A Novel Algorithm of Machine Learning: Fractional Gradient Boosting Decision Tree

Kangkai Gao and Yong Wang(✉)

Department of Automation, University of Science and Technology of China, Hefei 230026, China
yongwang@ustc.edu.cn

Abstract. The gradient boosting decision tree is a commonly used and effective ensemble machine learning method. In this paper, a factional gradient boosting decision tree scheme is first proposed with several loss functions discussed. This scheme implies that different algorithms can be established when adopting different fractional order gradient methods. It is then shown that with satisfactory convergence of fractional gradient descent algorithms, the proposed algorithms can train models with required accuracy in less number of iterations. Finally, several examples are provided to demonstrate the accuracy and efficiency of the algorithms.

Keywords: Fractional calculus · Gradient descent method · Ensemble method · GBDT

1 Introduction

Machine learning evolves from the broad field of artificial intelligence and plays an essential part in many areas. In the study of machine learning, ensemble methods refer to a kind of state-of-art learning approaches that combine multiple training learners. They have shown excellent performance in various machine learning applications and analytics competitions, e.g., Kaggle challenges. There are two primary techniques to associate the base learners: boosting [10,11,15] and bagging [1]. Gradient boosting is a classic ensemble technique for handling regression and classification problems. It combines the outputs of many base learners, typically decision trees, to produce a powerful learner in an additive form. Gradient boosting decision tree (GBDT) has been widely used recently mainly due to its high accuracy and fast training. Some works have been done to speed up the computation of GBDT under different parallel settings, e.g., XGBoost [3], LightGBM [12], and PLANET [14]. Some other works exploit benefits of GBDT for different machine learning applications. For instance, LambdaMART used GBDT to solve the ranking problem in [2].

S.-H. Wang and Y.-D. Zhang (Eds.): ICMTEL 2022, LNICST 446, pp. 735–748, 2022.
https://doi.org/10.1007/978-3-031-18123-8_58

In GBDT, each new tree is trained on the per-point residual defined as the negative gradient of loss function with regard to output of previous trees. Hence, GBDT can be regarded as an analogy of gradient descent method from parameter space to function space. A novel algorithm is proposed in this article by applying the fractional gradient descent method from this perspective.

The authors respectively proposed fractional gradient methods which can converge but with a low convergence rate in [5,7,16,20]. Nevertheless, it cannot be assured that the method will converge to the real global solution. The shortcoming has been partially overcome in [17]. After that, by approximating the exact fractional derivative, a new fractional gradient method was proposed by us [4,18]. Subsequently, three new viable solutions to the convergence problem were came up with in [19]. The algorithm proposed in [4,18] has been widely used in many applications, such as least mean square algorithm [6,21], system identification [8,9], recommender systems [13], etc. It is shown that the fractional gradient method is a promising algorithm. Additionally, the convergence speed should be taken into account. In [19], both theoretical analysis and simulation study indicate that all the designed methods can achieve the true convergence quickly. Therefore, the multiple learners boosted by fractional gradient may reach the required accuracy within less iterations and time costs.

Motivated by the previous discussions, a generalized fractional gradient boosting decision tree algorithm is presented in this article. It is shown by experiments with early stopping that the proposed algorithm trains the model quicker with the same tolerance compared to traditional gradient boosting decision tree. It also indicates the wide and potential applications of fractional gradient descent method.

The rest of this paper is organized as follows. A succinct review of fractional calculus, fractional gradient descent method, and ensemble methods is provided in Sect. 2. Then a detailed description of fractional gradient boosting decision tree algorithm is presented in Sect. 3, with several loss functions discussed. Section 4 exhibits several simulations to show the accuracy and effectiveness of the proposed algorithm. At last, Sect. 5 draws some conclusions.

Notation: Throughout the paper, the following notations are used. x and $\mathbf{x}$ denote the variable and vector respectively. $\Gamma(\alpha) = \int_0^\infty \mathrm{e}^{-t} t^{\alpha-1} \mathrm{d}t$ is the Gamma function which is an extension of the factorial function to real number arguments. $\arg\min_f L(f(\cdot))$ stands for argument of the minimum. The indicator function of a subset of a whole instance space is a function defined as $\boldsymbol{I}(\mathbf{x} \in R_{jm}) = \begin{cases} 0, \mathbf{x} \notin R_{jm}, \\ 1, \mathbf{x} \in R_{jm}. \end{cases}$ $E_x L(f(x))$ is the mathematical expectation of function $f(x)$ to x.

2 Preliminaries

A brief introduction of mathematical background of fractional order calculus and fractional gradient descent, and ensemble methods will be presented in this section.

2.1 Fractional Gradient Descent Method

Fractional order calculus is a generalization of integer order calculus. There are three widely used definitions for fractional order calculus, Riemann-Liouville definition, Caputo definition and Grünwald-Letnikov definition. In this article, only Riemann-Liouville and Caputo definitions are adopted.

