frontiers in Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 21 December 2018
doi: 10.3389/fams.2018.00063

A Tree-Based Multiscale Regression Method

Haiyan Cai and Qingtang Jiang*

The Department of Mathematics and Computer Science, University of Missouri–St. Louis, St. Louis, MO, United States

A tree-based method for regression is proposed. In a high dimensional feature space, the method has the ability to adapt to the lower intrinsic dimension of data if the data possess such a property so that reliable statistical estimates can be performed without being hindered by the "curse of dimensionality." The method is also capable of producing a smoother estimate for a regression function than those from standard tree methods in the region where the function is smooth and also being more sensitive to discontinuities of the function than smoothing splines or other kernel methods. The estimation process in this method consists of three components: a random projection procedure that generates partitions of the feature space, a wavelet-like orthogonal system defined on a tree that allows for a thresholding estimation of the regression function based on that tree and, finally, an averaging process that averages a number of estimates from independently generated random projection trees.

Keywords: regression, non-linear, high dimension data, tree methods, multiscale (MS) modeling, manifold learning

OPEN ACCESS

Edited by:
Xiaoming Huo,
Georgia Institute of Technology,
United States

Reviewed by:
Don Hong,
Middle Tennessee State University,
United States
Shao-Bo Lin,
Wenzhou University, China

****Correspondence:***
Haiyan Cai
caih@umsl.edu

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *07 October 2018*
Accepted: *07 December 2018*
Published: *21 December 2018*

Citation:
Cai H and Jiang Q (2018) A Tree-Based Multiscale Regression Method. Front. Appl. Math. Stat. 4:63. doi: 10.3389/fams.2018.00063

1. INTRODUCTION

We consider the problem of estimating the unknown function in the model:

$$Y = f(X) + \varepsilon, X \in \mathbb{R}^p, Y \in R \tag{1}$$

with a sample $(x_i, y_i), i = 1, ..., n$, where ε is a random variable with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$. We are mainly interested in the case where the dimension p of the feature variable X is large, either relative to the sample size n of data or in itself, although the method of this paper also applies well to lower dimensional cases. When p is large, we assume that the domain of the function f is actually restricted to a much lower dimensional but unknown sub-manifold of $\mathbb{R}^p$. We refer to the dimension of this sub-manifold as the intrinsic dimension of the regression problem.

An important question to ask in this regression setting is that without having to learn the sub-manifold first, can one find an estimator for f that automatically adapts to the intrinsic low dimensional structure of data in the sense that it can still provide efficient and reliable estimates, without being hindered by the "curse of dimensionality" [1, 2] due to the large p? Bickel and Li [3] provide an affirmative answer to this question by showing that local polynomial regressions can achieve such a property under regularity conditions so that the decay of the prediction error in sample size depends only on the intrinsic dimension rather than p.

It turns out that the local polynomial regression is not the only approach that is capable of being adaptive to the intrinsic dimension of data. This is a more general phenomenon. Tree-based methods are useful and efficient alternatives and can be computationally more efficient. A work of Binev et al. [4, 5] provides arguments implying that certain tree-based approximations are adaptive. Later, Dasgupta and Freund [6] and Kpotufe and Dasgupta [7] demonstrate explicitly

such adaptability for their random projection trees when the intrinsic dimension is defined to be the so-called doubling dimension. A tree-based regression [7–11] is to partition recursively the feature space $\mathbb{R}^p$ into finer and finer subsets and then estimate f locally within each finest subset. Such a process has a naturally hierarchical tree structure. The corresponding tree is called a partitioning tree. An extension of the tree methods is the random-forest approach [8]. In this approach, an average of individual tree estimates based on bootstrap samples is used. In addition to the adaptive property, Dasgupta et al. [6] and Kpotufe and Dasgupta [7] argue that tree methods can be computationally more efficient than kernel smoothing methods, including the local polynomial regressions, or k-nearest neighbor methods, because while it takes only $O(\log n)$ steps, the height of a balanced tree, for a tree method to reach to a prediction, it takes $\Omega(n)$ for a kernel-type method and $\Omega(n^{1/2})$ for a k-nearest neighbor method to obtain a prediction.

We note that most of the tree-based regression methods depend exclusively on the means of y_i's over the partitioning subsets in constructing their partitioning trees, and use the means of the tree leaves for final estimates. While such estimates can be MLEs of the corresponding sampling distributions, they need not be optimal in terms of minimax theory [12]. Better nonparametric estimates can often be obtained through shrinkage and thresholding of the mean values. Some recent developments provide a framework for a wavelet-like representation and analysis for tree-structure data. These methods adapt naturally to the shrinkage and thresholding processes in estimation. The unbalanced Haar orthogonal systems (also called the Haar-type wavelet orthogonal systems) on a given tree were constructed by several authors [13–15]. We note that the unbalanced Haar wavelets were first constructed in Mitrea [16] for dyadic cubes in $\mathbb{R}^p$ and were later generalized in Girardi and Sweldens [15] to more general trees.

With the background above, we propose in this paper to integrate the wavelet-like analysis on trees with some efficient tree partitioning method so that an estimator for f that has the ability of adapting to the intrinsic low dimension of data can be obtained. However, our numerical experiments show that a simple combination of those ideas does not work well. Note that the ideas and results discussed above are mostly theoretical. There is little discussion in the literature on the actual numerical performance of those approaches. For example, the partition methods discussed in Binev et al. [4, 5], while possessing very nice analytical properties (the rates of convergence), would have a computational complexity that grows exponentially in p. Our numerical experiments also indicate that, when compared to the standard methods like random-forest, support vector machines, or even classical CART, the random projection tree regression of Dasgupta et al. [6] and Kpotufe and Dasgupta [7] or a wavelet soft-thresholding method based on trees generated through various partitioning procedures, including CART and random projection, do not show much, if any, significant improvements in terms of prediction error. A better approach is needed in order to fulfill the theoretical potential of these ideas. We propose a regression estimator in this paper that mitigates the difficulties. Our focus in this paper is mainly the methodology.

We only consider binary trees in this paper. The same ideas can be easily applied to more general trees. The tree-based estimator proposed here consists of three components. The first component is a partitioning algorithm. In principle any reasonable binary classifier, supervised or un-supervised, can be a candidate. The random projection algorithm of Dasgupta et al. [6] and Kpotufe and Dasgupta [7] or uniform partition of Binev et al. [4, 5] are un-supervised. For the kind of data we have, we however prefer supervised classifiers so that the information from y_i's can be utilized. We propose one such classifier below. The second component is a wavelet-like tree-based estimation process that incorporates thresholding and shrinkage operations. It turns out that this can be done in a statistically rather intuitive manner. The final component, a necessary and crucial one, is an averaging process, taking the average of the estimates across several independently generated partitioning trees on the same training data. This averaging process is different from that of a random-forest. In random-forest, the average is taken over bootstrapped samples, and our experiments indicate that it does not work well in some problems. We will call our method "averaging random tree regression."

While proposed as a computationally feasible algorithm for regression with low intrinsic dimensional data, our method also possesses other good properties. This estimator is to be compared through numerical experiments to some standard non-parametric estimators using higher dimensional data. We will also compare it in a low dimensional setting to a smoothing spline and random-forest. In low dimensional experiments, we observe that the proposed estimator is capable producing visually smoother estimates than CART-based estimators and, at the same time, being much more sensitive than a smoothing spline estimator to discontinuities of f, as it would be expected.

This paper is organized as follows. Section 2 gives a detailed description of the regression method and demonstrates some interesting features of the method. Section 3 gives some examples of our method. Section 4 provides a formulation of the method in terms of wavelet-like multilevel analysis on a tree. It includes a formal description of decomposition and reconstruction algorithms for the soft-thresholding estimation and an analytical study that establishes properties necessary for the thresholding method to work.

2. THE AVERAGING RANDOM TREE REGRESSION

Suppose a dataset $\{(x_i, y_i), x_i \in \mathbb{R}^p, y_i \in R, i = 1, ..., n\}$ is given. Let $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{Y} = \{y_1, ..., y_n\}$. For any subset $B \subset \mathcal{X}$, we will write $\bar{y}_B$ for the mean of y_is in B: $\sum_{x_i \in B} y_i / n_B$, where n_B denotes the size of B. Our tree-based method consists of three parts. We give a description for each of them below.

