Wavelet Decomposition of Gradient Boosting

May 8, 2018

Shai Dekel, Oren Elisha, and Ohad Morgan

Abstract

In this paper we introduce a significant improvement to the popular tree-based Stochastic Gradient Boosting algorithm using a wavelet decomposition of the trees. This approach is based on harmonic analysis and approximation theoretical elements, and as we show through extensive experimentation, our wavelet based method generally outperforms existing methods, particularly in difficult scenarios of class unbalance and mislabeling in the training data.

1 Introduction

In the setting of regression and classification tasks on structured data, decision tree ensembles are extremely useful as off-the-shelf tools [18], being relatively fast to construct, adaptive, and when trees are small, they also produce interpretable models. Tree-based boosting methods [18, 10, 20] are popular ensemble methods that are constructed by a sequential generation of pruned decision trees. These 'weak learners' are then combined to form a 'strong' estimator. In this work we focus on improving the well-known Gradient Boosting (GB) technique [10, 11], which computes a sequence of trees, that are trained to predict the residual between the response variable and the negative gradient direction of the previous tree. These residuals are used as the response variables for the next tree in the iterative processes so that the sum of the weighted trees in the ensemble is the final estimator.

Wavelets [5, 16] are a powerful, yet simple, tool for sparse representations of 'complex' functions. In [8], a wavelet-based GB method was introduced, using a classic wavelet decomposition of the original predictors, followed by a componentwise linear least squares GB mechanism. Our wavelets approach is very different. We use the theoretical foundation developed in [9] and apply a wavelet decomposition of the decision trees formed during the boosting process. This mathematical model allows us to design a more robust pruning algorithm by re-ordering of the nodes of the trees based on their significance. As we will show in the experimental part, the adaptive wavelet pruning approach generally outperforms existing methods, particularly in difficult scenarios of class unbalance

and mislabeling in the training data. In addition, we also demonstrate how to employ our method on the stochastic GB version, which uses bagging to gain diversity and use the Out Of Bag (OOB) samples to improve generalization.

The rest of paper is organized as follows. In section 2, we overview Gradient Boosting (GB) algorithms and in particular tree-based GB algorithms. In addition, we describe the stochastic GB algorithm which combines a bagging procedure into the GB. In section 3, we present the "Geometric Wavelets" (GW) decomposition, and highlight some theoretical and practical properties that emphasis their correspondence with sparsity. In Section 4 we present the Geometric Wavelets Gradient Boosting (GWGB) algorithm which combines stochastic GB with the GW decomposition. In section 5, we conclude with experiment results that compare our algorithm with competing boosting algorithms in different challenging settings.

2 Gradient boosting trees

2.1 Decision trees

In the setting of statistics and machine learning [18, 4] the construction we present in this chapter is referred as Decision Tree or the Classification and Regression Tree (CART).

When we are given with a real-valued function or a discrete data-set

$$\{x_i \in \Omega_0, y_i = f(x_i)\}_{i=1}^{m}\,, \tag{1}$$

in some convex bounded domain $\Omega_0 \subset \mathbb{R}^n$, our goal is to find an efficient representation of this data $\hat{f}(x)$, overcoming the complexity, geometry and possibly non-smooth nature of the function values. The efficiency of $\hat{f}(x)$ is typically estimated by minimization of a loss function L, with respect to the data (1), and is usually combined with additional regularization condition that aims to reduce the generalization error. For example as seen in (5), a sparsity condition is applied to reduce overfitting artifacts.