The Riemann-Liouville derivative is expressed as

$$ {}_{c}^{\mathrm{RL}}\mathscr{D}_{x}^{\alpha} f(x) = \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_c^x \frac{f(\tau)}{(x-\tau)^{\alpha-n+1}} \mathrm{d}\tau, \tag{1} $$

where $n-1 < \alpha < n, n \in \mathbb{Z}_+, c$ is the lower terminal.

The form of Caputo derivative is as follows

$$ {}_{c}^{\mathrm{C}}\mathscr{D}_{x}^{\alpha} f(x) = \frac{1}{\Gamma(n-\alpha)} \int_c^x \frac{f^{(n)}(\tau)}{(x-\tau)^{\alpha-n+1}} \mathrm{d}\tau. \tag{2} $$

Furthermore, if function $f(x)$ can be expanded as a Taylor series, the fractional derivatives can be rewritten as

$$ {}_{c}^{\mathrm{RL}}\mathscr{D}_{x}^{\alpha} f(x) = \sum_{i=0}^{+\infty} \binom{\alpha}{i} \frac{f^{(i)}(x)}{\Gamma(i+1-\alpha)} (x-c)^{i-\alpha}, \tag{3} $$

$$ {}_{c}^{\mathrm{C}}\mathscr{D}_{x}^{\alpha} f(x) = \sum_{i=n}^{+\infty} \binom{\alpha-n}{i-n} \frac{f^{(i)}(x)}{\Gamma(i+1-\alpha)} (x-c)^{i-\alpha}, \tag{4} $$

or

$$ {}_{c}^{\mathrm{RL}}\mathscr{D}_{x}^{\alpha} f(x) = \sum_{i=0}^{+\infty} \frac{f^{(i)}(c)}{\Gamma(i+1-\alpha)} (x-c)^{i-\alpha}, \tag{5} $$

$$ {}_{c}^{\mathrm{C}}\mathscr{D}_{x}^{\alpha} f(x) = \sum_{i=n}^{+\infty} \frac{f^{(i)}(c)}{\Gamma(i+1-\alpha)} (x-c)^{i-\alpha}, \tag{6} $$

From the Formulas (3–6), we can note that the fractional derivative, both in the sense of Riemann-Liouville definition and Caputo definition, can be expanded to a series of integer order derivatives.

Gradient descent is one of the simplest of the frequently used numerical minimization methods. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

$$ x_{k+1} = x_k - \mu \nabla f(x_k), \tag{7} $$

where x_k is the current point, x_{k+1} is the next point, μ is the learning rate and $\nabla f(x_k)$ is the first-order gradient at $x = x_k$.

There are three fractional order gradient descent methods introduced in [19]: fixed memory step, higher order truncation and variable fractional order. Then the update law can be rewritten as

$$ x_{k+1} = x_k - \mu h, \tag{8} $$

where h is calculated by different equations which are listed in Eqs. (9–11).

$$h = \sum_{i=1}^{+\infty} \binom{\alpha - 1}{i - 1} \frac{f^{(i)}(x_k)}{\Gamma(i + 1 - \alpha)} (x_k - x_{k-K})^{i-\alpha}, \tag{9}$$

$$h = \frac{f^{(1)}(x_k)}{\Gamma(2 - \alpha)} (|x_k - c| + \epsilon)^{1-\alpha}, \tag{10}$$

$$h = \sum_{i=1}^{+\infty} \binom{\alpha(x_k) - 1}{i - 1} \frac{f^{(i)}(x_k)}{\Gamma(i + 1 - \alpha(x))} (x_k - c)^{i-\alpha(x)}, \tag{11}$$

thereinto, $\alpha(x)$ in Eq. (11) can be designed as follows

$$\alpha(x) = \frac{1}{1 + \beta f(x)}, \tag{12}$$

$$\alpha(x) = \frac{2}{1 + \mathrm{e}^{\beta f(x)}}, \tag{13}$$

$$\alpha(x) = 1 - \tanh(\beta f(x)). \tag{14}$$

2.2 Ensemble Methods

One major task of machine learning, pattern recognition and data mining is to construct good models from data sets. A formal formulation of learning process is as follows

$$\begin{aligned} f^* &= \arg\min_f E_{y,\mathbf{x}} L(y, f(\mathbf{x})) \\ &= \arg\min_f E_{\mathbf{x}}[E_y(L(y, f(\mathbf{x})))|\mathbf{x}], \end{aligned} \tag{15}$$

where $\mathbf{x}$ denotes the instance, y denotes the label, L is the loss function used in the problem, and f mapping $\mathbf{x}$ to y is the goal of the function estimation. Besides, if the label is categorical, the task is called classification and the learner is called classifier; if the label is numerical, the task is called regression and the learner is called fitted regression model [22].