2.1. A Classifier for Space Partition and the Partitioning Tree

We first need a binary classifier which acts on the subsets of $\mathcal{X}$ and partitions any given subset $A \subset \mathcal{X}$ of at least two points into two finer subsets based on some classification criterion. Note

that the partitioning method of the classical CART regression can certainly be one of such classifiers, but it can be computationally inefficient for higher dimensional data. Here we propose a different classifier based on the combination of an extension of the random projection proposed in Dasgupta and Freund [6] and Kpotufe and Dasgupta [7] and the idea of minimum sum of squares criterion of CART. The original random projection partition method do not utilize any information from y_i's. This is a weakness of that method and we try to avoid it in the current approach. The idea is to choose among several random projection partitions the one that has the least mean square error for y_i's. The partitioning classifier works as follows.

Let M be a preset integer number. For any subset A of $\mathcal{X}$ with r points, we generate M unit directional vectors $v_1, ..., v_M$ in $\mathbb{R}^p$ from a priori distribution (an uniform distribution for example). For each $v_j, j = 1, ...M$, we project all the p-dimensional points x in A onto this unit vector to obtain a set of r scalars $v_j^T x$. Let m_{v_j} be the median of these scalars. Let

$$A_{L,j} = \{x \in A : v_j^T x < m_{v_j}\} \text{ and } A_{R,j} = \{x \in A : v_j^T x > m_{v_j}\}.$$

If there are one or more $x_i \in A$, $i = 1, ..., r \geq 1$ such that $v_j^T x_i = m_{v_j}$, we split these points randomly with equal probability into subsets of $\lfloor r \rfloor$ (floor of r) and $\lceil r \rceil$ (ceiling of r) points and assign the subsets with equal probability to $A_{L,j}$ or $A_{R,j}$ and still denote these subsets as $A_{L,j}$ and $A_{R,j}$. In this way we obtain the partition $A = A_{L,j} \cup A_{R,j}$. Next we find the sum of squares for this directional vector v_j:

$$S(v_j) = \sum_{x_i \in A_{L,j}} (y_i - \bar{y}_{A_{L,j}})^2 + \sum_{x_i \in A_{R,j}} (y_i - \bar{y}_{A_{R,j}})^2. \quad (2)$$

We determine the vector $v \in \{v_1, ..., v_M\}$ with the smallest $S(v_j)$ value:

$$v = \arg_{v_j,\, j=1,...,M} \min S(v_j),$$

and choose the corresponding partitioning subsets A_L and A_R as the final partition of A and let m_v be the corresponding dividing median. Let us denote the above process of obtaining a binary partition for a set A as

$$\pi_M(A) = \{A_L, A_R, v, m_v\}, \quad (3)$$

where M is an adjustable parameter. Clearly, the outcomes of π_M are random, depending on realizations of the random vectors $v_1, ..., v_M$. The reason for using median as the splitting point for partition is to keep the corresponding partition tree balanced. A balanced tree possesses some nice analytical properties for wavelet-like analysis on the tree, as we will see later.

With the partitioning classifier π_M, we construct recursively a partitioning tree $\mathcal{T}$ so that each node in the tree $\mathcal{T}$ is a subset of $\mathcal{X}$ and the children nodes are partitioning subsets of their parent nodes. In other words, we set the root of $\mathcal{T}$ to be the whole point set $\mathcal{X}$ and, starting from the root, we keep partitioning each node into two children nodes with about the same size until a node becomes a singleton set. The leaves of $\mathcal{T}$ are singleton sets of $\mathcal{X}$.

In this construction, each non-leaf node is associated with a pair (v, m_v) of a unit directional vector and the corresponding dividing median. All the pairs are pre-calculated and stored, and will serve as the parameters for partitioning the whole feature space $\mathbb{R}^p$. The space $\mathbb{R}^p$ can now be partitioned into disjoint regions identified by the leaves of $\mathcal{T}$ as follows. To begin with, we classify every point $x \in \mathbb{R}^p$ as an "$\mathcal{X}$ point." Next, for any $x \in \mathbb{R}^p$, if it is already classified as an "A point" for some tree node $A \subset \mathcal{X}$ of at least two points and if $\pi_M(A) = \{A_L, A_R, v, m_v\}$, we calculate the projection of x onto v and then classify x further as either an "A_L point" if $v^T x < m_v$, or an "A_R point" if $v^T x > m_v$. In the case of $v^T x = m_v$, we classify x with equal probability into either A_L or A_R. This classification process goes on and will eventually classify x into one and only one of the leaf nodes. If x is ultimately classified as a leaf node "$\{x_i\}$ point" for some $x_i \in \mathcal{X}$, we write $\mathcal{T}(x) = x_i$. For every $x_i \in \mathcal{X}$, let

$$A_i := \{x \in \mathbb{R}^p : \mathcal{T}(x) = x_i\}. \quad (4)$$

Then $A_i, i = 1, ..., n$ forms a partition of $\mathbb{R}^p$ according to the partitioning tree $\mathcal{T}$.

These trees are maximal trees. In order to preserve more local features of f in data, we do not prune them like CART. In fact, a default pruning process in CART destroys its ability to be adaptive to the intrinsic low dimension of data. On the other hand, as an implementation issue, if all y_is in a node A have very close values so that $S(v_j)$ is small in all M selected directions, we can stop splitting this node and treat it as a leaf and use the mean of y_is as the y-value for this node. All computations below can still be carried out without change. The overfitting issue caused by a large tree will be addressed in the last step of our estimation process to be described later.

2.2. A Multiscale Soft-Thresholding Estimate of $f(x_i)$

As we will see in section 4, the procedure described below is actually a modified version of a common wavelet denoising process with an unbalanced Haar wavelet on a tree. But since this process can also be described in a statistically more intuitive manner with simpler notations, we give the following algorithmic description first.

Suppose a partitioning tree $\mathcal{T}$ is obtained. A hierarchical representation of the data $\mathcal{X} \times \mathcal{Y} = \{(x_i, y_i), i = 1, ..., n\}$ can be obtained according to the tree $\mathcal{T}$ as follows. For each non-leaf node in the tree $A \subset \mathcal{X}$, we find the mean $\bar{y}_A$ of the node and the difference of the means of its children nodes A_L and A_R:

$$d_A := \bar{y}_{A_L} - \bar{y}_{A_R}. \quad (5)$$

If A is a leaf, we set $d_A = 0$. The original data can now be represented with the set of the numbers

$$\mathcal{D} = \{\bar{y}_{\mathcal{X}}\} \cup \{d_A,\, A \in \mathcal{T}\}$$

based on which regression estimates will be obtained.

To estimate $\hat{f}(x_i), i = 1, ..., n$, we first apply a soft-thresholding operation with a given $\alpha \geq 0$ to d_A for all the non-leaf nodes A according to the formula

$$\hat{d}_A = \text{sign}(d_A) \max \left\{ 0, |d_A| - \alpha \sqrt{\frac{1}{|A_L|^2} + \frac{1}{|A_R|^2}} \right\}. \quad (6)$$

Next, we calculate estimates $\hat{y}_A$ of $E(\bar{y}_A|X = x)$ for all nodes A of $\mathcal{T}$ based on the data

$$\hat{\mathcal{D}} = \{\bar{y}_{\mathcal{X}}\} \cup \{\hat{d}_A,\ A \in \mathcal{T}\}$$

as follows. We start with setting

$$\hat{y}_{\mathcal{X}} = \bar{y}_{\mathcal{X}}.$$

For each node A, once $\hat{y}_A$ is obtained, we calculate the estimates for its two children nodes, $\hat{y}_{A_L}$ and $\hat{y}_{A_R}$, according to the formula:

$$\hat{y}_{A_L} = \hat{y}_A + \frac{|A_R|}{|A|}\hat{d}_A \quad (7)$$

and

$$\hat{y}_{A_R} = \hat{y}_A - \frac{|A_L|}{|A|}\hat{d}_A. \quad (8)$$

Repeat this process until one reaches all the leaves. If $A = \{x_i\}$ is a leaf, we use

$$\hat{f}(x_i) := \hat{y}_{\{x_i\}}$$

as an estimate of $f(x_i)$. We will write $\hat{y}_i$ for $\hat{y}_{\{x_i\}}$ for short below.

