5 variables, respectively). For each dataset, we normalized the response values Y subtracting the mean and dividing by the standard deviation in order to control the scale of MSE values. Additionally, we implemented a full random RF that for each tree at each node to split, selects a random feature and performs a random split between its maximum and minimum values until there is a single datapoint in the leaves ($k = 1$).

We computed the MSE statistics between the true values and predictions given by the models to assess their performance. We additionally computed other statistics for analysis purposes. The results of the experiments are shown in Table 1.

TABLE 1. Comparisons of the results of $\hat{f}^*_{R1S}$ and $\hat{f}_{RF}$ estimates for the selected datasets. The first two rows contain mean square error comparisons between real values and predicted values of both estimators, the range (±) is simply the standard deviation of the predictions, the third is the average of the average PNN distance that each testing point obtained w.r.t. the rest of the points in the dataset and the fourth is the ratio between the dimension of the dataset *d* and the sample size *n*.

	MSE - $\hat{f}^*_{R1S}$	MSE - $\hat{f}_{RF}$	$\overline{PNN}$	$\frac{d}{n}$
Bike	3.89±0.02	4.23±0.08	1.46	0.25
Concrete	1.18±0.00	2.56±0.60	0.00	0.30
Ozone	1.88±0.00	1.66±0.08	0.00	0.60
Boston	1.27±0.00	1.00±0.41	0.00	0.70
Friedman1	1.00±0.00	0.84±0.21	0.00	0.55
Friedman2	0.91±0.11	0.38±0.53	0.22	0.25
Friedman3	0.86± 0.14	0.41±0.54	0.16	0.25

In Table 1, most remarkable results come from analyzing the included statistics. In $\hat{f}^*_{R1S}$ evaluations, four of the seven datasets present no variability between predictions, independently of the test point. In those sets, the average PNN distance is 0, result that can only occur if all datapoints in all cases were 1-PNNs of each datapoint used in testing. This is expected, as $\hat{f}^*_{R1S}$ only distinguishes between predictions by

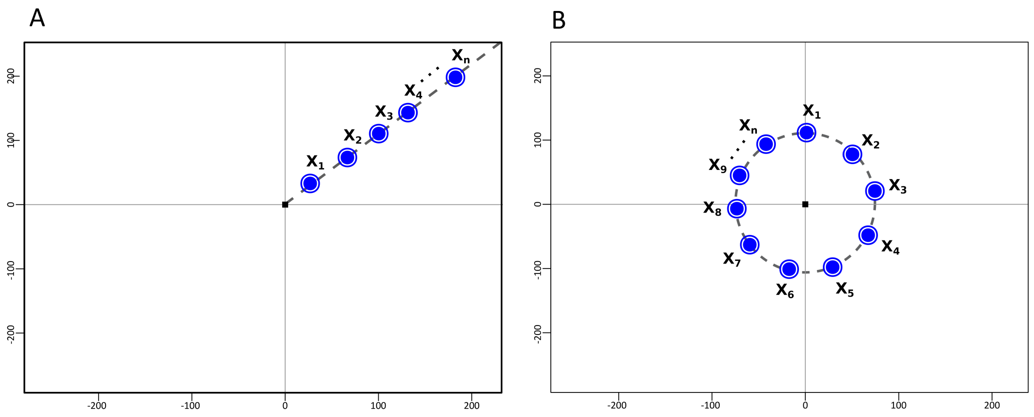


FIGURE 5. Two different cases showing maximum k-PNN distance arrangement and minimum k-PNN distance arrangement between a set of datapoints and the point to predict.

differences in the k-PNN distances. Additionally, this case seems to occur at the highest levels of the $\frac{d}{n}$ ratio.

It is not difficult to notice that increasing dimensionality (d) while maintaining sample size could reduce the average PNN distance in the general case. To see this clearly, lets imagine a set of datapoints distributed in the perimeter of a circle around the testing datapoint $\mathbf{x}_0$ in a 2-dimensional setting (Figure 5, case B). It can be verified that all datapoints in this setting exhibit a PNN distance of 0 w.r.t. $\mathbf{x}_0$. If we now project all datapoints onto a line by eliminating one of the dimensions, the result is that at most only the two immediate neighbors of $\mathbf{x}_0$ have PNN distance of 0.

To illustrate this practically, we removed all but two features on the previous datasets and repeated the computations, see Table 2.

TABLE 2. Comparisons of the results of $\hat{f}^*_{R1S}$ and $\hat{f}_{RF}$ estimates for the selected datasets and the reduced number of variables.

