Gradient Boosting Machine with Partially Randomized Decision Trees

Andrei Konstantinov, Lev Utkin, Vladimir Muliukha
Peter the Great St.Petersburg Polytechnic University
St.Petersburg, Russia
andrue.konst@gmail.com, lev.utkin@gmail.com, mulyukha_va@almazovcentre.ru

***Abstract*—The gradient boosting machine is a powerful ensemble-based machine learning method for solving regression problems. However, one of the difficulties of its using is a possible discontinuity of the regression function, which arises when regions of training data are not densely covered by training points. In order to overcome this difficulty and to reduce the computational complexity of the gradient boosting machine, we propose to apply the partially randomized trees which can be regarded as a special case of the extremely randomized trees applied to the gradient boosting. The gradient boosting machine with the partially randomized trees is illustrated by means of many numerical examples using synthetic and real data.**

I. Introduction

Ensemble-based techniques can be regarded as one of the most efficient ways to improve machine learning models. The basic idea behind ensemble-based techniques is to combine the base or weak classifiers or regressors [1], [2], [3], [4], [5], [6], [7]. The most well-known and efficient ensemble-based techniques are random forests [8] as a combination of the bagging [9] and the random subspace [10] methods, Adaboost [11], the gradient boosting machines (GBMs) [12], [13] and its modifications XGBoost [14], LightGBM [15], CatBoost [16].

GBMs have illustrated their efficiency for solving regression problems. According to the technique, the first iteration starts from the guessed prediction, then residuals are calculated as differences between guessed predictions and target variables. The residuals instead of target variables form a new dataset such that the next base model is built on the dataset. A regression tree is often used to predict new residuals. The GBM iteratively computes the sum of all previous regression tree predictions and updating residuals to reflect changes in the model. As a result, a set of regression trees is built in the GBM such that each successive tree predicts the residuals of the preceding trees given an arbitrary differentiable loss function [5]. An interesting modification of the GBM on the basis of the so-called deep forests [17] is the multi-layered gradient boosting decision tree model [18]. Another modification is the soft GBM [19].

In order to improve ensemble-based models using decision trees, Geurts et al. [20] proposed to apply the individual extremely randomized trees which aim to generate a decision forest while injecting randomness. According to the extremely randomized trees, the best splitting feature is selected from a random subset of features. Cut-points splitting features are also randomly selected during training the trees. The possible high bias and variance of the trees can be canceled by fusing a sufficiently large forest of trees [20].

Various types of randomized trees have been developed and used in ensemble-based models. A similar idea has been implemented for a randomized C4.5 decision tree [21]. Instead of selecting the best attribute at each stage, it selects an attribute from the set of the best m features randomly with equal probability. Cutler and Zhao [22] proposed the PERT (Perfect Ensemble Random Trees) approach for building perfect-fit classification trees with random split selection. Every base model in the PERT ensemble randomly chooses both the feature on which to split and the split itself.

It should be noted that one of the drawbacks of the GBM is its computational complexity. Therefore, there are many attempts to reduce it by introducing various randomization schemes. For example, Lu and Mazumder [23] proposed the Randomized GBM which leads to significant computational gains compared to the GBM by using a randomization scheme to reduce the search in the space of weak learners. Lu et al. [24] proposed the Accelerated GBM by incorporating Nesterov's acceleration techniques into the design of the GBM.

Another difficulty of using the GBM is a possible discontinuity of the regression function, which arises when some regions of training data are not densely covered by points, for example, when the training set is very small. We illustrate by examples, that the standard GBM as well as the extremely randomized trees may lead to incorrect prediction and the discontinuity of the regression function.

In order to overcome the problems of the computational complexity as well as the discontinuity of the regression function, we propose the GBM with partially randomized trees as base learners. In contrast to extremely randomized trees, the cut-point for partitioning each feature in these trees is determined randomly from a uniform distribution, but the best feature is selected such that it maximizes the score. On the one hand, the partially randomized trees can be regarded as a special case of extremely randomized trees. On the other hand, their incorporation into the GBM differ them from the extremely randomized trees. Many numerical examples show that the GBM with partially randomized trees provide better prediction results in comparison with many ensemble-based models, including extremely randomized trees. Moreover, the corresponding GBMs are computationally simpler in comparison with the original GBM.