Note that, if $\alpha = 0$ in (6), then $\hat{d}_A = d_A$ for all nodes A and, as a consequence, $\hat{y}_A = \bar{y}_A$ for all $A \in \mathcal{T}$. To see this, we note that $\hat{y}_{\mathcal{X}} = \bar{y}_{\mathcal{X}}$, and if we have $\hat{y}_A = \bar{y}_A$ and $\hat{d}_A = d_A$, then

$$\hat{y}_{A_L} = \bar{y}_A + \frac{|A_R|}{|A|}(\bar{y}_{A_L} - \bar{y}_{A_R}) = \bar{y}_{A_L}$$

and

$$\hat{y}_{A_R} = \bar{y}_A - \frac{|A_L|}{|A|}(\bar{y}_{A_L} - \bar{y}_{A_R}) = \bar{y}_{A_R}.$$

Also note that $|A_L|$ and $|A_R|$ are differed by at most 1 and therefore for large nodes, $|A_L|/|A| \simeq |A_R|/|A| \simeq 1/2$. The smoothing parameter α can be determined through cross-validation.

In the framework of wavelet analysis, $\bar{d}_A$ are the wavelet coefficients and (7)–(8) is the fast wavelet reconstruction algorithm to be derived in section 4.

2.3. The Averaging Random Tree Regression

Now to estimate $f(x)$ for any $x \in \mathbb{R}^p$, we can in principle use

$$\hat{f}(x) = \sum_{x_i \in \mathcal{X}} \hat{y}_i \chi_{A_i}(x). \quad (9)$$

In other words, a search along the tree $\mathcal{T}$ (with $O(\log_2 n)$ steps) will allow us to determined for each $x \in \mathbb{R}^p$ which partitioning subset A_i, as defined in (4), it belongs to and then use $\hat{y}_i$ as an estimate for $f(x)$. However, despite some nice theoretical properties, this estimate itself doesn't perform very well in our simulation experiments. It turns out that a significant improvement can be achieved by an averaging procedure described below.

For a given integer K, we repeat the process described in Section 2.1 K times, independently, on the same data to obtain trees $\mathcal{T}_k, k = 1, ..., K$. For each tree $\mathcal{T}_k$, we calculate estimates $\hat{f}_k(x)$ according to (9). Finally, we take the average

$$\hat{f}_*(x) = \frac{1}{K}\sum_{k=1}^{K} \hat{f}_k(x) \quad (10)$$

as the final estimate of $f(x)$. Let's call this estimate an "Averaging Random Tree Regression" estimate, or ARTR estimate for short. This averaging step improves significantly the accuracy of the estimates for the regression function. The resulting estimates can be adaptive to lower intrinsic dimension of data. It can also be visually smoother than other tree-based regression methods in two or three dimensional cases even with piecewise constant Haar wavelets, and being sensitive to discontinuities at the same time. It also addresses efficiently the overfitting problem.

We summarize the process into the following steps:

1. Generate a partitioning tree $\mathcal{T}$ according to classifier π_M.
2. Obtain $\bar{y}_{\mathcal{X}}$ and $d_A = \bar{y}_{A_L} - \bar{y}_{A_R}$ for all non-leave nodes $A \in \mathcal{T}$ and $A_L, A_R \in \pi_M(A)$.
3. Obtain $\hat{\mathcal{D}}$ from $\mathcal{D}$ based on (6) for a given α and $\hat{y}_i, i = 1, ..., n$, through recursively applying (7) and (8).
4. Calculate the estimate $\hat{f}$ of f using (9).
5. Repeat steps 1–4 K times and take the average according to (10).

There are three tuning parameters in this process: M, the number of random projection directions for each partition in generating a partitioning tree; K, the number of trees to generate for averaging; and α, a factor in the threshold for smoothing. Cross-validation can be used to choose the values of these parameters.

Simulation studies show that ARTR estimation outperforms some standard methods, as can be seen through the examples in the following subsection. Note that our averaging approach is similar in its appearance to but different in principle from that of the random forests. In fact, the random forests approach which averages tree estimates based on resampling dataset does not produce good results in our experiments. Formally, our estimate can be viewed as a kernel method which takes a weighted average of nearby data points to obtain an estimate.

3. EXAMPLES OF APPLICATIONS TO SOME STATISTICAL PROBLEMS

We give three examples demonstrating potential applications of the ARTR method. In all computations below, we set the number

of random projections for partitioning $M = 10$ and the number of partition trees for averaging $K = 36$.

Example 1: Regression on a Lower Intrinsic Dimensional Manifold

This example is about high dimensional regression where the values of predictors are actually from an embedded unknown sub-manifold with a much lower dimension. In our example, this sub-manifold is the surface of a two-dimensional "Swiss roll" given by the parametric equations: $x_1 = u\cos u$, $x_2 = u\sin u$, $x_3 = v$, for $u, v \in [0, 4\pi]$. The regression function is

$$f(x_1, x_2, x_3) = \left(x_3 - \sqrt{x_1^2 + x_2^2}\right)^2/2 = (v-u)^2/2$$

and $\varepsilon \sim N(0,1)$. Our data do not contain exact values of (x_1, x_2, x_3). Rather, all the 3-dimensional points are embedded isometrically (with an arbitrary and unknown isometric transformation) into the space $\mathbb{R}^{4,000}$. Therefore, each feature point is recorded as a $p = 4,000$ dimensional point. We apply support vector machine regression (SVMR), random forests regression (RFR), extreme gradient boost regression (XGBR) with the parameter *nrounds* set to 100, our ART regression (ARTR) without averaging ($K = 1$) and with averaging ($K = 36$) to a training dataset of $n = 1,000$ points and then apply the estimated models from these different methods to an independent testing dataset of 1,000 points. Predictions of the values for the function are obtained at these 1,000 testing points. The approximation error at a point x is the difference between the predicted value at x and the true value of the function at x. **Table 1** shows the mean squared approximation errors in this computation. We see that in terms of the mean approximation errors, ARTR with $K = 36$ is better than SVMR, and significantly better than RFR or XGBR. We also see that without taking average, ARTR with $K = 1$ has the worse performance. In ARTR we have used $\alpha = 2.0$.

Example 2: Discovering Mixtures in a Higher Dimensional Space

This is a higher dimensional example. The problem in this example can be described in more general terms as follows. Suppose S_1 and S_2 are two unknown lower dimensional sub-manifolds embedded in $\mathbb{R}^p$. Suppose that data are sampled from $S_1 \cup S_2$ and that y_i's sampled from S_1 have distribution $N(\mu_1, \sigma^2)$ and y_i's from S_2 has distribution $N(\mu_2, \sigma^2)$. In other words,

$$y = \mu_1 \chi_{S_1}(x) + \mu_2 \chi_{S_2}(x) + \varepsilon,\ x \in S_1 \cup S_2,$$

where $\varepsilon \sim N(0, \sigma^2)$. We ask, from the data can we discover that y_i's are a mixture of two different distributions? We show that ARTR can achieve this while some standard method fails in the following experiment.

In this example, S_1 and S_2 are two intersecting unit 2-spheres embedded into a $p = 6,000$ dimensional space $\mathbb{R}^{6,000}$. The centers of the 2-spheres are set to be 0.35 apart. We set $\mu_1 = 0$, $\mu_2 = 2$, and $\sigma = 1$. The data consists of $n = 4,000$ sample points. A histogram of y_i's would reveal no signs that y_i are from a mixture of two distributions. Four different methods are applied to fit the data: SVMR, RFR, XGBR, all with the default settings in R packages "svm," "randomForest," and "xgboost" (*nround*=100), and our ART regression. Estimates $\hat{y}_i$ of $f(x_i)$ are obtained from the methods and their histograms are displayed also in **Figure 1**. For the ARTR we choose $K = 36$ and $\alpha = 2.0$.