The decision tree's first level is formed by a partition of the initial domain Ω_0, into two sub-domains, e.g. by intersecting it with a hyper-plane, so as to minimize a given cost function. This subdivision process then continues recursively on the nested sub-domains until some exit criterion is met, which in turn, determines the leaves of the tree. We now describe one instance of the cost function. At each stage of the subdivision process, at a certain node of the tree, the algorithm finds, for the convex domain $\Omega \subset \mathbb{R}^n$ associated with the node, a partition by an hyper-plane into two convex sub-domains Ω', Ω'', and two multivariate low-order polynomials $Q_{\Omega'}, Q_{\Omega''}$, of fixed (typically low) total degree $r-1$, that minimize the following quantity

$$\|f - Q_{\Omega'}\|^p_{L_{p(\Omega')}} + \|f - Q_{\Omega''}\|^p_{L_{p(\Omega'')}}\,,\ \Omega' \cup \Omega'' = \Omega. \tag{2}$$

If the data-set is discrete, consisting of feature vectors $x_i \in \mathbb{R}^n$, with response

values $f(x_i)$, then a discrete functional is minimized

$$\sum_{x_i \in \Omega'} |f(x_i) - Q_{\Omega'}(x_i)|^p + \sum_{x_i \in \Omega''} |f(x_i) - Q_{\Omega''}(x_i)|^p \tag{3}$$

Observe that for any given subdividing hyperplane, the approximating polynomials in (2) can be uniquely determined for $p = 2$, by least square minimization.

For $r = 1$, the approximating polynomials are nothing but the mean of the function values over each of the sub-domains

$$Q_{\Omega'} = c_{\Omega'} = \frac{1}{\#\{x_i \in \Omega'\}} \sum_{x_i \in \Omega'} f(x_i),$$
$$Q_{\Omega''} = c_{\Omega''} = \frac{1}{\#\{x_i \in \Omega''\}} \sum_{x_i \in \Omega''} f(x_i). \tag{4}$$

Denoting by Ω_t^j a node on level j of the tree with counting index t. It is easy to see that for each fixed level J, $\Omega_0 = \bigcup_{t=1}^{2^J} \Omega_t^J$. Therefore, we can describe the tree evaluated at any fixed level J by $T^J(x) = \sum_{t=1}^{2^J} Q_{\Omega_t^J}(x) 1_{\Omega_t^J}(x)$, or simply by $T(x)$, when evaluation is done on the terminal nodes or to a predefined level J. Here, $1_\Omega(x) = 1$, if $x \in \Omega$ and $1_\Omega(x) = 0$, if $x \notin \Omega$.

In classification problems, the input training set consists of labeled data using P classes instead of function values. In this scenario, each input training point $x_i \in \mathbb{R}^n$ is assigned with a class $C(x_i)$. To convert the problem to the same 'functional' setting described above one assigns to each class C the value of a node on the regular simplex consisting of P vertices in $\mathbb{R}^{P-1}$ (all with equal pairwise distances). Thus, we may assume that the input data is in the form $\{x_i, y_i\}_{i=1}^m \in \left(\mathbb{R}^n, \mathbb{R}^{P-1}\right)$. In this case, if we choose approximation using constants $(r = 1)$, then the calculated mean over any sub-domain Ω is in fact a point $\overrightarrow{E}_\Omega \in \mathbb{R}^{P-1}$, inside the simplex. Obviously, any value inside the multidimensional simplex, can be mapped back to a class, along with an estimated certainty confidence level, by calculating the closest vertex of the simplex to it. As will become obvious, these mappings can be applied to any wavelet approximation of functions receiving multidimensional values in the simplex.

In many algorithms that are based on decision trees, the high-dimensionality of the data does not allow to search through all possible subdivisions. As in our experimental results, one may restrict the subdivisions to the class of hyperplanes aligned with the main axes. In contrast, there are cases where one would like to consider more advanced form of subdivisions, where they take certain hyper-surface form, such as conic-sections. Our paradigm of wavelet decomposition can support in principle all of these forms.