II. Regression problem statement

The standard regression problem can be stated as follows. Given N training data (examples, instances, patterns)

$D = \{(x_1, y_1), ..., (x_N, y_N)\}$, in which x_i belongs to a set $\mathcal{X} \subset \mathbb{R}^m$ and represents a feature vector involving m features, and $y_i \in \mathbb{R}$ represents the observed output or the target value such that $y_i = f(x_i) + \varepsilon$. Here ε is the random noise with expectation 0 and unknown finite variance. Machine learning aims to construct a regression model or an approximation g of the function f that minimizes the expected risk or the expected loss function

$$\begin{aligned} L(f) &= \mathbb{E}_{(x,y)\sim P}\ l(y, g(x)) \\ &= \int_{\mathcal{X}\times\mathbb{R}} l(y, g(x)) \mathrm{d}P(x, y), \end{aligned} \tag{1}$$

with respect to the function parameters. Here $P(x, y)$ is a joint probability distribution of x and y; the loss function $l(\cdot, \cdot)$ may be represented, for example, as follows:

$$l(y, g(x)) = (y - g(x))^2 . \tag{2}$$

There are many powerful machine learning methods for solving the regression problem, including regression random forests [25], [8], the support vector regression [26], etc. One of the powerful methods is the GBM [13], which will be considered below.

III. A brief introduction to the GBM

Let us consider the gradient boosting decision tree algorithm [13]. The algorithm is an iterative construction of a model as an ensemble of base (weak) prediction models built in a stage-wise fashion where each base model is constructed, based on data obtained using an ensemble of models already built on previous iterations, as an approximation of the loss function derivative. A model of size M is a linear combination of $M + 1$ base models:

$$g_M(x) = \sum_{i=0}^{M} \gamma_i h_i(x), \tag{3}$$

where h_i is the i-th base model; γ_i is the i-th coefficient or the i-th base model weight.

The gradient boosting algorithm consists of the following steps:

1) Initialize the zero base model $h_0(x)$, for example, with the constant value.
2) Calculate the residual $r_i^{(t)}$ as a partial derivative of the expected loss function $L(x_i, y_i)$ at every point of the training set, $i = 1, ..., N$.
3) Build the base model $h_t(x)$ as regression on residuals $\{(x_i, r_i^{(t)})\}$;
4) Find the optimal coefficient γ_t at $h_t(x)$ regarding the initial expected loss function (8);
5) Update the whole model $g_t(x) = g_{t-1}(x) + \gamma_t h_t(x)$;
6) If the stop condition is not fulfilled, go to step 2.

Here the loss function depends on the machine learning problem solved (classification or regression). Suppose that $(M - 1)$ steps produce the model $g_{M-1}(x)$. For constructing the model $g_M(x)$, the model $h_M(x)$ has to be constructed, i.e., there holds

$$g_M(x) = \sum_{t=0}^{M} \gamma_t h_t(x) = g_{M-1}(x) + \gamma_M h_M(x). \tag{4}$$

The dataset for constructing the model $h_M(x)$ is chosen in such a way as to approximate the expected loss function partial derivatives with respect to the function of the already constructed model $g_{M-1}(x)$. Let us denote residuals $r_i^{(M)}$ defined as the values of the loss function partial derivative at point $g_{M-1}(x_i)$ in the current iteration M,

$$r_i^{(M)} = - \left. \frac{\partial L(z, y_i)}{\partial z} \right|_{z=g_{M-1}(x_i)} . \tag{5}$$

By using the residuals, a new training set D_M is derived as follows:

$$D_M = \left\{ \left(x_i, r_i^{(M)} \right) \right\}_{i=1}^{N}, \tag{6}$$

and the model h_M can be constructed on D_M by solving the following optimization problem

$$\min \sum_{i=1}^{N} \left\| h_M(x_i) - r_i^{(M)} \right\|^2 . \tag{7}$$

Hence, an optimal coefficient γ_M of the gradient descent can be obtained as:

$$\gamma_M = \arg\min_{\gamma} \sum_{i=1}^{N} L\left[g_{M-1}(x) + \gamma h_M(x_i), y_i \right]. \tag{8}$$

Then we get the following model at every point x_i of the training set

$$\begin{aligned} g_M(x) &= g_{M-1}(x) + \gamma_M h_M(x) \\ &\approx g_{M-1}(x) - \gamma_M \left. \frac{\partial L(z, y)}{\partial z} \right|_{z=g_{M-1}(x)} . \end{aligned} \tag{9}$$

The above algorithm minimizes the expected loss function by using decision trees as base models. Its parameters include depths of trees, the learning rate, the number of iterations. They are selected to provide a high generalization and accuracy depending on an specific task.