We see from the histograms in **Figure 1** that SVMR fails to recognize correctly the underlying lower dimensional structure of the data. It mistakenly treats the central region bounded by the intersection of the two spheres as the third region in which y_i's have a mean value $(\mu_1 + \mu_2)/2 = 1$. A much larger sample size is needed for SVMR to achieve a correct estimate, demonstrating the effect of the curse of dimensionality on the method. The RFR is capable of noticing the mixture and XGBR is better than RFR, but ARTR is clearly much more powerful than all others in detecting the mixture. This comparison between ARTR and RFR or XGBR supports the comment we make in the introduction section that a mean-based estimate can be significantly improved through shrinkage and thresholding operations.

The mean squared errors, calculated using $\hat{y}_i, i = 1, ..., 4,000$, and the true means $\mu_1 = 0$ and $\mu_2 = 2$, are listed in **Table 2**. It shows that ARTR has the smallest mean squared error among the four methods. We also notice that, while the mean squared error of SVMR is smaller than those of RFR or XGBR, the estimates from SVMR can be totally misleading.

Example 3: Smoothness and Sensitivity to Discontinuities

This is a two dimensional example to show smoothness and sensitivity to discontinuities of ARTR estimates. We do this through comparing ARTR to the thin plate splines (TPS) and random forests regression (RFR). The regression function we use is

$$f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x)$$

with

$$\begin{aligned} f_1(x) &= 10e^{-2.5[x_1^2 + (x_2 - 0.5)^2]} \\ f_2(x) &= 7e^{-3[(x_1-0.5)^2 + (x_2-0.5)^2]} \\ f_3(x) &= 4e^{-4[(x_1-0.5) + (x_2+0.5)^2]} \\ f_4(x) &= 4I(x_1 + x_2 < -0.5) \end{aligned}$$

and $\varepsilon \sim N(0,1)$. An image of $f(x)$ is given in **Figure 2** (in these figures, the valleys are shown in red and the peaks are shown in white). The data consist of $n = 4,000$ points with points in $\mathcal{X}$ generated uniformly inside a two-dimensional unit square. Setting $\alpha = 2.0$, we obtain ARTR estimates at 100×100 grid-points which is compared to the estimates at the same grid-points from the TPS and the RFR. In **Figure 2**, we observe that both TPS and ARTR produce smoother estimates than RFR does. Furthermore, the estimates from TPS and ARTR look more similar in local regions.

We note that the surface of the true regression function has a discontinuity line. **Figure 3** provides a comparison among

the three estimates. The two figures in the top row are the plots of errors from ARTR estimates and TPS estimates respectively. A difference between the two estimates along the discontinuity line is visible. The left plot in the second row of **Figure 3** shows an even more significant difference. This figure displays a difference in sensitivity to discontinuities between the two methods. Note that, while such a property is what one would expect from a wavelet-type method, estimates from ARTR are smoother than those from a Haar-type wavelet. In contrast, the difference between the TPS estimates and the RFR estimates, while visible, is more noisy and less definite. Overall, ARTR performs best in this example. The "ARTR vs TPS" figure in **Figure 3** suggests that one might even consider using such a figure, obtained completely from data, as a mean for detecting discontinuity of a noisy surface.

The mean squared approximation errors of the three methods from the same simulation computation are listed in **Table 3**. We see that in terms of approximation errors, ARTR is similar to (or slightly better than) TPS, and RFR is worse among the three.

TABLE 1 | The mean squared approximation errors in example 1.

ARTR, K=36	ARTR, $K = 1$	SVMR	RFR	XGBR
1.691675	9.329235	2.684860	7.934476	7.759801

4. WAVELET-LIKE ANALYSIS ON TREES

It is possible that the tree-based method proposed above be formulated and applicable within a more general context in which $\mathcal{X}$ can be an arbitrary point set with a given partitioning tree $\mathcal{T}$. In particular, $\mathcal{X}$ need not be a subset of $\mathbb{R}^p$. An important concept that makes an analysis possible for this setting is the tree metric [14] which characterizes the smoothness of the functions defined on $\mathcal{X}$ in terms of their tree wavelet coefficients. In this section we first give a formal description of a Haar wavelet-like orthogonal system on the tree $\mathcal{T}$, then we present a fast-algorithm for decomposition and reconstruction of functions defined on $\mathcal{X}$ according to the tree $\mathcal{T}$. This algorithm includes the algorithm in section 2 as a special case. The discussion below is for more general trees for which a leaf node can have more than one point and in this case the corresponding y-value is the average of y_is in the node.

4.1. Wavelet-Like Orthogonal Systems on $\mathcal{T}$

Without loss of generality, we can assume that the binary partitioning tree $\mathcal{T}$ has L levels, with the root $\mathcal{X}$ at level 0 and

TABLE 2 | The mean squared errors of the estimates in example 2.

ARTR	SVMR	RFR	XGBR
0.1605106	0.3114145	0.4000920	0.4136200

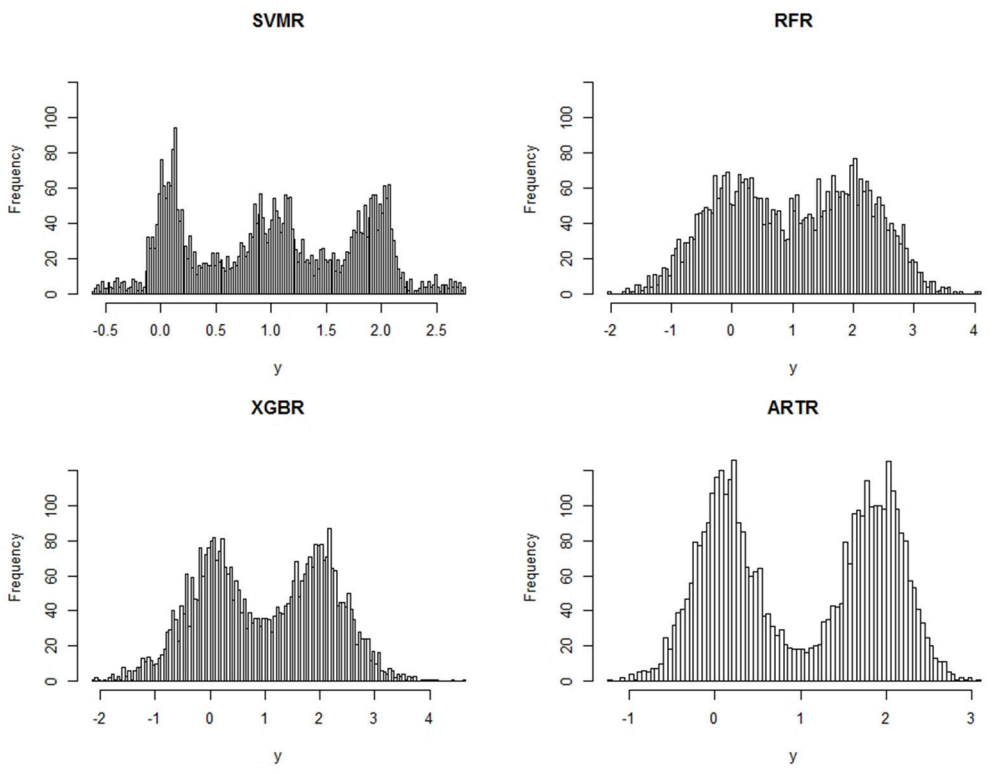

FIGURE 1 | Histograms of fitted data from different methods.