The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models which can be decision tree, neural network, support vector machine or other kinds of machine learning algorithms. In GBDT, decision tree is adopted as the base learner. Ensemble learning can be broken down into two parts: developing a population of base learners from the training data and then combining them to form the composite predictor. The base learners can be generated in a parallel style, e.g., bagging, or in a sequential style, e.g., boosting.

3 Main Results

In this section, a fractional gradient boosting machine algorithm will be proposed with both regression and classification loss functions discussed.

3.1 Fractional Gradient Boosting Decision Tree

Decision trees partition the space of all joint predictor variable values into disjoint regions $R_j,\ j = 1,\ 2,\ \ldots,\ J$, as represented by the terminal nodes of the tree. Hence, the formula of a tree can be expressed as

$$T(\mathbf{x}; \Theta) = \sum\nolimits_{j=1}^{J} \gamma_j \boldsymbol{I}(\mathbf{x} \in R_j), \tag{16}$$

with parameters $\Theta = \{R_j, \gamma_j\}_1^J$. There are several strategies to induce a tree, e.g., ID3, C4.5, CART. The boosted trees model takes an additive form

$$f_M(\mathbf{x}) = \sum\nolimits_{m=1}^{M} T(\mathbf{x}; \Theta_m). \tag{17}$$

As mentioned before, the boosted trees are induced in a sequential way: the later trees will take the former trees into account. In this situation, we can try a greedy stagewise approach: assuming that the current model $f_{m-1}(\mathbf{x})$ is already attained, solve the Eq. (18) to grow the m-th tree at m iteration

$$\widehat{\Theta}_m = \arg\min_{\Theta_m} \sum\nolimits_{i=1}^{N} L(y_i, f_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i; \Theta_m)). \tag{18}$$

Then how to build a tree at each iteration is described next. Since there is a finite data sample $\{y_i, \mathbf{x}_i\}_i^N$ and the loss function in using $f(\mathbf{x}_i)$ to predict label on the training data is

$$L(f) = \sum\nolimits_{i=1}^{N} L(y_i, f(\mathbf{x}_i)), \tag{19}$$

to minimize the Eq. (19), a numerical optimization problem is obtained

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} L(\mathbf{f}), \tag{20}$$

where the parameters $\mathbf{f} \in \mathbb{R}^N$ are the values of the approximating function $f(\mathbf{x}_i)$ at each data point $\mathbf{x}_i$

$$\mathbf{f} = \{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)\}. \tag{21}$$

Obviously, due to the addictive form, the equation $\mathbf{f} = \sum_{m=1}^{M} \mathbf{h}_m$, where $\mathbf{h}_m \in \mathbb{R}^N$ is the predicted value of the tree at m iteration, can be obtained. Taking the greedy stagewise approach, a natural and acceptable approach is inducing a tree to fit the negative gradient of loss function at each data points by applying gradient descent method.

$$g_{im} = \frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)}, \tag{22}$$

$$\widetilde{\Theta}_m = \arg\min_{\Theta} \sum\nolimits_{i=1}^{N} (-g_{im} - T(\mathbf{x}_i; \Theta))^2, \tag{23}$$

where squared error is used to measure closeness.

GBDT can be viewed as an application of gradient descent method in function space, rather than parameter space. Hence, a modified fractional gradient boosting algorithm can be designed where fractional gradient descent is applied.

There are three fractional order gradient descent methods developed: fixed memory step, higher order truncation, and variable fractional order [19]. Since the commonly used loss function in classification problem is cross entropy and the series number of fractional gradient of cross entropy in Eq. (9) and (11) is infinite, we will only adopt higher order truncation to improve GBDT in this article. There is an emphasize of the update law expressed as

$$x_{k+1} = x_k - \mu \frac{f^{(1)}(x_k)}{\Gamma(2-\alpha)} (|x_k - c| + \epsilon)^{1-\alpha}, \tag{24}$$

where α is the fractional order, ϵ is a small nonnegative number.

Furthermore, with the initial instant varying, the equation can be re-expressed as

$$x_{k+1} = x_k - \mu \frac{f^{(1)}(x_k)}{\Gamma(2-\alpha)} (|x_k - x_{k-1}| + \epsilon)^{1-\alpha}, \tag{25}$$

where x_{k-1} is the predicted value in previous iterations.

The update law can be viewed as an optimized gradient descent method with the step length

$$\mu_k = \mu \frac{(|x_k - c| + \epsilon)^{1-\alpha}}{\Gamma(2-\alpha)}, \tag{26}$$

or

$$\mu_k = \mu \frac{(|x_k - x_{k-1}| + \epsilon)^{1-\alpha}}{\Gamma(2-\alpha)}, \tag{27}$$

varying according to the iteration.

Suppose the function f is convex and differentiable, and the gradient is Lipschitz continuous with constant $L > 0$, if the adaptive step size is less than $1/L$, the gradient descent will yield a solution f_k which satisfies

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|_2^2}{2\mu_{\min} k}. \tag{28}$$

That means the convergence rate of the method is at least sublinear.