The gradient boosting algorithm is a powerful and efficient tool for solving regression problems, which can cope with complex non-linear function dependencies [27].

IV. Improved gradient boosting algorithm

A. Motivation

Let us consider a peculiarity of GBMs constructed using decision trees. It is easy to show that a function having the GBM structure allows us to approximate any continuous function using decision trees with the number of decision rules equal to the number of features. Obviously, if we would not restrict the number of decision rules, then we can approximate any continuous function by constructing only a single tree. On the other hand, if all features are significant, then, in a general case, it is impossible to approximate any function by a linear

combination of decision trees with the number of decision rules less than the number of features. Indeed, if it were possible, then any function of several variables could be decomposed into the sum of functions of one variable, but this is incorrect.

A regression tree aims to recursively split the training set with binary tests in the form $x_i \leq t$, where x_i is one of the input variables or features, and t is a threshold. A criterion for determining the splitting parameters in the tree growing procedure is to reduce as much as possible the variance of y in the two subsamples resulting from that split. The regression tree can be seen as a kind of additive model of the form [28], [29]:

$$tree(x) = \sum_{l \in V} b_l \cdot \mathbb{I}[x \in R_l], \tag{10}$$

where $\mathbb{I}[\cdot]$ is the indicator function taking the value 1 if its argument is true and 0 otherwise; b_l is the value in the l-th leaf; R_l is the region defined by disjoint partitions of the training set at the l-th leaf; V is the set of leaves of the decision tree.

The approximation function g from (3) can be rewritten by using (10) as follows:

$$g_M(x) = \sum_{i=0}^{M} \gamma_i \sum_{l \in V_i} b_l \cdot \mathbb{I}[x \in R_l], \tag{11}$$

where V_i is the set of leaves of the i-th tree.

The indicator function $\mathbb{I}[x \in R_l]$ is determined through all decision rules corresponding to the l-th leaf:

$$\mathbb{I}[x \in R_l] = \left(\prod_{j \in Left(l)} \mathbb{I}\left[x_j \leq t_j^{(left)}\right] \right) \times \left(\prod_{j \in Right(l)} \mathbb{I}\left[x_j > t_j^{(right)}\right] \right), \tag{12}$$

where $Left(l)$ is the set of true rules leading to the l-th leaf; $Right(l)$ is the set of false rules leading to the l-th leaf; t_j is the threshold of the j-th rule.

If the number of rules corresponding to each feature in every tree is 1, then such trees correspond to the angles of parallelepipeds whose faces are hyperplanes parallel to all axes except for one axis. The entire domain of the function is covered by means of such parallelepiped angles. In addition, the entire domain of the function can be divided into equal cells with an orthogonal grid using the sum of such trees, and a unique value can be assigned to each cell. So, having selected the parameter values in a certain way, we can approximate an arbitrary continuous function with a given accuracy.

In practice, observations of target values may be noisy. Moreover, some parts of the function domain are often less densely covered with points of the training set than others. When constructing a GBM on the basis of decision trees, the GBM may have large discontinuities in regions of the absence of points as it is shown in Fig. 1. It can be seen from Fig. 1 that the distance between the target values $f(A)$ and $f(B)$ of points A and B denoted as Δ is much larger than distances between the target values of the remaining neighboring points to the left of A or to the right of B. After constructing the decision tree, distances between predictions of points to the left of T and to the right of T will be no less than Δ. This is due to the algorithm for constructing the decision tree: since the average values to the left and right of T are far from each other, then the locally optimal partition lies between A and B. As a rule, according to [30], T is chosen exactly in the middle between A and B. In the best case, if decision trees near A and B will have the values of the points closest to T, that is, the values at points A and B, then the interval of the function discontinuity predicted by the decision tree will be strictly equal to Δ, or it will be larger than Δ. As a result, using the standard algorithms for constructing decision trees, it is difficult to construct a function that behaves in the discontinuity intervals in a more smooth way. Moreover, if the region of training data is not densely covered by points, then the loss function and the absolute value of its derivative near points A and B will be close to zero even if the interval of discontinuity is large. This is because there are no observations in the region between A and B. This implies that the predicted value for an example close to T at the next iterations of the GBM will be close to zero, and, therefore, the function will not be smoothed. If the absolute value of the derivative is large enough, for example, if the predictions of the previous models (iterations) in the GBM are constant in this region, then this situation will be repeated, and the function will contain a similar discontinuity interval even of a larger size. Since there are no new points between A and B, it is impossible to obtain a split different from T in this interval.