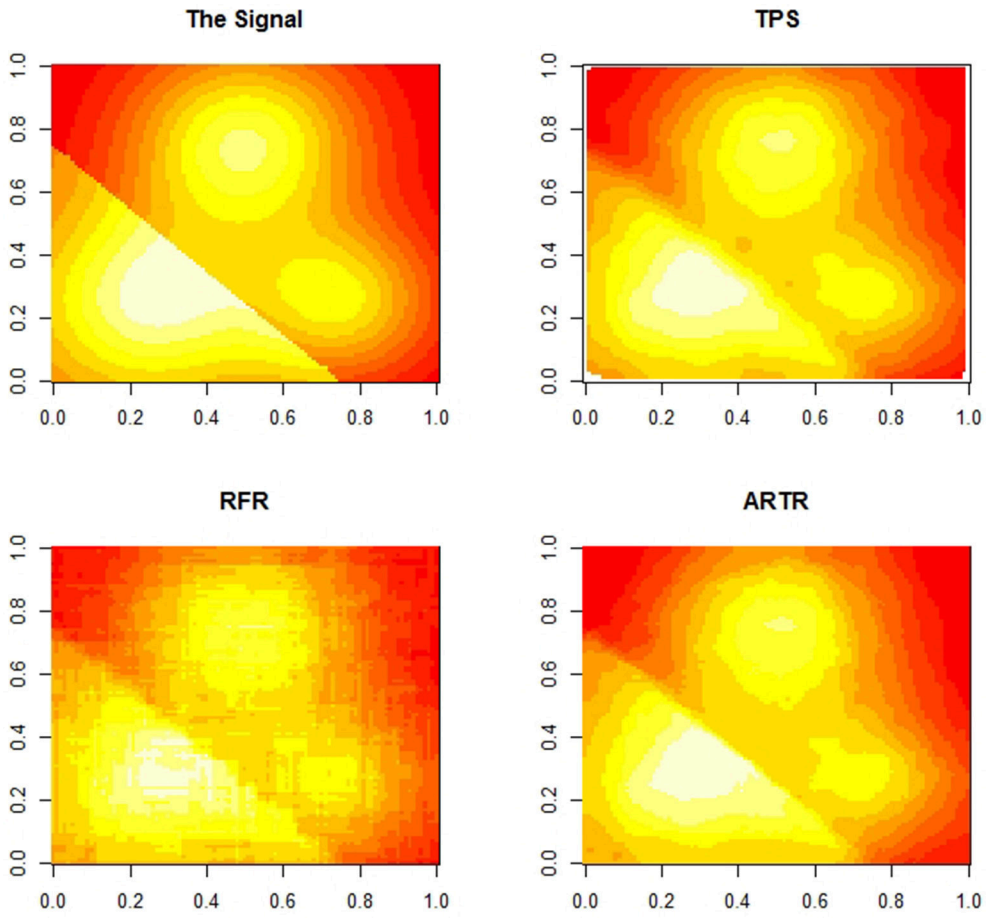

FIGURE 2 | The regression function and its estimates from different methods.

all the leaves at level L. This would be exactly the case for a tree $\mathcal{T}$ constructed in the previous section if $\mathcal{X}$ consists of 2^L points. To achieve this for an arbitrary $\mathcal{X}$ and tree $\mathcal{T}$, if A is a node of the tree at level $\ell < L$ which is a leaf node, we simply define its offspring at level $\ell + 1$ to be the node itself.

For each $\ell = 0, 1, ..., L$, we index all the nodes in $\mathcal{T}$ at level ℓ with an index set I_ℓ and let P_ℓ be the set of these nodes :

$$P_\ell = \{A_{\ell,j},\ j \in I_\ell\},\ 0 \le \ell \le L.$$

Then P_ℓ forms a partition of $\mathcal{X}$: $A_{\ell,i} \cap A_{\ell,j} = \emptyset$ for $i \neq j$ and $\mathcal{X} = \bigcup_{j \in I_\ell} A_{\ell,j}$. Further more, $P_{\ell+1}$ is a refinement of P_ℓ with

$$A_{\ell,j} = A_{\ell+1,j'} \cup A_{\ell+1,j''},\ A_{\ell+1,j'} \cap A_{\ell+1,j''} = \emptyset \qquad (11)$$

for some $A_{\ell+1,j'}, A_{\ell+1,j''} \in P_{\ell+1}$ if $|A_{\ell,j}| > 1$, and

$$A_{\ell,j} = A_{\ell+1,j'}$$

for some $A_{\ell+1,j'} \in P_{\ell+1}$ if $|A_{\ell,j}| = 1$. With these notations, $A_{0,0} = \mathcal{X}$ is the root and $A_{L,j}, j \in I_L$ are the leaves of $\mathcal{T}$.

A wavelet-like orthogonal system can now be defined on $\mathcal{T}$ as follows. Let

$$V = \{f | f : \mathcal{X} \to \mathbb{R}\}$$

be the space of all functions defined on $\mathcal{X}$, equipped with the inner product:

$$\langle f, g \rangle = \frac{1}{n} \sum_{x \in \mathcal{X}} f(x) g(x),\ f, g \in V. \qquad (12)$$

Let ν be the empirical probability measure of X induced by the set $\mathcal{X}$ of sample feature points:

$$\nu(A) = \frac{|A \cap \mathcal{X}|}{|\mathcal{X}|} = \frac{1}{n} |A \cap \mathcal{X}|,\ \forall A \subset C, \qquad (13)$$

with the understanding that ν depends on the sample size n. For a function f on $\mathcal{X}$, then

$$\|f\| := \left(\int_{\mathcal{X}} |f(x)|^2 d\nu(x) \right)^{\frac{1}{2}}$$

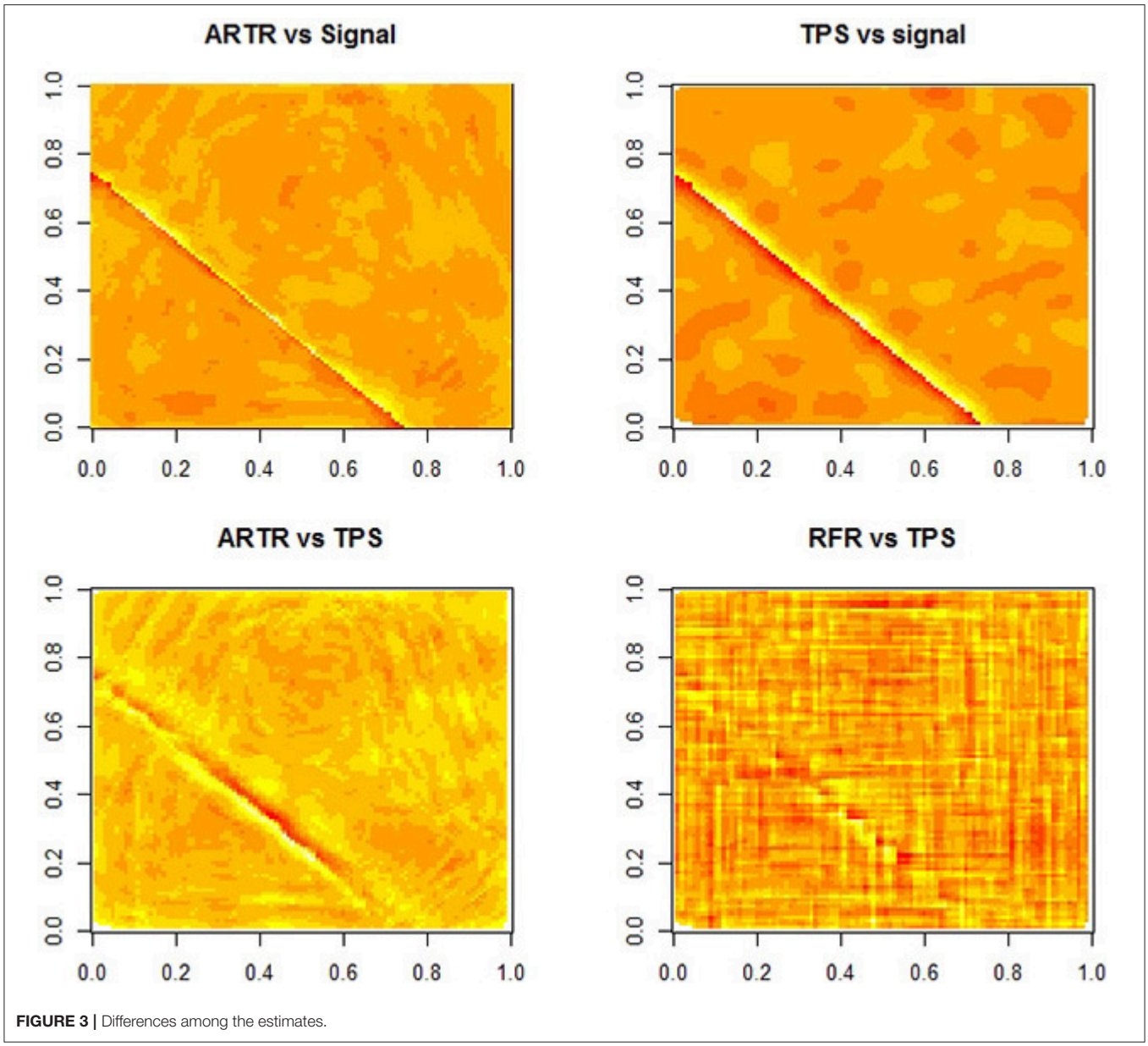

FIGURE 3 | Differences among the estimates.