	MSE - $\hat{f}^*_{R1S}$	MSE - $\hat{f}_{RF}$	$\overline{PNN}$	$\frac{d}{n}$
Bike	4.40 ±0.09	4.58±0.14	8.14	0.15
Concrete	1.17±0.10	3.11±0.88	0.52	0.15
Ozone	1.82±0.05	1.86±0.13	2.12	0.15
Boston	1.22±0.39	1.33±0.51	3.77	0.15
Friedman1	0.72±0.28	0.71±0.49	2.06	0.15
Friedman2	0.73±0.35	0.82±0.59	2.11	0.15
Friedman3	0.70±0.42	0.73±0.70	1.67	0.15

For Table 2, MSE results lie in favor of $\hat{f}^*_{R1S}$ for most cases, decreasing with respect to Table 1, in opposition with the tendency shown by MSE results of $\hat{f}_{RF}$. Variability in $\hat{f}^*_{R1S}$, as expected, has increased yet remains substantially inferior to that of $\hat{f}_{RF}$. This seems to indicate that the reason for $\hat{f}^*_{R1S}$ to perform better is that the effects of bootstrapping have a higher influence on the outcome of the estimator when the average PNN distance increases.

Further understanding allows for the following characterization: consider point arrangement cases of Figure 5. Computing the b-terms of any weight with any splitting criteria that requires to select $k = 1$ datapoints will output the bootstrap weights (case limit of equality in Lemma 4) for case A, as for any given setting, the splitting criterion always selects 1 out of 1 datapoints. Thus, the influence of the employed $L_{.}(\mathbf{x}_i, b_i)S(b_i)$ function degenerates to its form of minimum variance, $L_{.}(\mathbf{x}_i, b_i) = 1$. For case B, the b-terms sum expansion is the same for every datapoint, and differences in weights can be attributed exclusively to the $L_{.}(\mathbf{x}_i, b_i)$ function variability.

Thus, we can safely argue that for any given problem, each prediction will fall between cases A and B, and thus the average PNN distance could be a good indicator of the overall contribution of each part of the estimator to the final outcome (and partially explain the results of Tables 1 and 2). Also this implies that the differences between bagged $||.||_p$ norm estimators (always case A) are governed exclusively by the differences in order in the ranking of datapoints. Our analysis seems to concur with literature [15] in the use of bootstrapping in high $\frac{d}{n}$ ratio (case B) problems as having a mild to poor effect on the overall quality of the predictions.

Finally, we repeated the experiments in Table 2 substituting L_{R1S} for L_{RF} and $k = 1$, to show that we can achieve a functional practical implementation of a RF using sums of weighted b-terms (Figure 6).

In Figure 6, it is clearly noticeable that virtually identical results were achieved for all datasets, where the minimal divergences in MSE can be safely attributed to a finite number of trees (less than all its possible variations) used to train $\hat{f}_{RF}$, together with the approximation of the bootstrap weights values shown in Equation (15).

With this we have shown that our results can produce models that are equivalent to traditional versions of RFs through an alternative path, without computing a single tree and effectively opening a new way of analyzing regression estimates that conform to the proposed framework. While we believe that the ideal use of b-terms and Equation (13) is analytical, on a practical sense we believe to have uncovered a way for new classes of algorithms to arise, perhaps taking advantage of heuristics to overcome the exponential expansion

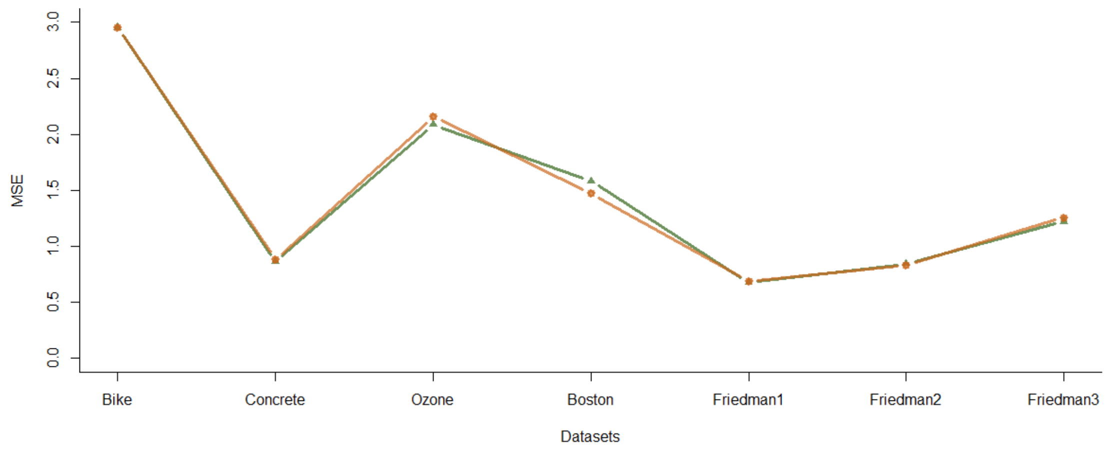


FIGURE 6. MSE comparisons between our regression estimate equipped with the L_{RF} (green triangles) and the RF method $\hat{f}_{RF}$ (orange circles) across the seven selected datasets.

of b-terms while making affordable compromises in MSE values.