2.2 Pruning decision trees

In many cases, the response variable $f(x_i)$ in (1) is obtained with noise of different types. Thus, bias-variance consideration (e.g. avoiding over fit) [18]

encourages pruning techniques that restrict the size of a tree in varies ways. Pre-pruning [15] involves a "termination condition" to determine when it is desirable to terminate some of the branches prematurely when a decision tree is generated. For example, in [3] a minimal node size is used as an exit criterion for tree generation. On the other hand, post-pruning [15], may be applied to remove some of the branches after tree is generated and could be evaluated. For example, in the CART algorithm [4] after a tree model had been generated, one applies a regularization condition with a factor γ, that penalize adding more nodes, by minimizing

$$\left\| \sum_{i=1}^{m} \left(y_i - T(x_i)\right)^2 \right\|_{l_2} + \gamma \# \left\{ \Omega_t^j \in T(x),\, j = 1, ..J \right\}. \tag{5}$$

As will be described in the next sections, while most Boosting algorithms [18] set a fixed level J as a pruning strategy, our GWGB method uses a different pruning approach that is based on the data encapsulated in each node rather than the level of the tree.

2.3 Gradient boosting

Gradient Boosting (GB) [10, 17, 18] is based on computing a sequence of weak learners such as pruned decision trees, using a gradient descent iterative method. A functional gradient view of boosting was first presented in [17]. This led to the development of boosting algorithms in many areas of machine learning and statistics, beyond regression and classification.

Let $X \in \mathbb{R}^n$ denote a real value random input vector, and $Y \in \mathbb{R}^{P-1}$ a real value random output vector, with $Pr(X, Y)$ their joint distribution. in such case we have a typically unknown function $f(X) = E[Y|X]$, so we may seek for an approximation function $\hat{f} : \mathbb{R}^n \to \mathbb{R}^{P-1}$ based on the training samples (1). Ideally, the approximation $\hat{f}(x)$ should be the one that minimizes the expected prediction error with respect to some specified loss function $L(Y, f(X))$,

$$\hat{f}(X) = \underset{f}{argmin} E_{X,Y} L(Y, f(X)). \tag{6}$$

Frequently employed loss functions L include squared-error $(y - f(x))^2$ and absolute error $|y - f(x)|$ for $y \in \mathbb{R}$ (regression), and negative binomial log-likelihood $log\left(1 + e^{-2yf(x)}\right)$, when $y \in \{-1, 1\}$ (binary classification). Here, we present a unified approach for regression and classification and choose the squared-error, as described in (2) and 3.

In the setting of tree-based GB algorithm, the approximation function $\hat{f}(x)$ is a combination of $K + 1$ weak learners

$$\hat{f}_K(x) := \hat{f}_0(x) + \nu \sum_{k=1}^{K} T_k(x), \tag{7}$$

where K is the number of boosting iterations, $T_k(x)$ are the pruned trees, and ν is the step size (also called Learning Rate or Shrinkage).

The function $\hat{f}_0(x)$ is typically an initial estimate such as

$$\hat{f}_0(x) = \underset{c}{argmin} \sum_{i=1}^{N} L\left(y_i, c\right).$$

To generate the weak learners, one typically constructs K decision trees $\{T_k\}_{k=1}^{K}$, while applying a fixed tree level J, in the following way. In each iteration k, a decision tree T_k is built, so residuals could be set as the response variables for the next $k+1$ step

$$\left\{y_i^{k+1} = \hat{f}_k(x_i) - y_i^k\right\}_{i=1}^{m} \text{ with } \left\{y_i^0 = y_i\right\}_{i=1}^{m}.$$

This iterative procedure resembles a gradient decent in the sense that at each iteration we set the next step in the opposite direction of the pseudo gradient of the loss function L.

As described in [18], it is common practice to set $4 \leq J \leq 8$, to have good results in the context of boosting. As we shall see, our approach is to choose a higher level J and then apply the wavelet-based approach of pruning specific nodes. The selection of K should set a balance between reducing the training error and avoiding overfitting when $K \to \infty$. Thus, [19] uses a validation samples or an OOB samples (when applying the stochastic version of GB), to find the optimal K.