TABLE 3 | The mean squared approximation errors in example 3.

ARTR	TPS	RFR
0.1267824	0.1363984	0.2560183

is just $\sqrt{\langle f,f\rangle}$.

An orthogonal system on V with respect to the inner product in (12) can now be constructed based on the partitioning tree $\mathcal{T}$. The following describes the construction of Haar-type wavelets on $\mathcal{T}$, [13–15].

For each $\ell = 0, ..., L$ and $P_\ell = \{A_{\ell,j} : j \in I_\ell\}$, let

$$V_\ell := \{f \in V : f\big|_{A_{\ell,j}} \text{ is a constant for every } j \in I_\ell\}.$$

Clearly, we have $V_\ell \subset V_{\ell+1}$.

Let $\phi_{\ell,j}, 0 \le \ell \le L$ and $j \in I_\ell$, be functions on $\mathcal{X}$ defined by

$$\phi_{\ell,j}(x) := \chi_{A_{\ell,j}}(x). \tag{14}$$

For each $\ell, 0 \le \ell \le L-1$, let $J_\ell \subset I_\ell$ denote the index set for those nodes $A_{\ell,j}$ with $|A_{\ell,j}| > 1$. For each $j \in J_\ell$, with $A_{\ell,j} = A_{\ell+1,j'} \cup A_{\ell+1,j''}$, let $\psi_{\ell,j}$ be functions defined by

$$\begin{aligned}\psi(x) :&= \frac{\nu(A_{\ell+1,j''})}{\nu(A_{\ell,j})}\chi_{A_{\ell+1,j'}}(x) - \frac{\nu(A_{\ell+1,j'})}{\nu(A_{\ell,j})}\chi_{A_{\ell+1,j''}}(x) \\ &= \frac{\nu(A_{\ell+1,j''})}{\nu(A_{\ell,j})}\phi_{\ell+1,j'}(x) - \frac{\nu(A_{\ell+1,j'})}{\nu(A_{\ell,j})}\phi_{\ell+1,j''}(x).\end{aligned} \tag{15}$$

Then we have

$$\langle \phi_{\ell,j}, \psi_{\ell',j'} \rangle = 0, \text{ for } \ell \le \ell',\ j' \in J_\ell$$
$$\langle \psi_{\ell,j}, \psi_{\ell,j'} \rangle = 0, \text{ for } j \ne j',\ j, j' \in J_\ell.$$

Denote

$$W_\ell = \text{span}\{\psi_{\ell,j}, j \in J_\ell\}.$$

The fact $V_\ell = \text{span}\{\phi_{\ell,j} : j \in I_\ell\}$ and the orthogonality of $\phi_{\ell,j}$ and $\psi_{\ell,j}$ imply that $W_\ell \subset V_{\ell+1}$ and $V_\ell \perp W_\ell$. In addition, by a direct calculation, we have, for $A_{\ell,j}, A_{\ell+1,j'}$, and $A_{\ell+1,j''}$ as related in (11) and $j \in J_\ell$,

$$\phi_{\ell+1,j'}(x) = \frac{\nu(A_{\ell+1,j'})}{\nu(A_{\ell,j})}\phi_{\ell,j}(x) - \psi_{\ell,j}(x),$$
$$\phi_{\ell+1,j''}(x) = \frac{\nu(A_{\ell+1,j''})}{\nu(A_{\ell,j})}\phi_{\ell,j}(x) - \psi_{\ell,j}(x),$$

which implies that $V_{\ell+1} \subseteq V_\ell + W_\ell$. Therefore, W_ℓ is the orthogonal complement of V_ℓ in $V_{\ell+1}$, i.e., $V_{\ell+1} = V_\ell \oplus^\perp W_\ell$, where $\oplus^\perp$ denotes the orthogonal sum. Hence

$$V_L = W_{L-1} \oplus^\perp V_{L-1} = W_{L-1} \oplus^\perp W_{L-2} \oplus^\perp V_{L-2} = \cdots$$

and finally

$$V_L = V_0 \oplus^\perp W_0 \oplus^\perp \cdots \oplus^\perp W_{L-1}. \tag{16}$$

From (16), we see $W_\ell \perp W_{\ell'}$ for $\ell \ne \ell'$. Hence $\psi_{\ell,j}, 0 \le \ell \le L-1,\ j \in J_\ell$ are orthogonal to each other. More precisely, we have

$$\langle \psi_{\ell,j}, \psi_{\ell',i} \rangle = B_{\ell,j}\ \delta(\ell - \ell')\delta(j - i),$$

for all $0 \le \ell, \ell' \le L-1, j \in J_\ell, i \in J_{\ell'}$, where

$$B_{\ell,j} := \langle \psi_{\ell,j}, \psi_{\ell,j} \rangle = \frac{\nu(A_{\ell+1,j'})\ \nu(A_{\ell+1,j''})}{\nu(A_{\ell,j})}. \tag{17}$$

We summarize this into the following theorem.

Theorem 1. *The system of Haar-type wavelets*

$$\psi_{\ell,j}(x),\ 0 \le \ell \le L-1, j \in J_\ell, \tag{18}$$

together with $\phi_{0,0}(x) \equiv 1$ form an orthogonal basis of V_L. More precisely, V_L can be decomposed as (16) with $f \in V_L$ represented as

$$f(x) = \langle f, \phi_{0,0} \rangle \phi_{0,0}(x) + \sum_{\ell=0}^{L-1} \sum_{j \in J_\ell} \frac{1}{B_{\ell,j}} \langle f, \psi_{\ell,j} \rangle \psi_{\ell,j}(x), \tag{19}$$

where $B_{\ell,j} = \|\psi_{\ell,j}\|^2$ is given by (17).

Since $V_L = V$, (19) is a representation of all functions f on $\mathcal{X}$.

4.2. Fast Multiresolution Algorithm For Wavelet Transform

The orthogonal system we discussed above allows for a fast algorithm for computing wavelet coefficients $\langle f, \psi_{\ell,j} \rangle$.

Let $f \in V_L$ be the input data given by

$$f(x) = \sum_{j \in I_L} a_{L,j}\ \phi_{L,j}(x) \tag{20}$$

with $a_{L,j} := n\langle f, \phi_{L,j} \rangle / |A_{L,j}|$ = the average value of $f(x)$ at the leave node $A_{L,j}$.

From

$$V_L = V_{L-1} \oplus^\perp W_{L-1} = \cdots = V_0 \oplus^\perp W_0 \oplus^\perp \cdots \oplus^\perp W_{L-1}$$

and that for any L_0 with $0 \le L_0 \le L-1$, $\phi_{L_0,i}, \psi_{\ell,j}, L_0 \le \ell \le L-1, i \in I_{L_0}, j \in J_\ell$ form an orthogonal basis for V_L, we know that $f \in V_L$ can also be represented as

$$\begin{aligned} f(x) &= \sum_{j \in I_{L-1}} a_{L-1,j}\phi_{L-1,j}(x) + \sum_{j \in J_{L-1}} d_{L-1,j}\psi_{L-1,j}(x) \quad (21)\\ &= \sum_{j \in I_{L-2}} a_{L-2,j}\phi_{L-2,j}(x) + \sum_{j \in J_{L-2}} d_{L-2,j}\psi_{L-2,j}(x)\\ &\quad + \sum_{j \in J_{L-1}} d_{L-1,j}\psi_{L-1,j}(x)\\ &= \cdots\\ &= a_{0,0}\phi_{0,0}(x) + \sum_{\ell=0}^{L-1}\sum_{j \in J_\ell} d_{\ell,j}\psi_{\ell,j}(x), \end{aligned}$$

where $a_{0,0} = \frac{1}{n}\sum_{x \in \mathcal{X}} f(x)$, and the wavelet coefficients $d_{\ell,j}$ are given by

$$d_{\ell,j} = \frac{1}{B_{\ell,j}} \langle f, \psi_{\ell,j} \rangle.$$

A multiscale fast algorithm to compute the wavelet coefficients can be obtained based on the refinement of the scaling function $\phi_{\ell,j}$. Next, let us look at the decomposition algorithm for calculating $a_{L-1,j}, d_{L-1,j}$ from $a_{L,j}$, and the reconstruction algorithm for recovering $a_{L,j}$ from $a_{L-1,j}, d_{L-1,j}$.