A continuous (or piecewise continuous) function can be approximated by a piecewise constant function, having the structure of the GBM with decision trees, on a uniform grid. However, many algorithms for building decision trees like CART can build for each feature only the grid nodes located exactly in the middle between pairs of neighboring points of the training set. It should be noted that there are histogram-based methods [31] that do not necessarily build partitions exactly in the middle, but these methods consider smaller numbers of nodes which are determined by points of the training set. Therefore, the grid turns out to be larger and coarser in regions not densely covered by the points of the training set. Since decision trees correspond to piecewise constant functions, then large jumps are produced due to the rapid growth of the approximated function in such regions. The jumps lead to an increase of the loss. Moreover, the GBM does not reduce the loss because:

1) each new tree is not able to build partitions anywhere except in the middle between pairs of neighboring points;
2) empirical loss does not take into account the error in the interval between adjacent points.

B. Partially randomized decision trees and GBM

In order to solve the aforementioned problem, it is enough to ensure the construction of a denser grid, that is, to change the algorithm for building decision trees so that thresholds of the partition rules are selected not only from the set of midpoints of pairs of neighboring points of the training dataset. A simple way is to replace the threshold of each partition with a random one located within the interval corresponding to the threshold after building a tree by means of a deterministic

algorithm like CART. For example, Fig. 2 shows how the partition T is replaced by T_2, T_3 or T_4, or any other values in the interval (A, B). As a result, partitions in the ensemble trees will be independent, and will allow covering the interval more tightly. However, by taking into account the possible overfitting of the GBM as well as its inability to parallelize the training process which takes considerable time, we propose an approach which is not based on updating partitions constructed by the deterministic algorithm and has several advantages.

The basic idea behind the proposed approach is to use an algorithm for constructing partially randomized decision trees as a special case of the so-called extremely randomized trees [20]. The extremely randomized decision tree is grown by selecting at each node K random splits such that a split includes the random choice of a feature x_i and the random choice of a threshold t_i. One of the splits is fixed which maximizes the score. If the parameter K is 1, then the corresponding trees are extremely randomized ones. As indicated by Wehenkel et al. [32], the tree model may be improved with respect to a certain dataset by using larger values of K. Since we aim to compact the grid, it makes sense to consider all the features at every stage of building a tree, not limited to K random features. Otherwise, a larger number of iterations may be required to construct a sufficient number of partitioning rules for each feature, and also to ensure the convergence if the approximated function depends on the number of features exceeding K. According to the partially randomized decision trees, the cut-point (threshold) for partitioning each feature in these trees is determined randomly from a uniform distribution. Then the best feature is selected such that it maximizes the score, and the partition is performed by using this best feature in accordance with the same random cut-point.

We call decision trees constructed by the well-known deterministic algorithms, such as CART or C4.5, as deterministic trees below in order to distinguish them from partially randomized trees. A single partially randomized tree is worse on average (in the sense of the standard error) than a deterministic decision tree with locally optimal cut-points of the same depth. Moreover, it often has larger jumps at places where the deterministic tree has smaller ones, for example, smaller than the average jump between two adjacent points. However, ensembles of partially randomized trees in the form of the GBM have better properties than ensembles of deterministic trees.

First, we consider splitting procedures for one feature for simplicity and return to Fig. 1. Every tree in the ensemble independently determines its cut-point for a particular feature, perhaps more than once, that is, it splits the domain of the function into several regions (intervals) such that every region corresponds to a certain cut-point. Some of these regions contain points to the left of A, some of the them contain points to the right of B (see Fig. 2). The remaining part contains points between A and B as it is depicted in Fig. 2 (regions defined by intervals $[T_2, T_3]$, $[T_3, T_4]$). Regions of the last remaining part allow filling the interval between A and B and solve the problem of the function discontinuity.

Thus, a single partially randomized decision tree does not allow solving the aforementioned problem. It is worth noting that each tree contribution to the model is reduced by a learning rate. Hence, after adding one tree, loss will not be equal to zero near split point of the tree. It will lead to building more trees, that will split the interval along with others, making the model smoother. However, the gradient boosting machine with partially randomized decision trees allows us to construct smoother approximations of functions. We will call the proposed model as a partially randomized gradient boosting machine (PRGBM).