A modification to tree-based GB algorithm, called Stochastic GB (SGB), was proposed in [11] by applying a bagging step at each iteration, that uses only a random subsample of the training data. This randomly selected subsample is then used, instead of the full sample, to fit the regression tree. The OOB samples are then used for validation. In our algorithm, as well as previous algorithms, the OOB are used to determine the pruning, not only validation.

3 Geometric wavelets

The Geometric Wavelet (GW) decomposition of decision trees were presented at [6], based on the theory of [13, 14]. It was recently generalized to a decomposition of Random Forests [9] and used to enhance their performance as well as introduce a novel algorithm for feature importance.

Let Ω' be a child of Ω in a decision tree T, i.e. $\Omega' \subset \Omega$ and Ω' has been created by a partition of Ω, and let two polynomials $Q_{\Omega'}$,$Q_{\Omega''}$ that minimize the quantity (2). We use the polynomial approximations $Q_{\Omega'}$,$Q_{\Omega} \in \Pi_{r-1}\left(\mathbb{R}^n\right)$ and define

$$\psi_{\Omega'} := \psi_{\Omega'}(f) := 1_{\Omega'}(Q_{\Omega'} - Q_{\Omega}) \tag{8}$$

as the **geometric wavelet** associated with the sub-domain Ω' and the function f, or the given discrete data-set (1).

Each wavelet $\psi_{\Omega'}$ is a 'local difference' component that belongs to the detail space between two levels in the tree, a 'low resolution' level associated with Ω and a 'high resolution' level associated with Ω'. Also, the wavelets (8) have the 'zero moments' property, i.e., if the response variable is sampled from a polynomial of degree $r-1$ over Ω, then our local scheme will compute $Q_{\Omega'} = Q_{\Omega} = f(x)$, $\forall x \in \Omega'$ and therefore $\psi_{\Omega'} = 0$.

Under certain mild conditions on the tree T and the function f, we have by the nature of the wavelets, the 'telescopic' sum of differences:

$$f = \sum_{\Omega \in T} \psi_{\Omega}, \qquad \text{where } \psi_{\Omega_0} := Q_{\Omega_0}. \tag{9}$$

For example, (9) holds in L_p-sense ($1 \leq p < \infty$), if $f \in L_p(\Omega_0)$, and for any $x \in \Omega_0$ and series of domains $\Omega_j \in T$, each on a level j with $x \in \Omega_j$, we have that $\lim_{j \to \infty} diam(\Omega_j) = 0$ (see Theorem 2.1 in [6]).

In the theoretical setting, the norm of a wavelet is computed by

$$\|\psi_{\Omega'}\|_2^2 = \int_{\Omega'} \left(Q_{\Omega'}(x) - Q_{\Omega}(x)\right)^2 dx, \tag{10}$$

and in the discrete case by

$$\|\psi_{\Omega'}\|_2^2 = \sum_{x_i \in \Omega'} \left|Q_{\Omega'}(x_i) - Q_{\Omega}(x_i)\right|^2, \tag{11}$$

where Ω' is a child of Ω. This wavelet norm tell us how much information this wavelet encapsulate (see [6],[9]).

Recall that our approach converts classification problems into a 'functional' setting by assigning the P class labels to vertices of a simplex in $\mathbb{R}^{P-1}$ (see discussion in 2.1) . In such cases of multi-valued functions, choosing $r = 1$, the wavelet $\psi_{\Omega'} : \mathbb{R}^n \to \mathbb{R}^{P-1}$ is

$$\psi_{\Omega'} = 1_{\Omega'} \left(\overrightarrow{E}_{\Omega'} - \overrightarrow{E}_{\Omega}\right),$$

and its norm is given by

$$\|\psi_{\Omega'}\|_2^2 = \left\|\overrightarrow{E}_{\Omega'} - \overrightarrow{E}_{\Omega}\right\|_{l_2}^2 \# \left\{x_i \in \Omega'\right\},$$

where for $\overrightarrow{v} \in \mathbb{R}^{P-1}$, $\|\overrightarrow{v}\| := \sqrt{\Sigma_{i=1}^{P-1} v_i^2}$. Accordingly, we can consider the squared-error as loss function for classification problems.