Clearly, if $k \in I_{L-1} \backslash J_{L-1}$, then $a_{L-1,k'} = a_{L-1,k}$, where $k' \in I_{L-1}$ is such an index that $A_{L-1,k'} = A_{L,k}$. Next we consider $k \in J_{L-1}$, and let $A_{L,k'}$ and $A_{L,k''}$ be two children of $A_{L-1,k}$. From (20), (21), the orthogonality of $\phi_{L-1,j}, \psi_{L-1,j}$ and the fact $\text{supp}(\phi_{L-1,k}) = A_{L-1,k} = A_{L,k'} \bigcup A_{L,k''}$, we have

$$\begin{aligned} a_{L-1,k}\|\phi_{L-1,k}\|^2 &= \Big\langle \phi_{L-1,k}, \sum_{j \in J_{L-1}} a_{L-1,j}\phi_{L-1,j} + \sum_{j \in J_{L-1}} d_{L-1,j}\psi_{L-1,j} \Big\rangle\\ &= \langle f, \phi_{L-1,k} \rangle = \Big\langle \phi_{L-1,k}, \sum_{j \in I_L} a_{L,j}\phi_{L,j} \Big\rangle\\ &= a_{L,k'}\langle \phi_{L-1,k}, \phi_{L,k'} \rangle + a_{L,k''}\langle \phi_{L-1,k}, \phi_{L,k''} \rangle\\ &= a_{L,k'}\nu(A_{L,k'}) + a_{L,k''}\nu(A_{L,k''}). \end{aligned}$$

With $\|\phi_{L-1,k}\|^2 = \nu(A_{L-1,k}) = \nu(A_{L,k'}) + \nu(A_{L,k''})$, we have

$$a_{L-1,k} = \frac{\nu(A_{L,k'})}{\nu(A_{L,k'}) + \nu(A_{L,k''})} a_{L,k'} + \frac{\nu(A_{L,k'})}{\nu(A_{L,k'}) + \nu(A_{L,k''})} a_{L,k''}.$$

Similarly, we have

$$\begin{aligned} d_{L-1,k}\|\psi_{L-1,k}\|^2 &= \langle \psi_{L-1,k}, f\rangle = \Big\langle \psi_{L-1,k}, \sum_{j\in I_L} a_{L,j}\phi_{L,j}\Big\rangle \\ &= a_{L,k'}\langle \psi_{L-1,k}, \phi_{L,k'}\rangle + a_{L,k''}\langle \psi_{L-1,k}, \phi_{L,k''}\rangle \\ &= a_{L,k'}\int_{A_{L,k'}} \psi_{L-1,k}(x)d\nu(x) + a_{L,k''}\int_{A_{L,k''}} \psi_{L-1,k}(x)d\nu(x) \\ &= a_{L,k'}\frac{\nu(A_{L,k'})\nu(A_{L,k''})}{\nu(A_{L-1,k})} - a_{L,k''}\frac{\nu(A_{L,k'})\nu(A_{L,k''})}{\nu(A_{L-1,k})} \\ &= (a_{L,k'} - a_{L,k''})\|\psi_{L-1,k}\|^2. \end{aligned}$$

Thus, we have

$$d_{L-1,k} = a_{L,k'} - a_{L,k''}.$$

Combining these calculations, we obtain the following decomposition algorithm

$$\begin{bmatrix} a_{L-1,k} \\ d_{L-1,k} \end{bmatrix} = \begin{bmatrix} \frac{\nu(A_{L,k'})}{\nu(A_{L,k'})\nu(A_{L,k''})} & \frac{\nu(A_{L,k''})}{\nu(A_{L,k'})+\nu(A_{L,k''})} \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a_{L,k'} \\ a_{L,k''} \end{bmatrix}$$

One can obtain as above by the refinement of $\phi_{L,j}$: $\phi_{L-1,j} = \phi_{L,j'} + \phi_{L,j''}$ and the orthogonality of $\phi_{L,j}$ the following reconstruction algorithm (which can also be obtained directly from the above decomposition algorithm):

$$\begin{bmatrix} a_{L,k'} \\ a_{L,k''} \end{bmatrix} = \begin{bmatrix} 1 & \frac{\nu(A_{L,k''})}{\nu(A_{L,k'})+\nu(A_{L,k''})} \\ 1 & -\frac{\nu(A_{L,k'})}{\nu(A_{L,k'})+\nu(A_{L,k''})} \end{bmatrix} \begin{bmatrix} a_{L-1,k} \\ d_{L-1,k} \end{bmatrix}$$

We can obtain the algorithms in the same way for all other $a_{\ell,k}, d_{\ell,k}$. To summarize, we have the following theorem.

Theorem 2. *Let $a_{\ell,k}, d_{\ell,k}$ be coefficients of $f \in V_L$ with the wavelet expansion. Then for every non-leave node $A_{\ell-1,k}, k \in J_{\ell-1}$, and its children nodes $A_{\ell,k'}$ and $A_{\ell,k''}$, where $1 \le \ell \le L$, we have the decomposition algorithm:*

$$\begin{bmatrix} a_{\ell-1,k} \\ d_{\ell-1,k} \end{bmatrix} = \begin{bmatrix} \frac{\nu(A_{\ell,k'})}{\nu(A_{\ell,k'})+\nu(A_{\ell,k''})} & \frac{\nu(A_{\ell,k''})}{\nu(A_{\ell,k'})+\nu(A_{\ell,k''})} \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a_{\ell,k'} \\ a_{\ell,k''} \end{bmatrix} \quad (22)$$

and the reconstruction algorithm:

$$\begin{bmatrix} a_{\ell,k'} \\ a_{\ell,k''} \end{bmatrix} = \begin{bmatrix} 1 & \frac{\nu(A_{\ell,k''})}{\nu(A_{\ell,k'})+\nu(A_{\ell,k''})} \\ 1 & -\frac{\nu(A_{\ell,k'})}{\nu(A_{\ell,k'})+\nu(A_{\ell,k''})} \end{bmatrix} \begin{bmatrix} a_{\ell-1,k} \\ d_{\ell-1,k} \end{bmatrix}. \quad (23)$$

It can be easily verified that here $a_{\ell,k}$ is exactly the mean of $f(x_i)$'s over the subset $A_{\ell,k}$ and $d_{\ell,k}$ is the difference of the means of $f(x_i)$ over the children subsets $A_{\ell+1,k'}$ and $A_{\ell+1,k''}$, respectively. This justifies (5), (7), and (8) without shrinking.

After the wavelet coefficients $d_{\ell,k}$ are thresholded with

$$\hat{d}_{\ell,k} = \mathrm{sign}(d_{\ell,k})\max\left\{0, |d_{\ell,k}| - \alpha\sqrt{\frac{1}{|A_{\ell,k'}|^2} + \frac{1}{|A_{\ell,k''}|^2}}\right\} \quad (24)$$

for some $\alpha > 0$, the estimation of $f(x)$ is obtained:

$$\hat{f}(x) = \langle f, \phi_{0,0}\rangle\phi_{0,0}(x) + \sum_{\ell=0}^{L-1}\sum_{j\in J_\ell} \hat{d}_{\ell,j}\psi_{\ell,j}(x). \quad (25)$$

A fast algorithm to evaluate the estimation $\hat{y}_j := \hat{f}(x_j)$ can be given as follows.