It is worth noticing that with the iterations increasing, the distance between x_k and x_{k-1} decreases. This causes the step length become larger than it should be in classical gradient descent method. By the virtue of growing step length, the learning rate is faster than the classical gradient descent in the eventual stages.

Naturally, since fractional order gradient descent method converges to extreme points with more efficiency compared to classical gradient one, we can use the value calculated by fractional gradient descent method to induce new decision trees. Then Eq. (22) is replaced by

$$g_{im} = \frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)} \frac{[|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}}{\Gamma(2-\alpha)}, \tag{29}$$

or

$$g_{im} = \frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)} \frac{[|f_{m-1}(\mathbf{x}_i) - f_{m-2}(\mathbf{x}_i)| + \epsilon]^{1-\alpha}}{\Gamma(2-\alpha)}. \tag{30}$$

A brief description of the process is demonstrated in Algorithm 1.

Algorithm 1. Fractional gradient boosting decision tree.

1: **Input:** Data sample $\{y_i, \mathbf{x}_i\}_1^N$
2: **Initialize**: $f_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$.
3: **for** $m = 1 \to M$ **do**
4: Compute the fractional gradient as the working response

$$g_{im} = \frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)} \frac{[\big|f_{m-1}(\mathbf{x}_i) - c\big| + \epsilon]^{1-\alpha}}{\Gamma(2-\alpha)},$$

or

$$g_{im} = \frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)} \frac{[\big|f_{m-1}(\mathbf{x}_i) - f_{m-2}(\mathbf{x}_i)\big| + \epsilon]^{1-\alpha}}{\Gamma(2-\alpha)},$$

where $i = 1, \ldots, N$.
5: Build a decision tree, $T(\mathbf{x}; \Theta) = \sum_{j=1}^{J} \gamma_j \boldsymbol{I}(\mathbf{x} \in R_j)$, through CART

$$\{R_{jm}\}_1^J = J - \text{terminal node } tree(\{-g_{im}, \mathbf{x}_i\}_1^N).$$

6: Improve the fit by optimizing the coefficients for each of these terminal nodes

$$\gamma_{jm} = \arg\min_{\gamma} \sum\nolimits_{\mathbf{x}_i \in R_{jm}} L(y_i, f_{m-1}(\mathbf{x}_i) + \gamma).$$

7: Update the approximation

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum\nolimits_{j=1}^{J} \gamma_{jm} \boldsymbol{I}(\mathbf{x} \in R_{jm}).$$

8: **end for**

3.2 Loss Functions of Regression and Classification

Several popular loss criteria commonly used in regression and classification problems will be discussed in this part. In view of the only difference between Eq. (29) and (30) which is the initial point, we consider the fractional gradient with the same initial point only in this section.

Least-squares is the most commonly used regression loss function, generally known as

$$L(y_i, f(\mathbf{x}_i)) = \frac{[y_i - f(\mathbf{x}_i)]^2}{2}. \tag{31}$$

The corresponding fractional gradient is expressed as

$$g_{im} = -\frac{y_i - f_{m-1}(\mathbf{x}_i)}{\Gamma(2-\alpha)} [|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}. \tag{32}$$

The formula of least absolute deviation is shown below

$$L(y_i, f(\mathbf{x}_i)) = |y_i - f(\mathbf{x}_i)|, \tag{33}$$

and fractional gradient is displayed as

$$g_{im} = -\frac{\text{sign}(y_i - f_{m-1}(\mathbf{x}_i))}{\Gamma(2-\alpha)}[|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}. \tag{34}$$

Huber loss is more robust to outliers in data than least square loss function and also differentiable at 0 compared to least absolute deviation. It is defined as

$$L(y_i, f(\mathbf{x}_i)) = \begin{cases} \frac{1}{2}[y_i - f(\mathbf{x}_i)]^2, & |y_i - f(\mathbf{x}_i)| \le \delta, \\ \delta[|y_i - f(\mathbf{x}_i)| - \frac{\delta}{2}], & |y_i - f(\mathbf{x}_i)| > \delta. \end{cases} \tag{35}$$

Moreover, the fractional gradient goes like this

$$g_{im} = \begin{cases} -\dfrac{y_i - f_{m-1}(\mathbf{x}_i)}{\Gamma(2-\alpha)} \times \\ [|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}, \end{cases} \quad |y_i - f_{m-1}(\mathbf{x}_i)| \le \delta,$$

$$g_{im} = \begin{cases} -\dfrac{\delta\,\text{sign}(y_i - f_{m-1}(\mathbf{x}_i))}{\Gamma(2-\alpha)} \times \\ [|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}, \end{cases} \quad |y_i - f_{m-1}(\mathbf{x}_i)| > \delta. \tag{36}$$

In addition, there are other common loss functions to choose: the quantile loss, likelihood loss, and so on.