It is easy to see that the decision tree T can be writen as

$$T(x) = \sum_{\Omega_j \in T} \psi_{\Omega_j}(x). \tag{12}$$

The theory (see Theorem 4 in [6]) tells us that sparse approximation is achieved by ordering the wavelet components based on their norm

$$\left\|\psi_{\Omega_{k_1}}\right\|_2 \geq \left\|\psi_{\Omega_{k_2}}\right\|_2 \geq \left\|\psi_{\Omega_{k_3}}\right\|_2 ... \tag{13}$$

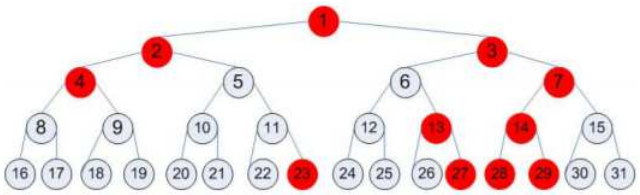

Figure 1: Illustration of greedy node selection by wavelet norms

Thus, the adaptive M-term approximation of a decision tree T is

$$T_M(x) := \sum_{j=1}^{M} \psi_{\Omega_{k_j}}. \tag{14}$$

This pruning method is, in some sense, a generalization of the classical M-term wavelet sum, where the wavelets are constructed over dyadic cubes (see [7]).

4 Wavelet decomposition of gradient boosting

In this section, we introduce a combination of Stochastic GB Tree algorithm and Geometric Wavelet decomposition. In our setting, instead of the pruned decision tree T at some fixed level J as weak learner, we retrieve the M "most important" nodes, in term of wavelet-norm (see (13) and (14)), which is the M-term approximation T_M. This M-term approximation is used as the weak learner at each boosting iteration. To select M at each iteration, we use the OOB data as in 2 and add the wavelets, one by one according to its wavelets norm until error of the model is minimized on the OOB set. Our implementation for "Geometric Wavelets Gradient Boosting" described in Algorithm 1. .

One of the advantages of this form of tree pruning is the fact that nodes are selected according to their contribution to the prediction (9), rather than their position at a certain level in the tree. This allows an adaptive selection of high and low resolution at the same step of the boosting. An illustration of M-term GW collection whose graph representation includes some unconnected components is shown in Figure 1. The M-term nodes are marked in red, while the rest of the nodes in the tree are not used for the estimation.

Another advantage of relaying on the wavelets norms for pruning in the GBM setting, is an efficient feature selection for the ensemble. In some cases, explanatory attributes may be non-descriptive and even noisy, leading to the creation of problematic nodes in the decision trees. Nevertheless, in these cases, the corresponding wavelet norms are controlled and these nodes can be omitted from the representation (14). An example that demonstrates this phenomen is presented in [9] (see Example 1). The example shows that with high probability, the wavelets associated with the correct variables have relatively higher norms than wavelets associated with non-descriptive variables. Hence the wavelet based criterion will choose, with high probability the correct variable. Since the tree

Algorithm 1 Geometric Wavelets Gradient Boosting

1. Initialize $\hat{f}_0(x) = \underset{c}{argmin} \sum_{i=1}^{m} L(y_i, c)$.
2. set $\left\{y_i^0 = y_i\right\}_{i=1}^{m}$.
3. For $k = 1, 2, ..., K$
   (a) update the residuals $\left\{y_i^k = \hat{f}_{k-1}(x_i) - y_i^{k-1}\right\}_{i=1}^{m}$.
   (b) Choose randomly subset of m' variables from the original data-set, denote by $\left\{x_i, y_i^k\right\}_{i=1}^{m'}$. based on this training set generate a tree $T(x) = \sum_j \psi_{\Omega_{k_j}}$, where $\left\{\psi_{\Omega_{k_j}}\right\}_j$ are the GW sorted by wavelet norm (see 13).
   (c) Denote the OOB subset by $OOB = \left\{(x,y) \mid (x,y) \notin \{x_i, y_i\}_{i=1}^{m'}\right\}$, then compute:

$$M_k = \underset{M}{argmin} \sum_{(x,y)\in OOB} L\left(y^k, \sum_{j=1}^{M} \psi_{\Omega_{k_j}}(x)\right).$$