Set $\hat{a}_{0,0} = a_{0.0} = \langle f, \phi_{0,0}\rangle$. Assume $\hat{a}_{\ell-1,k}$ for $k \in I_{\ell-1}$ have been obtained. Define $\hat{a}_{\ell,k}$ for $k \in I_\ell$ as follows. If $k \notin J_{\ell-1}$, then the corresponding node $A_{\ell-1,k}$ is a leaf node, and let $\hat{a}_{\ell,k} = \hat{a}_{\ell-1,k'}$, where k' is the index such that $A_{\ell,k} = A_{\ell-1,k'}$. If $k \in J_{\ell-1}$, then the corresponding node $A_{\ell-1,k}$ has two children, denoted by $A_{\ell,k'}$ and $A_{\ell,k''}$, and we define

$$\begin{bmatrix} \hat{a}_{\ell,k'} \\ \hat{a}_{\ell,k''} \end{bmatrix} = \begin{bmatrix} 1 & \frac{\nu(A_{\ell,k''})}{\nu(A_{\ell,k'})+\nu(A_{\ell,k''})} \\ 1 & -\frac{\nu(A_{\ell,k'})}{\nu(A_{\ell,k'})+\nu(A_{\ell,k''})} \end{bmatrix} \begin{bmatrix} \hat{a}_{\ell-1,k} \\ \hat{d}_{\ell-1,k} \end{bmatrix}. \quad (26)$$

Theorem 3. *Let $\hat{a}_{L,j}, j \in I_L$ be the scalars defined above iteratively with $\ell = 1, 2, \cdots, L$ and let $\hat{f}(x)$ in be the estimation for $f(x)$ given in (25). Then $\hat{f}(x_j) = \hat{a}_{L,j}$. More precisely, $\hat{f}(x)$ in (25) can be represented as*

$$\hat{f}(x) = \sum_{j\in I_L} \hat{a}_{L,j}\chi_{A_{L,j}}(x). \quad (27)$$

Clearly, (26) is actually the wavelet reconstruction algorithm (23). Thus we can use a fast algorithm to evaluate $\hat{f}(x)$.

The representation (27) for $\hat{f}(x)$ defined by (25) can be proved easily by applying the following claim.

Claim 1. For $k \in J_{\ell-1}$ with $A_{\ell-1,k} = A_{\ell,k'} \cup A_{\ell,k''}$, where $1 \le \ell \le L$, we have

Proof of Calim 1: By the definitions of $\phi_{\ell-1,k}(x)$ and $\psi_{\ell-1,k}(x)$, we have

$$\begin{aligned} &\hat{a}_{\ell-1,k}\phi_{\ell-1,k}(x) + \hat{d}_{\ell-1,k}\psi_{\ell-1,k}(x) \\ &= \hat{a}_{\ell-1,k}\big(\phi_{\ell,k'}(x) + \phi_{\ell,k''}(x)\big) + \hat{d}_{\ell-1,k} \\ &\qquad \Big(\frac{\nu(A_{\ell,k''})}{\nu(A_{\ell-1,j})}\phi_{\ell,k'}(x) - \frac{\nu(A_{\ell,k'})}{\nu(A_{\ell-1,j})}\phi_{\ell,k''}(x)\Big) \\ &= \Big(\hat{a}_{\ell-1,k} + \hat{d}_{\ell-1,k}\frac{\nu(A_{\ell,k''})}{\nu(A_{\ell-1,j})}\Big)\phi_{\ell,k'}(x) \\ &\qquad + \Big(\hat{a}_{\ell-1,k} - \hat{d}_{\ell-1,k}\frac{\nu(A_{\ell,k'})}{\nu(A_{\ell-1,j})}\Big)\phi_{\ell,k''}(x) \\ &= \hat{a}_{\ell,k'}\phi_{\ell,k'}(x) + \hat{a}_{\ell,k''}\phi_{\ell,k''}(x), \end{aligned}$$

as desired. □

Observe that (26) is the algorithm (7) and (8) we use in section 2, and (27) is the representation (9) we also use in section 2 for estimation of $f(x)$.

5. DISCUSSION

The regression method discussed in this paper is based on a tree-based representation of data and a wavelet-like multiscale analysis. The tree-based representation organizes data into hierarchically related partitioning subsets of feature points together with the differences of means of the response variables over the partitioning children subsets. With this representation, a wavelet soft-thresholding and reconstruction procedure allow us to fit the data into the tree-structure. For normal data, the soft-thresholding is equivalent to shrink a t-confidence interval about the origin to the origin on the real-line.

Through the examples (section 3), we see that this tress regression method can be an effective alternative to CART, random-forest, smoothing splines, or support vector machines in various circumstances. Its ability of being adaptive to intrinsic low dimension of data allows it to detect some hidden features of data, as is shown in Example 2, when the standard methods like support vector machine fail to archive this. It outperforms another popular tree-based method, random-forest in terms of prediction error in our regression example (Example 1) in high dimensional feature space with low intrinsic dimension of data. When applied to lower dimensional data (Example 3), it again shows lower prediction error than CART or random-forest, and outperforms other smoothing method when regression function has discontinuities.

Other partitioning trees could be used in subsection 2.1. Rules for stopping further splitting a node can also be considered, provided that the local structures of the regression function in data can be optimally preserved. Another feature of this regression is that, unlike a standard wavelet analysis, the unbalanced Haar orthogonal system here is data dependent.

For large and high dimensional datasets, our method takes significantly more computation time than the other algorithms we used in our examples above. How to improve the speed of computation in our method is a challenge. One possible direction in searching for a faster algorithm is to use smaller and random subsets of the features for splitting each node in growing a tree.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

FUNDING

The work was partially supported by Simons Foundation (Grant No. 353185).

REFERENCES

1. Stone CJ. Optimal rates of convergence for nonparametric estimators. *Ann Stat.* (1980) **8**:1348–60.
2. Stone CJ. Optimal global rates of convergence for nonparametric regression. *Ann Stat.* (1982) **10**:1040–53.
3. Bickel P, Li B. Local polynomial regression on unknown manifolds. In: Liu R, Strawderman W, Zhang CH. *Complex Datasets and Inverse Problems: Tomography, Networks, and Beyond.* IMS Lecture Notes Monograph Series. Vol. 54. Bethesda, MD: Institute of Mathematical Statistics (2007). p. 177–86.
4. Binev P, Cohen A, Dahmen W, DeVore R, Temlyakov V. Universal algorithms for learning theory part I: piecewise constant functions. *J Mach Learn Res.* (2005) **6**:1297–321.
5. Binev P, Cohen A, Dahmen W, DeVore R. Universal algorithms for learning theory part II: piecewise polynomial functions. *Construct Approx.* (2007) **26**:127–52. doi: 10.1007/s00365-006-0658-z
6. Dasgupta S, Freund Y. Random projection trees and low dimensional manifolds. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. New York, NY (2008) p. 537–46.
7. Kpotufe S, Dasgupta S. A tree-based regressor that adapts to intrinsic dimension. *J Comput Syst Sci.* (2012) **78**:1496–515. doi: 10.1016/j.jcss.2012.01.002
8. Breiman L. Random forests. *Mach Learn.* (2001) **45**:5–32. doi: 10.1023/A:1010933404324
9. Breiman L, Friedman JH, Olshen RA, Stone CJ. *Classification and Regression Trees.* Belmont, CA: Wadsworth (1984).
10. Loh WY. Fifty years of classification and regression trees. *Int Stat Rev.* (2014) **82**:329–48. doi: 10.1111/insr.12016
11. Morgan JN, Sonquist JA. Problems in the analysis of survey data, and a proposal. *J Am Stat Assoc.* (1963) **58**: 415–34.
12. Wasserman L. *All of Nonparametric Statistics* Springer (2006).
13. Chui CK, Filbir F, Mhaskar HN. Representation of functions on big data: graphs and trees. *Appl Comput Harmon Anal.* (2015) **38**:489–509. doi: 10.1016/j.acha.2014.06.006
14. Gavish, M., Nadler B, Coifman RR. Multiscale wavelets on trees, graphs and high dimensional data: theory and applications to semi supervised learning. In: *Proceedings of the 27th International Conference on Machine Learning*. Haifa (2010). p. 367–74.
15. Girardi M, Sweldens W. A new class of unbalanced Haar wavelets that form an unconditional basis for L_p on general measure spaces. *J Fourier Anal Appl.* (1997) **3**:457–74.
16. Mitrea M. Singular integrals, hardy spaces and cliord wavelets. *Lecture Notes in Mathematics*. Vol. 1575. Berlin; Heidelberg: Springer-Verlag (1994). doi: 10.1007/BFb0073556

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.