VI. SUMMARY AND CONCLUSIONS

In this work we have shown advances in our understanding of the statistical forces behind RFs, by means of their analogy with the k-PNNs. We first discovered that the developments to obtain the bagged 1-NN regression estimate in [3] could be extended to show the calculation of the bootstrap weights for k-PNN based regression estimates when substituting the corresponding monotone distance for its alternative PNN distance.

Then, we analyzed the influence of adding a point selection strategy to the previous results. A point selection strategy, such as any splitting criterion in a RF, turned out to act as an additional selector of k datapoints within the k-PNNs, causing some of k-PNNs to be finally selected, and some others to be not. Thus, the weights assigned to each datapoint had to be updated from bootstrap weights to the final weights. We first proved that the bootstrap weights act as upper bounds of the final weights for a RF equipped with any splitting criterion. Furthermore, we obtained an explicit expression for the final weights considering a specific point selection strategy in the k-PNN, the random k-PNN selection, which induces the bagged regression estimate $\hat{f}^*_{RkS}(\mathbf{x}_0)$, and showed that $\hat{f}^*_{RkS}(\mathbf{x}_0) = \hat{f}^*_{k-PNN}(\mathbf{x}_0)$. In doing so, we created the concept of b-terms as a list of inclusion/non-inclusion restrictions on the datapoints present in all bootstrap variations, defined the value of a b-term to be the proportion of bootstrap variations that comply with the list (and derived a mathematical expression to calculate that value). We then showed that datapoint weights can be expressed as sums of b-terms coupled with a local weight on each b-term.

Further understanding uncovered a framework for bagging estimators that included all classes of RF with a k datapoints stopping criterion. With this, we derived the regression estimate that corresponds with a RF equipped with random splitting criterion and showed the case of $k = 1$ at a practical level, where MSE values of our regression estimate and a full RF implemented in the classical way w.r.t. the real values were virtually identical. We were also able to conduct additional practical experiments that revealed how b-terms and k-PNN distance can be used to analyze the effect of bootstrapping in bagged regression estimators in contrast with the effect of the point selection strategy (splitting criterion in RF). With this, we validated the $\hat{f}^*_{RkS}(\mathbf{x}_0)$ for $k = 1$ as a competent regression estimate. Our results suggest that it is recommended for problems with high scale disparity between features (since PNN are distance invariant) and high $\overline{PNN}$. Additionally, it is fast to implement (k-PNN calculation for a given datapoint with arbitrary k was $\mathcal{O}(n^2)$ in our methods) and intuitive to work with. It may also be a considerable choice over 1-NN (or bagged 1-NN) when the nearest neighbors assumptions (namely, that datapoints close in distance have also close y-associated values) do not hold.

We believe that the ideal use of our work would be as an analysis tool for other regression estimates and as a design platform for variants of random forests. In this setting, a researcher may follow a similar path to the one shown in the paper to write a regression estimate as a weighted sum of datapoints were the weights are expressed as weighted sums of b-terms. Then, analysis on the particular form of that expression may allow for simplifications previously inaccessible, detection of grouping patterns/regularities in the addends of the sum and in general, to enjoy a higher degree of algebraic manipulation than in the initial proposal for that regression estimate.

This work opens numerous possibilities for regression estimates to have an alternative written form (as weighted sums of b-terms) that has desirable properties. Perhaps reductions in computational complexity for the calculation of well-known methods, that have remained hidden so far, can now be unlocked, and Monte Carlo simulation as the preferred method for computation of those regression estimators can be eschewed.

As a future work, we believe to have obtained the necessary tools to tackle the specification of other splitting criteria in RFs in terms of weighted PNNs, as well as have provided access to a new form of analysis of RF models and other regression estimates

REFERENCES

[1] Z.-D. Bai, L. Devroye, H.-K. Hwang, and T.-H. Tsai, "Maxima in hypercubes," *Random Struct. Algorithms*, vol. 27, no. 3, pp. 290–309, 2005.

[2] O. E. Barndorff-Nielsen and M. Sobel, "On the distribution of the number of admissible points in a vector random sample," *Theory Probab. Appl.*, vol. 11, no. 2, pp. 249–269, 1966.

[3] G. Biau, F. Cerou, and A. Guyader, "On the rate of convergence of the bagged nearest neighbor estimate," *J. Mach. Learn. Res.*, vol. 11, pp. 687–712, Jan. 2010.