   (d) Update the prediction model: $\hat{f}_k(x) = \hat{f}_{k-1}(x) + \nu \sum_{j=1}^{M_k} \psi_{\Omega_{k_j}}(x)$.
4. Output $\hat{f}(x) = \hat{f}_K(x)$.

partitions are based on (2) and (13), the non-descriptive variables are less likely to form partitions that are part of the GWGB ensemble.

5 Experimental results

In this section we compare the Geometric Wavelets GB algorithm (GWGB, algorithm 1) with other boosting and bagging methods in terms of classifying imbalance datasets, improving regression tasks and overcoming mislabeling noise in classification tasks.

Our GWGB code was written in C# and is publicly available [1]. At each iteration we use the OOB technique as described in section 2, with 80% of the training set to build the wavelets tree, and 20% for M-term selection. Moreover, a fixed step size (ν) of 0.1 is used throughout all of our experiments.

[1] https://github.com/ohadmorgan/GeometricWaveletGradeintBoosting.git

5.1 Classification with imbalance class distributions

Classification problem with data-sets that suffer from imbalanced class distributions is a challenging problem in the field of machine learning.

We present a comparison of our algorithm performance, with state of the art ensemble-based techniques for imbalanced data-sets presented in [12]. The experiment is based on the testing a variety of methodologies on 44 real-world imbalanced problems from KEEL data-set repository [1]. We use the same 5-fold cross-validation data and partitions that are provided in [12] to measure the AUC (Area Under the Curve) metric. Moreover, at each iteration we grow the tree to a fixed level of depth level 8, and selected M_k terms according to our algorithm, while using the same $K = 10$ which was used in [12]. Comparison results, including GWGB algorithm, are presented in table 1. Since the authors of [12] reviewed 37 different bagging, boosting and classics algorithms, for brevity and space limitation, we present a comparison of our method to the best algorithm in each category (in term of mean AUC) for each data set.The success of our method is due to the fact that we could build deeper trees and reach high resolution areas where the rare categories might accord, and select these nodes in early stages of the ensemble (lower K). This is the advantage of wavelets reordering according to thier norm which enables a pruning strategy that is not dependent on the tree's depth or level.

5.2 Regression

In this section we compare our method with the most recent Boosting schemes presented at [20]. We follow the same randomization process presented by [20] with 20 random trails of 2-fold cross validation, and we have followed the same technique for adaptive selection of the number of iterations as in [20] as we chose the best $k \in [0, 500]$ in term of RMSE on validation. As in the previous section, at each iteration we have grow the tree to a fixed level of depth level 8, and select M_k terms according to our algorithm. The results are presented in Table 2 are the average RMSE and standard deviation of the 20 random trails and are compared to the three best algorithms from [20].

5.3 Overcoming mislabeling noise in classification

Boosting methods are known to be sensitive to label noise [2]. The experiment is based on the same testing methodology presented in [2], with the injection of two noise levels (NL) of random 10% and 30% of the original labels in the datasets. Average and standard deviation of the misclassification rates computed from 10-fold cross validation. As in [2] we have restricted the number of iteration to $K = 150$, and restricted the level to2.