Let us point out two important advantages of the proposed PRGBM.

1) The PRGBM based on partially randomized decision trees has an enhanced training adaptability in comparison with extremely randomized trees because every subsequent partially randomized tree in the PRGBM is built with the aim to improve the prediction of an already constructed ensemble, whereas extremely randomized trees are built to approximate the same function.
2) An additional advantage of the PRGBM in comparison with the GBM based on deterministic trees is a significant reduction of the learning time. This follows from the fact that one partially randomized tree is built approximately N times faster than a deterministic tree because a single partition is evaluated to construct a rule for each feature in the partially randomized tree instead of N partitions in the corresponding deterministic tree. Hence, the construction time of the PRGBM in reduced in N times.

It is worth noting that the proposed approach has the following drawback: if the objective function contains intervals of discontinuity, and points from the training set cover regions around the intervals of discontinuity not densely enough, then the GBM fills the intervals with intermediate values.

Let us consider two explanation examples with one-dimensional (Fig. 3) and two-dimensional (Figs. 4-5) regression functions. The original (true) function depicted by the solid line and points corresponding to examples from a training set are shown in Fig. 3 (a). Predictions obtained by means of the GBM based on deterministic trees and partially randomized trees are depicted in the form of points in Fig. 3 (b) and Fig. 3 (c), respectively. The depth of the trees in both cases is the same and equal to 5. The horizontal axis in every figure corresponds to feature values, the vertical axis shows target values. Predictions are made on a uniform grid from 0 to 1 with 200 points. The original function is defined as follows:

$$f(x) = \sin(5x) \cdot \mathbb{I}\left[x \leq \frac{1}{2}\right] + x \cdot \mathbb{I}\left[x > \frac{1}{2}\right]. \qquad (13)$$

It can be seen from Fig. 3 (a) that the insufficient coverage by training examples of several regions of the feature negatively affects the predictions of the GBM with deterministic trees whereas predictions of the GBM with partially randomized trees successfully fill these regions. One can clearly see in Fig.3 (c) how predictions the GBM with partially randomized trees fill the discontinuity interval in the neighborhood of 0.5.

The two-dimensional function $f(x, y)$ is shown in Fig. 4. The function f is defined on the interval from 0 to 1 as follows:

$$f(x, y) = \sin(10 \cdot (x + \exp(1.1 \cdot y))).$$

It is depicted for convenience in the form of an image in Fig. 4 (a), where the horizontal and vertical axes correspond to the first x and the second y features, respectively, and the brightness corresponds to the function values $f(x, y)$. The original image consists of 100^2 points. The training set is composed as follows. A cross shown in Fig. 4 (b) is cut out of the image. Then a half of all remaining (not cut) points in the image is taken as the training set. In other words, the training set is a part of points in Fig. 4 (a) without points belonging to the cross in Fig. 4 (b). The discontinuity of the function f arises from the cross.