With regard to classification problem, cross entropy is generally chosen as the loss function. The binomial deviance is shown as

$$L(y_i, f(\mathbf{x}_i)) = -[y_i \log p_i + (1 - y_i) \log(1 - p_i)], \tag{37}$$

where $p_i = \frac{1}{1+\mathrm{e}^{-f(\mathbf{x}_i)}}$, $y_i \in \{0, 1\}$. It can be naturally extended to the K-class multinomial deviance loss function

$$\begin{aligned} L(\{y_i, f_k(\mathbf{x}_i)\}_1^K) &= -\sum_{k=1}^K \mathbf{I}(y_i = k) \log p_k(\mathbf{x}_i) \\ &= -\sum_{k=1}^K \mathbf{I}(y_i = k) f_k(\mathbf{x}_i) + \log(\sum_{l=1}^K \mathrm{e}^{f_l(\mathbf{x}_i)}), \end{aligned} \tag{38}$$

where $p_k(\mathbf{x}_i) = \frac{\mathrm{e}^{f_k(\mathbf{x}_i)}}{\sum_{l=1}^K \mathrm{e}^{f_l(\mathbf{x}_i)}}$, $y \in \{1, 2, \ldots, K\}$. The final estimates $\{f_k(\mathbf{x}_i)\}_1^K$ can be used to obtain corresponding probability estimates $\{p_k(\mathbf{x}_i)\}_1^K$. These in turn can be used for classification.

The fractional gradients of the two loss function can be expressed as

$$g_{im} = -\frac{y_i - p_{i,m-1}}{\Gamma(2-\alpha)}[|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}, \tag{39}$$

where $p_{i,m-1} = \frac{1}{1+\mathrm{e}^{-f_{m-1}(\mathbf{x}_i)}}$ for the binomial case, and

$$g_{kim} = -\frac{\mathbf{I}(y_i = k) - p_{k,m-1}(\mathbf{x}_i)}{\Gamma(2-\alpha)}[|f_{m-1}(\mathbf{x}_i) - c| + \epsilon]^{1-\alpha}, \tag{40}$$

where $p_{k,m-1}(\mathbf{x}_i) = \frac{\mathrm{e}^{f_{k,m-1}(\mathbf{x}_i)}}{\sum_{l=1}^K \mathrm{e}^{f_{l,m-1}(\mathbf{x}_i)}}$ for the multinomial case.

4 Experiments

Several examples are provided to illustrate the improved efficiency of the algorithms. As has been noted, fractional gradient boosting decision tree can handle both regression and classification problems, there are comparisons between the proposed and original algorithms of regression and classification datasets respectively. Since the proposed algorithm is designed to accelerate the process of training, we primarily focus on the iterations and time costs with the same tolerance. We will describe our experimental setup, datasets and the results of simulations, and analysis on the algorithms.

4.1 Regression

A major challenge in training models is how long is enough to obtain the model with desired accuracy. Early stopping is a simple, effective, and widely used approach to balance between underfitting and overfitting. That is when the improvement of the score in the last given-number iterations is less than the given tolerance $tol = 0.01$, the training procedure pauses.

For regression problem, there are three publicly available datasets adopted: boston housing price, diabetes and california housing price which is more complex than the first dataset. The basic information is summarized in Table 1.

Table 1. Datasets used in regression.

Name	Samples total	Dimensionality
Boston housing price	506	13
Diabetes	442	10
California housing price	20640	8

With the given tolerance, the averages of iterations, cost time and R^2 scores which are defined in (41) of 40 times training are listed in Table 3 with fixed initial point and Table 4 with changing initial point.

$$R^2(y, f(\mathbf{x})) = 1 - \frac{\sum_{i=1}^{N} [y_i - f(\mathbf{x}_i)]^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}, \tag{41}$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^{N} [y_i - f(\mathbf{x}_i)]^2$.

Looking at Table 3, it is obvious that with narrow difference of 0.01, fractional gradient boosting decision tree can bring nearly 2× speed-up by selecting a proper fractional order α. Nevertheless, what stands out in the table is that the convergence rate will decrease with α increasing. If α is greater than 1, the convergence rate will be less than the original algorithm. Figure 1 shows the iterations-R^2 score curves of diabetes datasets using the algorithm with the same

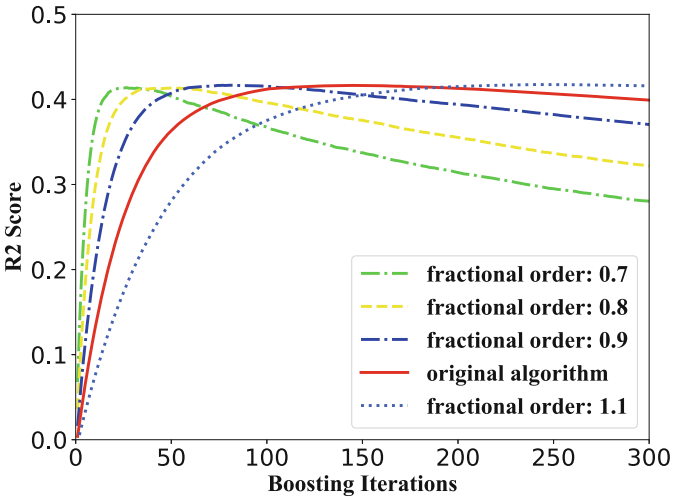

Fig. 1. Iterations-R^2 Score curve of diabetes.

initial point. The trend of the set of curves reveals the same result. Hence the range of α should be limited to $(0, 1)$.