The first method (rAdaBoost) is modification of the AdaBoost algorithm, using "robust classifiers" that combined and boosted using known AdaBoost algorithm. The two next methods (rBoost-Fixed gamma and rBoost) are new robust boosting algorithms where the objective function is a convex combination

Table 1: Class Imbalance results comparison (AUC)

Dataset name	**Best Bagging-based method** UB4	**Best Boosting-based method** RUS 1	**Best Classic method** SMT	**Geometric Wavelets** GWGB
glass1	0.737	0.763	0.737	**0.816**
ecoli0vsl	0.980	0.969	0.973	**0.986**
Wisconsin	0.960	0.964	0.953	**0.985**
Pima	0.760	0.726	0.725	**0.809**
Iris0	0.990	0.990	0.990	**1.000**
glass0	0.814	0.813	0.775	**0.880**
yeast1	0.722	0.719	0.709	**0.775**
vehicle1	0.787	0.747	0.730	**0.810**
vehicle2	0.964	0.970	0.950	**0.982**
vehicle3	0.802	0.765	0.728	**0.805**
Haberman	**0.664**	0.655	0.616	0.651
glass0123vs456	0.904	0.930	0.923	**0.960**
vehicle0	0.952	0.958	0.919	**0.982**
ecoli1	0.900	0.883	0.911	**0.951**
new-thyroid2	0.958	0.938	0.966	**0.996**
new-thyroid1	0.964	0.958	0.963	**0.993**
ecoli2	0.884	0.899	0.811	**0.918**
Segimmt0	0.988	**0.993**	**0.993**	0.987
glass6	0.904	0.918	0.884	**0.935**
yeast3	0.934	0.925	0.891	**0.957**
ecoli3	0.908	0.856	0.812	**0.923**
Page-blocks0	0.958	0.948	0.950	**0.990**
yeast2vs4	0.936	0.933	0.859	**0.981**
yeast05679vs4	0.794	0.803	0.760	**0.863**
vowel0	0.947	0.943	0.951	**0.988**
glass016vs2	**0.754**	0.617	0.606	0.720
glass2	0.769	**0.780**	0.639	0.690
ecoli4	0.888	**0.942**	0.779	0.906
suttle0vs4	**1.000**	**1.000**	**1.000**	**1.000**
yrast1vs7	**0.786**	0.715	0.700	0.760
glass4	0.846	0.915	0.887	**0.963**
page-blocks13vs4	0.978	0.987	**0.996**	0.992
abalone9vs18	0.719	0.693	0.628	**0.827**
glass016vs5	0.943	**0.989**	0.813	0.946
suttle2vs4	**1.000**	**1.000**	0.992	0.994
yrast1458vs7	**0.606**	0.567	0.537	0.594
glass5	0.949	0.943	0.881	**0.982**
yeast2vs8	0.783	0.789	**0.834**	0.616
yeast4	0.855	0.812	0.712	**0.865**
yeast1289vs7	0.734	0.721	0.683	**0.765**
yeast5	0.952	0.959	0.934	**0.968**
ecoli0137vs26	0.745	0.794	**0.814**	**0.814**
yeast6	0.869	0.823	0.829	**0.876**
Abalone19	**0.721**	0.631	0.521	0.594

Table 2: Regression results comparison

Dataset	Decision stumps			Vanilla neural networks			**GWGB**
name	**R-Boosting**	**ϵ-Boosting**	**RT-Boosting**	**R-Boosting**	**ϵ-Boosting**	**RT-Boosting**	
Diabetes	58.71±1.2	58.94±1.9	58.61±2.7	58.03±2.3	58.03±2.4	58.03±2.4	**57.01±3.0**
Housing	4.13±0.3	4.33±0.2	4.14±0.4	4.02±0.3	4.45±0.3	4.45±0.3	**3.38±0.4**
CCS	5.47±0.1	6.10±0.7	5.35±0.3	6.59±0.4	6.62±0.2	6.52±0.3	**4.80±0.4**
Abalone	2.28±0.02	2.40±0.05	2.28±0.05	2.12±0.05	**2.10±0.03**	2.13±0.02	2.17±0.06