[4] G. Biau and L. Devroye, "On the layered nearest neighbour estimate, the bagged nearest neighbour estimate and the random forest method in regression and classification," *J. Multivariate Anal.*, vol. 101, no. 10, pp. 2499–2518, 2010.

[5] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[6] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[7] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[8] P. Büchlmann and B. Yu, "The annals of statistics," *Ann. Statist.*, vol. 30, no. 4, pp. 927–961, 2002.

[9] B. Caprile, S. Merler, C. Furlanello, and G. Jurman, "Exact bagging with k-nearest neighbour classifiers," in *Multiple Classifier Systems*. Berlin, Germany: Springer, 2004, pp. 72–81.

[10] A. Cutler and G. Zhao, "Pert-perfect random tree ensembles," in *Computing Science and Statistics*, vol. 33. Fairfax Station, VA, USA: Interface Foundation of North America, 2001, pp. 490–497.

[11] L. Devroye, L. Györfi, and G. Lugosi," *A Probabilistic Theory of Pattern Recognition*, vol. 31. Berlin, Germany: Springer, 1996.

[12] E. Fix and J. Hodges, "Discriminatory analysis: small sample performance," USAF School Aviation Med., Randolph Field, TX, USA, Tech. Rep. 21-49-004, 1952

[13] E. Fix and J. Hodges, "Discriminatory analysis. Nonparametric discrimination: Consistency properties," USAF School Aviation Med., Randolph Field, TX, USA, Tech. Rep. 4, 1951.

[14] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[15] N. El Karoui and E. Purdom. (2016). "Can we trust the bootstrap in high-dimension?" [Online]. Available: https://arxiv.org/abs/1608.00696

[16] Y. Lin and Y. Jeon, "Random forests and adaptive nearest neighbors," *J. Amer. Statist. Assoc.*, vol. 101, no. 474, pp. 578–590, Jun. 2002.

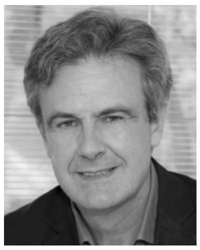
[17] R. J. Samworth, "Optimal weighted nearest neighbour classifiers," *Ann. Statist.*, vol. 40, no. 5, pp. 2733–2763, 2012.

[18] B. M. Steele, "Exact bootstrap k-nearest neighbor learners," *Mach. Learn.*, vol. 74, no. 3, pp. 235–255, Mar. 2009.

[19] C. J. Stone, "Consistent nonparametric regression," *Ann. Statist.*, vol. 5, no. 4, pp. 595–620, 1977.

PABLO FERNÁNDEZ-GONZÁLEZ received the degree in technical informatics engineering from the School of Computer Science, University of Oviedo, Spain, in 2012, and the master's degree in artificial intelligence from the Technical University of Madrid, in 2014, where he is currently pursuing the Ph.D. degree with the Artificial Intelligence Department, participating in the Cajal Blue Brain Project. His research interests include mathematics, machine learning, data mining, random forests, and probabilistic graphical models.

CONCHA BIELZA received the M.S. degree in mathematics from the Universidad Complutense de Madrid, Madrid, Spain, in 1989, and the Ph.D. degree in computer science from the Universidad Politécnica de Madrid, Madrid, in 1996, where she has been a Full Professor of statistics and operations research with the Departamento de Inteligencia Artificial, since 2010. She has supervised 12 Ph.D. theses. She has published more than 100 papers in impact factor journals. Her research interests are primarily in the areas of probabilistic graphical models, decision analysis, metaheuristics for optimization, data mining, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience, and industry. She received the 2014 UPM Research Prize and the Extraordinary Doctorate Award.



PEDRO LARRAÑAGA received the M.Sc. degree in mathematics (statistics) from the University of Valladolid and the Ph.D. degree in computer science from the University of the Basque Country. His academic career has been developed at the University of the Basque Country at several faculty ranks, as an Assistant Professor, from 1985 to 1998, as an Associate Professor, from 1998 to 2004, and as a Full Professor, from 2004 to 2007. He earned the Habilitation qualification for Full Professor, in 2003. He has been a Full Professor in computer science and artificial intelligence with the Technical University of Madrid (UPM), since 2007. His research interests are primarily in the areas of probabilistic graphical models, metaheuristics for optimization, data mining, classification models, and real applications, such as biomedicine, bioinformatics, and neuroscience. He has supervised more than 25 Ph.D. theses. He has published more than 200 papers in impact factor journals. He has been an ECCAI Fellow, since 2012, and a Fellow of the Academia Europaea, since 2018. He received the 2013 Spanish National Prize in computer science and the Prize of the Spanish Association for Artificial Intelligence, in 2018. He received the Excellence Award from the University of the Basque Country.