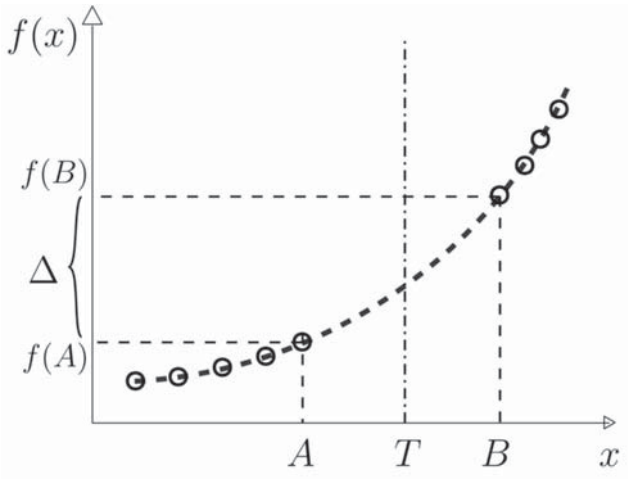

Fig. 1. Different discontinuities of the function f

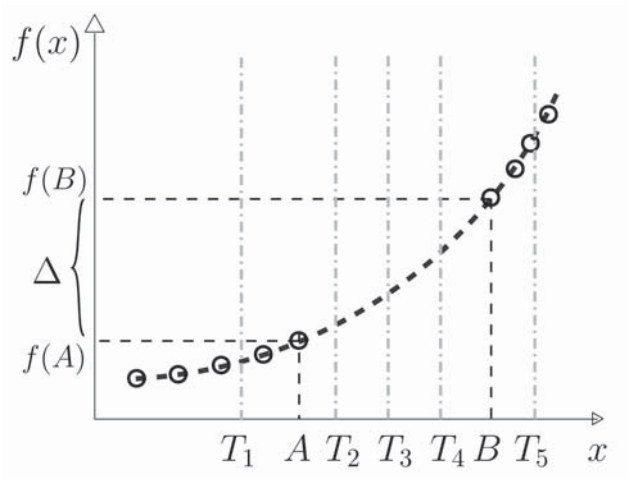

Fig. 2. The discontinuity of function f and the regions obtained by splitting

Predictions of three different models with trees of the same depth 9 are presented in Fig. 5. Fig. 5 (a) depicts results of the GBM with deterministic trees. Predictions of the GBM with extremely randomized trees are shown in Fig. 5 (b). Predictions of the GBM with partially randomized trees are depicted in Fig. 5 (c). The number of basic models in each ensemble is equal to 1000 for all cases. It can be seen from Fig. 5 (a) that the GBM with deterministic trees fills points of the cutout cross with values of the nearest points which are available in the training set. It is interesting to note that the

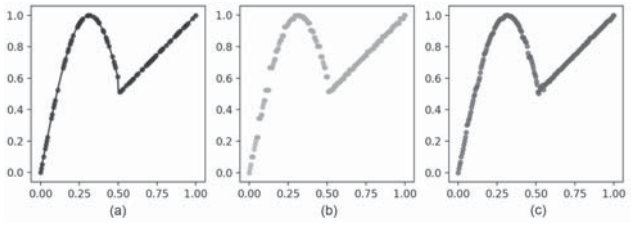

Fig. 3. Training points (a), predictions of the GBM with deterministic trees (b), and partially randomized trees (c)

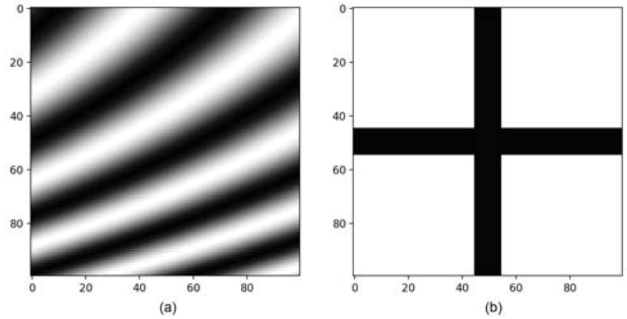

Fig. 4. The original image (a) produced by the function f, and the cross which is cut out of the image (b)

GBM with extremely randomized trees Fig. 5 (b) as well as the GBM with partially randomized trees Fig. 5 (c) fill the cross with more appropriate values than the GBM with deterministic trees. At the same time, the quality of the GBM with partially randomized trees is noticeably better than that of the GBM with extremely randomized trees.

V. Numerical experiments

In order to study the proposed approach for solving regression problems, we apply datasets described in Table I where the abbreviation of every datasets, the corresponding number of examples n, and the number of features m are shown. The datasets are taken from open sources, in particular, datasets California, Boston, and Diabetes can be found in the corresponding R Package "StatLib"; the dataset HouseART can be found in the Kaggle platform; synthetic datasets Friedman 1, 2, 3 are described at site: https://www.stat.berkeley.edu/~breiman/bagging.pdf; datasets Regression and Sparse are available in the Python Package "Scikit-Learn". Table I is a brief introduction about these datasets, while more detailed information can be found from, respectively, the data resources.

TABLE I. A brief introduction about the regression data sets

Data set	Abbreviation	m	n
California housing dataset	California	8	20640
House Prices: Advanced Regression	HouseART	79	1460
ML housing dataset	Boston	13	506
Diabetes	Diabetes	10	442
Friedman 1	Friedman 1	10	100
Friedman 2	Friedman 2	4	100
Friedman 3	Friedman 3	4	100
Scikit-Learn Regression	Regression	100	100
Scikit-Learn Sparse uncorrelated	Sparse	10	100