Table 3: Misclassification results comparison

Dataset name	**NL**	**r-AdaBoost**	**rBoost-Fixed γ**	**rBoost**	**GBoost**	**MBoost**	**GWGB**
Banana	**0.1**	86.87±1.1	87.06±0.9	87.04±0.9	83.91±1.6	78.13±3.4	**87.60±1.6**
	0.3	85.27±3.0	**85.53±2.1**	85.06±2.7	79.38±1.6	75.31±2.5	85.49±1.3
PID	**0.1**	74.20±2.3	74.37±1.5	74.80±2.4	72.60±2.0	**75.67±1.9**	74.21±6.2
	0.3	72.53±1.9	70.43±2.4	71.43±2.3	69.40±2.9	73.33±2.3	**75.65±7.5**
Heart	**0.1**	78.40±3.1	79.70±3.5	79.10±4.4	76.40±3.1	77.60±3.5	**80.74±8.2**
	0.3	**78.50±4.0**	77.40±6.5	78.10±4.3	70.00±5.5	75.20±3.7	73.70±11.5
Two-Norm	**0.1**	95.70±0.8	95.58±0.9	95.59±0.7	90.35±1.0	92.79±0.5	**96.40±0.8**
	0.3	93.33±0.9	93.13±1.3	93.40±1.1	83.94±2.0	91.16±0.9	**94.82±0.7**

of two exponential losses. The results for the well-known Gentle Boost (GBoost), and Modest Boost (MBoost) are also taken from [2] and presented in Table 3. The reason for improved results of our method in the case of mislabeling noise is due to removing wavelets with small GW norm. As seen in 11 the magnitude of a wavelet norm that corresponds to a single point (typical miss labeled point) is small and hence will be typically pruned while keeping informative nodes on the same level of the tree.

References

[1] J Alcalá, A Fernández, J Luengo, J Derrac, S García, L Sánchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2010.

[2] Jakramate Bootkrajang and Ata Kabán. Boosting in the presence of label noise. *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI2013)*, 2013.

[3] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[4] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[5] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[6] S. Dekel and D. Leviatan. Adaptive multivariate approximation using binary space partitions and geometric wavelets. *SIAM Journal on Numerical Analysis, To Appear*, 43:707–732, 2005.

[7] Ronald A. DeVore. Nonlinear approximation. *ACTA NUMERICA*, 7:51–150, 1998.

[8] Eugene Dubossarsky, Jerome H Friedman, John T Ormerod, and Matthew P Wand. Wavelet-based gradient boosting. *Statistics and Computing*, 26(1-2):93–105, 2016.

[9] Oren Elisha and Shai Dekel. Wavelet decompositions of random forests - smoothness analysis, sparse approximation and applications. *Journal of Machine Learning Research*, 17(198):1–38, 2016.

[10] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[11] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February 2002.

[12] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[13] Borislav Karaivanov and Pencho Petrushev. Nonlinear piecewise polynomial approximation beyond besov spaces. *Appl. Comput. Harmonic Anal*, pages 177–223, 2003.

[14] Borislav Karaivanov, Pencho Petrushev, Robert, and C. Sharpley. Algorithms for nonlinear piecewise polynomial approximation: Theoretical aspects. *Trans., Amer. Math. Soc*, 355:2585–2631, 2002.

[15] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, pages 1–23, 2013.

[16] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[17] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent in function space, 1999.

[18] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning, second edition*. springer, 2009.

[19] Greg Ridgeway with contributions from others. *Package gbm*, 2.1.1 edition, 03 2015.

[20] Lin Xu, Shaobo Lin, Yao Wang, and Zongben Xu. Shrinkage degree in l2-rescale boosting for regression. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

Shai Dekel, is with WIX AI and the School of Mathematics, Tel-Aviv University, Tel-Aviv. *shaidekel6@gmail.com*

Oren Elisha, is with Microsoft Israel and the School of Mathematics, Tel-Aviv University, Tel-Aviv. *orenelis@gmail.com*

Ohad Morgan, is with the School of Mathematics, Tel-Aviv University, Tel-Aviv. *ohadmorgan1989@gmail.com*