As to the algorithm with changing initial point, the results displayed in Table 4 indicate that it can also improve the convergence speed significantly. Furthermore, the range of α can extend to $(0, 2)$. Figure 2 shows the iterations-R^2 score curves of california housing price with changing initial point.

It can be seen from these two tables that selecting appropriate parameters is an essential step of fractional gradient boosting decision tree. Otherwise, we may have a lengthy training process.

4.2 Classification

For classification problem, there are also three datasets adopted: iris, digits and a portion of forest cover type.

Table 2. Datasets used in classification.

Name	Samples total	Dimensionality
Iris	150	4
Digits	1797	64
Forest cover type	10000	54

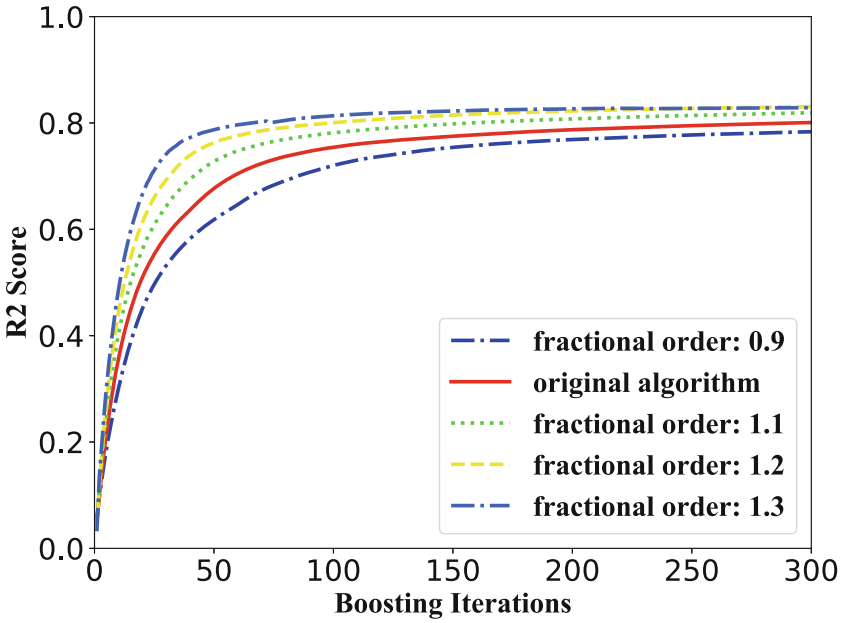

Fig. 2. Iterations-R^2 Score curve of california housing price.

Similarly, the averages of 40 times training are displayed in Table 6 with fixed initial point and Table 5 with changing initial point.

In Table 6, the results of iris and digits does not show evident improvement of fractional gradient compared to traditional algorithm. However, if we relax the restriction on accuracy score which is defined as $\text{accuracy}(y, f(\mathbf{x})) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{I}(f(\mathbf{x}_i) = y_i)$, there is also a nearly $2\times$ speed-up of forest cover type.

Table 5 details the results of fractional gradient boosting decision tree with changing initial point. But the data is not that satisfactory. The algorithm does not achieve notable but still encouraging improvement which is highlighted in Table 5.

Though the results of classification are not exactly as expected, it still indicates that fractional gradient boosting algorithm can obtain models with the same accuracy score within approximate or even much less iterations compared to classical algorithm.

Table 3. Comparison on iterations, time cost, and test scores with different α and fixed initial point.