To evaluate the average accuracy, we perform a cross-

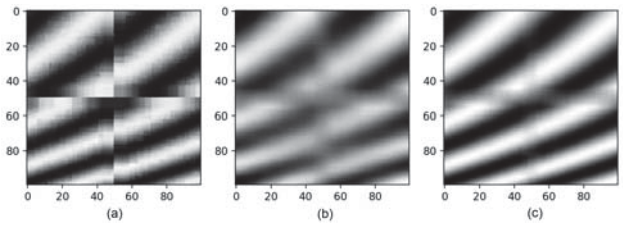

Fig. 5. Predictions of the GBM with deterministic trees (a), extremely randomized trees (b), and partially randomized trees (c)

TABLE II. COMPARISON OF PRGBM WITH RF, ERT, GBM ON THE REGRESSION DATASETS

Data set	RF	ERT	GBM	PRGBM
California	2.56×10^{-1}	2.49×10^{-1}	2.08×10^{-1}	$\mathbf{2.04 \times 10^{-1}}$
HouseART	9.80×10^{8}	9.65×10^{8}	9.53×10^{8}	$\mathbf{8.98 \times 10^{8}}$
Boston	1.22×10^{1}	1.14×10^{1}	$\mathbf{1.05 \times 10^{1}}$	1.11×10^{1}
Diabetes	3.37×10^{3}	3.24×10^{3}	3.29×10^{3}	$\mathbf{3.11 \times 10^{3}}$
Friedman 1	1.06×10^{1}	8.87×10^{0}	7.23×10^{0}	$\mathbf{4.09 \times 10^{0}}$
Friedman 2	5.84×10^{3}	1.48×10^{3}	5.24×10^{3}	$\mathbf{7.06 \times 10^{2}}$
Friedman 3	2.08×10^{-2}	1.58×10^{-2}	1.88×10^{-2}	$\mathbf{9.76 \times 10^{-3}}$
Regression	1.25×10^{4}	1.16×10^{4}	9.93×10^{3}	$\mathbf{8.98 \times 10^{3}}$
Sparse	2.79×10^{0}	2.20×10^{0}	1.95×10^{0}	$\mathbf{1.44 \times 10^{0}}$

validation with 100 repetitions, where in each run, we randomly select $n_{\text{tr}} = 3n/4$ training data and $n_{\text{test}} = n/4$ testing data. Different values for the tuning parameters have been tested, choosing those leading to the best results.

Numerical results in the form of the mean squared errors for the regression datasets are shown in Tables II and III. The best performance for each dataset is shown in bold. We compare the following six models: random forest (RF), extremely randomized trees (ERT), GBM, CatBoost [16], XGBoost [14], PRGBM. It can be seen from Tables II and III that the proposed model provides better results for 8 datasets from 9 ones.

We should point out also that the use of PRGBM reduces the training time in comparison with the GBM. Table IV shows the training time for the GBM and the PRGBM, including the time of selecting the tuning parameters by means of the cross-validation, for four datasets with the largest number of training examples. The comparison results are presented only for these two models because they provide the best prediction accuracy. It can be seen from Table IV that the PRGBM provides the best time of training for all datasets.

VI. CONCLUSION

The GBM based on partially randomized trees has been considered in the paper. The proposed model aims to reduce the computational complexity of the GBM and to take into account cases when regions of training data are not densely covered by points. The advantages of the partially randomized trees have been illustrated by means of synthetic and real data. By studying the proposed trees, we aimed to extend a large amount of the GBM modifications as well as the ensemble-based models by an additional model which may improve the regression accuracy and to reduce the model complexity. Moreover, the use of the proposed model, we solve the problem of the grid density and inject the additional randomness which may reduce the overfitting problem which arises due to greedy of the tree building procedure.

TABLE III. COMPARISON OF PRGBM WITH CATBOOST AND XGBOOST ON THE REGRESSION DATASETS

Data set	CatBoost	XGBoost	PRGBM
California	2.20×10^{-1}	2.06×10^{-1}	$\mathbf{2.04 \times 10^{-1}}$
HouseART	9.15×10^{8}	9.84×10^{8}	$\mathbf{8.98 \times 10^{8}}$
Boston	$\mathbf{1.07 \times 10^{1}}$	1.12×10^{1}	1.11×10^{1}
Diabetes	3.55×10^{3}	3.28×10^{3}	$\mathbf{3.11 \times 10^{3}}$
Friedman 1	8.49×10^{0}	7.07×10^{0}	$\mathbf{4.09 \times 10^{0}}$
Friedman 2	1.01×10^{4}	5.34×10^{3}	$\mathbf{7.06 \times 10^{2}}$
Friedman 3	2.87×10^{-2}	1.93×10^{-2}	$\mathbf{9.76 \times 10^{-3}}$
Regression	1.24×10^{4}	9.80×10^{3}	$\mathbf{8.98 \times 10^{3}}$
Sparse	2.64×10^{0}	2.05×10^{0}	$\mathbf{1.44 \times 10^{0}}$

TABLE IV. TRAINING TIMES OF THE GBM AND THE PRGBM

Data set	GBM	PRGBM
California	2.2×10^{2}	$\mathbf{4.2 \times 10^{1}}$
HouseART	3.8×10^{1}	$\mathbf{2.1 \times 10^{1}}$
Boston	5.1×10^{0}	$\mathbf{3.9 \times 10^{0}}$
Diabetes	4.1×10^{0}	$\mathbf{3.7 \times 10^{0}}$

In spite of many numerical examples which show superiority of the proposed model, the theoretical justification of the obtained results is required. It can be viewed as a direction for further research. The approach has been studied for solving the regression problem. At the same time, it is interesting to consider it also for solving the classification problem. This is another direction for further research. There are other open questions related to partially randomized trees, for example, their use in random forests, in deep forests, etc. These questions can be also regarded as direction for further research.

ACKNOWLEDGEMENT

The reported study was funded by RFBR, project number 19-29-01004.

REFERENCES

[1] A. Ferreira and M. Figueiredo, “Boosting algorithms: A review of methods, theory, and applications,” in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. New York: Springer, 2012, pp. 35–85.

[2] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. New Jersey: Wiley-Interscience, 2004.

[3] R. Polikar, “Ensemble learning,” in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. New York: Springer, 2012, pp. 1–34.

[4] L. Rokach, *Ensemble Learning: Pattern Classification Using Ensemble Methods*. World Scientific, 2019, vol. 85.

[5] O. Sagi and L. Rokach, “Ensemble learning: A survey,” *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. e1249, pp. 1–18, 2018.

[6] M. Wozniak, M. Grana, and E. Corchado, “A survey of multiple classifier systems as hybrid systems,” *Information Fusion*, pp. 3–17, 2014.

[7] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton: CRC Press, 2012.

[8] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] ——, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] T. Ho, “The random subspace method for constructing decision forests,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[11] Y. Freund and R. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[12] J. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2001.

[13] ——, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM, 2016, pp. 785–794.

[15] K. Guolin, M. Qi, F. Thomas, W. Taifeng, C. Wei, M. Weidong, Y. Qiwei, and L. Tie-Yan, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17))*, 2017, pp. 3149–3157.

[16] A. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," Oct. 2018, arXiv:1810.11363.

[17] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. Melbourne, Australia: AAAI Press, 2017, pp. 3553–3559.

[18] J. Feng, Y. Yu, and Z.-H. Zhou, "Multi-layered gradient boosting decision trees," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018, pp. 3551–3561.

[19] J. Feng, Y. Xu, Y. Jiang, and Z.-H. Zhou, "Soft gradient boosting machine," Jun. 2020, arXiv:2006.04059.

[20] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[21] T. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[22] A. Cutler and G. Zhao, "Pert-perfect random tree ensembles," *Computing Science and Statistics*, vol. 33, pp. 490–497, 2001.

[23] H. Lu and R. Mazumder, "Randomized gradient boosting machine," Oct. 2018, arXiv:1810.10158v2.

[24] H. Lu, S. Karimireddy, N. Ponomareva, and V. Mirrokni, "Accelerating gradient boosting machine," Sep. 2019, arXiv:1903.08708v2.

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

[26] A. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[27] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, no. Article 21, pp. 1–21, 2013.

[28] T. Hastie and R. Tibshirani, *Generalized additive models*. CRC press, 1990, vol. 43.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer, 2009.

[30] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and regression trees*. CRC press, 1984.

[31] A. Guryanov, "Histogram-based algorithm for building gradient boosting ensemblesof piecewise linear decision trees," in *Proceedings of the 8th International Conference on Analysis of Images, Social Networks and Texts. AIST 2019*, ser. LNCS, vol. 11832. Cham: Springer, 2019, pp. 39–50.

[32] L. Wehenkel, D. Ernst, and P. Geurts, "Ensembles of extremely randomized trees and some generic applications," in *Proceedings of Robust Methods for Power System State Estimation and Load Forecasting*, 2006, pp. 1–10. [Online]. Available: http://hdl.handle.net/2268/13447