α	Boston housing price			Diabetes			California housing price		
	Iteration number	Cost time	R^2 score	Iteration number	Cost time	R^2 score	Iteration number	Cost time	R^2 score
0.5	**29.6000**	**0.0327**	**0.8424**	8.7000	0.0096	0.2524	54.3000	0.8056	0.7486
0.6	39.3500	0.0411	0.8515	10.9500	0.0111	0.3559	56.0000	0.8099	0.7441
0.7	48.6750	0.0506	0.8553	14.3750	0.0150	0.3917	**59.1250**	**0.7977**	**0.7411**
0.8	62.3000	0.0624	0.8535	**21.0250**	**0.0205**	**0.4038**	62.8000	0.8315	0.7373
0.9	88.5000	0.0864	0.8546	33.2250	0.0297	0.4045	67.6000	0.9014	0.7330
Origin	**117.3750**	**0.1128**	**0.8534**	**55.2000**	**0.0474**	**0.4031**	**72.7250**	**0.9146**	**0.7266**
1.1	165.0000	0.1598	0.8518	94.0250	0.0809	0.4032	79.4250	1.0536	0.7200
1.2	218.7750	0.2198	0.8485	154.3250	0.1420	0.4030	86.8000	1.2118	0.7093
1.3	310.5250	0.3390	0.8476	267.9000	0.2512	0.4021	96.6750	1.4491	0.6982
1.4	453.4500	0.5147	0.8445	488.0000	0.4961	0.4011	106.3000	1.8005	0.6795
1.5	667.6000	0.7105	0.8405	884.1750	0.8544	0.4003	113.1250	1.7205	0.6459

Table 4. Comparison on iterations, time cost, and test scores with different α and changing initial point

α	Boston housing price			Diabetes			California housing price		
	Iteration number	Cost time	R^2 score	Iteration number	Cost time	R^2 score	Iteration number	Cost time	R^2 score
0.5	267.3000	0.3323	0.7560	13.9250	0.0178	0.3592	72.7000	1.2214	0.2633
0.6	206.5000	0.2548	0.7833	24.2000	0.0303	0.3835	90.4750	1.4538	0.4106
0.7	147.9750	0.1680	0.8095	**30.8250**	**0.0318**	**0.3950**	100.4250	1.5616	0.5445
0.8	113.6000	0.1159	0.8250	36.8250	0.0354	0.4002	86.6750	1.2353	0.6296
0.9	105.5750	0.1182	0.8407	45.5000	0.0465	0.4037	80.4000	1.2634	0.6931
Origin	**117.3750**	**0.1564**	**0.8534**	**55.2000**	**0.0606**	**0.4031**	**72.7250**	**1.2455**	**0.7266**
1.1	99.6750	0.1141	0.8525	54.6250	0.0566	0.4044	61.2000	0.9091	0.7477
1.2	83.8250	0.1031	0.8508	60.0250	0.0670	0.3971	53.4750	0.8782	0.7674
1.3	**70.4000**	**0.0894**	**0.8510**	62.6750	0.0714	0.3966	44.3000	0.7607	0.7764
1.4	61.2250	0.0669	0.8273	60.3750	0.0589	0.4003	38.2750	0.5752	0.7739
1.5	46.4500	0.0522	0.7808	56.7000	0.0552	0.3504	**32.1250**	**0.4865**	**0.7601**

Table 5. Comparison on iterations, time cost, and test scores with different α and changing initial point

α	Iris			Digits			Forest cover type		
	Iteration number	Cost time	Accuracy scores	Iteration number	Cost time	Accuracy scores	Iteration number	Cost time	Accuracy scores
0.5	73.8750	0.3245	0.9483	202.9750	10.0915	0.9537	491.4500	57.6011	0.7906
0.6	71.4000	0.3246	0.9467	201.1000	10.4114	0.9543	468.2000	56.5454	0.7931
0.7	70.1000	0.3467	0.9458	190.2750	10.7328	0.9544	476.4000	61.7794	0.7999
0.8	67.1250	0.3240	0.9450	181.2000	10.2598	0.9543	435.9000	57.2084	0.8035
0.9	64.6250	0.3153	0.9458	**178.5500**	**9.9963**	**0.9536**	**415.5000**	**55.2197**	**0.8066**
Origin	**63.9000**	**0.3168**	**0.9450**	**181.1750**	**9.9996**	**0.9537**	**454.6500**	**60.7997**	**0.8131**
1.1	58.2500	0.2838	0.9467	21.0500	1.2051	0.7922	28.1500	3.9193	0.6998
1.2	**54.4500**	**0.2676**	**0.9458**	7.8750	0.4599	0.4778	7.7750	1.1326	0.4931
1.3	46.7500	0.2287	0.9183	6.5000	0.3722	0.1456	7.5500	1.0984	0.3156
1.4	20.4750	0.0998	0.5900	15.4750	0.8775	0.2952	8.4500	1.2138	0.2433
1.5	27.8500	0.1407	0.6558	12.3250	0.6984	0.1955	8.9000	1.2747	0.1734

Table 6. Comparison on iterations, time cost, and test scores with different α and fixed initial point

α	Iris			Digits			Forest cover type		
	Iteration number	Cost time	Accuracy scores	Iteration number	Cost time	Accuracy scores	Iteration number	Cost time	Accuracy scores
0.5	63.1250	0.3427	0.9450	133.7750	8.8068	0.9285	78.5000	12.1962	0.7355
0.6	63.3000	0.3023	0.9450	147.4500	8.7175	0.9410	104.1500	14.1998	0.7491
0.7	63.3750	0.3214	0.9450	156.8500	9.5226	0.9456	124.7750	18.2460	0.7622
0.8	62.6500	0.2983	0.9450	**166.4750**	**9.2084**	**0.9499**	192.2500	25.4559	0.7819
0.9	63.6750	0.2989	0.9433	181.8250	10.0452	0.9527	**252.4750**	**32.7774**	**0.7930**
Origin	**63.9000**	**0.2888**	**0.9450**	**181.1750**	**9.3711**	**0.9537**	**454.6500**	**57.9821**	**0.8131**
1.1	65.4000	0.3240	0.9442	244.8250	12.6987	0.8870	307.5000	40.6255	0.7933
1.2	65.1000	0.3180	0.9458	287.6500	14.2570	0.8136	96.9750	13.2535	0.6585
1.3	**63.9000**	**0.2944**	**0.9442**	51.5500	2.5292	0.4266	38.6250	5.6046	0.5369
1.4	59.7750	0.3068	0.9083	15.4000	0.9606	0.2727	25.9500	3.7102	0.4663
1.5	55.4000	0.2931	0.9000	10.1500	0.6157	0.2008	14.3500	2.0934	0.3289

5 Conclusions

In this paper, the fractional gradient boosting decision tree algorithm has been investigated. After introducing three proposed fractional gradient methods, this study applies higher order truncation to gradient boosting for the first time to develop fractional gradient boosting decision tree and then provides several examples to evaluate the effectiveness of the proposed algorithm. These experiments confirm that fractional gradient boosting decision tree can attain well-trained models with a significantly improved speed for regression compared with the traditional GBDT method. Furthermore precise fractional order ranges are obtained for regression and classification problem respectively through the results. In general, the paper demonstrate the promising applications of the fractional gradient descent method.

References

1. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
2. Burges, C.J.: From RankNet to LambdaRANK to LambdaMART: an overview. Learning **11**(23–581), 81 (2010)
3. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
4. Chen, Y., Gao, Q., Wei, Y., Wang, Y.: Study on fractional order gradient methods. Appl. Math. Comput. **314**, 310–321 (2017)
5. Chen, Y., Wei, Y., Liang, S., Wang, Y.: Indirect model reference adaptive control for a class of fractional order systems. Commun. Nonlinear Sci. Numer. Simul. **39**, 458–471 (2016)
6. Cheng, S., Wei, Y., Chen, Y., Li, Y., Wang, Y.: An innovative fractional order LMS based on variable initial value and gradient order. Signal Process. **133**, 260–269 (2017)
7. Cheng, S., Wei, Y., Chen, Y., Liang, S., Wang, Y.: A universal modified LMS algorithm with iteration order hybrid switching. ISA Trans. **67**, 67–75 (2017)

8. Cheng, S., Wei, Y., Sheng, D., Chen, Y., Wang, Y.: Identification for Hammerstein nonlinear ARMAX systems based on multi-innovation fractional order stochastic gradient. Signal Process. **142**, 1–10 (2018)
9. Cui, R., Wei, Y., Cheng, S., Wang, Y.: An innovative parameter estimation for fractional order systems with impulse noise. ISA Trans. **82**, 120–129 (2018)
10. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001)
11. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
12. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, pp. 3146–3154 (2017)
13. Khan, Z.A., Chaudhary, N.I., Zubair, S.: Fractional stochastic gradient descent for recommender systems. Electron. Mark. **29**(2), 275–285 (2019)
14. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: PLANET: massively parallel learning of tree ensembles with MapReduce. In: Proceedings of the 35th International Conference on Very Large Data Bases, pp. 1426–1437 (2009)
15. Schapire, R.E.: A brief introduction to boosting. In: International Joint Conference on Artificial Intelligence, pp. 1401–1406 (1999)
16. Tan, Y., He, Z., Tian, B.: A novel generalization of modified LMS algorithm to fractional order. IEEE Signal Process. Lett. **22**(9), 1244–1248 (2015)
17. Tseng, C., Lee, S.: Designs of fractional derivative constrained 1-D and 2-D FIR filters in the complex domain. Signal Process. **95**, 111–125 (2014)
18. Wei, Y., Chen, Y., Cheng, S., Wang, Y.: A note on short memory principle of fractional calculus. Fractional Calculus Appl. Anal. **20**(6), 1382–1404 (2017)
19. Wei, Y., Kang, Y., Yin, W., Wang, Y.: Generalization of the gradient method with fractional order gradient direction. J. Franklin Inst. **357**, 2514–2532 (2020)
20. Wei, Y., Sun, Z., Hu, Y., Wang, Y.: On line parameter estimation based on gradient algorithm for fractional order systems. J. Control Dec. **2**(4), 219–232 (2015)
21. Yin, W., Wei, Y., Liu, T., Wang, Y.: A novel orthogonalized fractional order filtered-x normalized least mean squares algorithm for feedforward vibration rejection. Mech. Syst. Signal Process. **119**, 138–154 (2019)
22. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC, London (2012)