



TRBoost: a generic gradient boosting machine based on trust-region method

Jiaqi Luo¹ · Zihao Wei¹ · Junkai Man¹ · Shixin Xu¹

Accepted: 3 September 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Gradient Boosting Machines (GBMs) have achieved remarkable success in effectively solving a wide range of problems by leveraging Taylor expansions in functional space. Second-order Taylor-based GBMs, such as XGBoost rooted in Newton's method, consistently yield state-of-the-art results in practical applications. However, it is important to note that the loss functions used in second-order GBMs must strictly adhere to convexity requirements, specifically requiring a positive definite Hessian of the loss. This restriction significantly narrows the range of objectives, thus limiting the application scenarios. In contrast, first-order GBMs are based on the first-order gradient optimization method, enabling them to handle a diverse range of loss functions. Nevertheless, their performance may not always meet expectations. To overcome this limitation, we introduce Trust-region Boosting (TRBoost), a new and versatile Gradient Boosting Machine that combines the strengths of second-order GBMs and the versatility of first-order GBMs. In each iteration, TRBoost employs a constrained quadratic model to approximate the objective and applies the Trust-region algorithm to obtain a new learner. Unlike GBMs based on Newton's method, TRBoost does not require a positive definite Hessian, enabling its application to more loss functions while achieving competitive performance similar to second-order algorithms. Convergence analysis and numerical experiments conducted in this study confirm that TRBoost exhibits similar versatility to first-order GBMs and delivers competitive results compared to second-order GBMs. Overall, TRBoost presents a promising approach that achieves a balance between performance and generality, rendering it a valuable addition to the toolkit of machine learning practitioners.

Keywords Gradient boosting · Trust-region method

1 Introduction

Gradient Boosting Machines (GBMs) [14–16, 25] have garnered immense popularity as ensemble models, demonstrating remarkable achievements in various data science tasks. From an optimization standpoint, GBMs approximate the optimal model by employing the line search method within the hypothesis space. This approach involves

two critical steps: computing a search direction and determining an appropriate step size. In the realm of search direction methods, GBMs can be classified into two intriguing categories that reveal distinct optimization dimensions: first-order GBMs and second-order GBMs. These distinct approaches differ based on the degree of Taylor expansion utilized. First-order GBMs leverage gradient information exclusively to construct new learners. They depend on the potency of gradients to traverse the hypothesis space and unleash the potential for enhancement. Conversely, second-order GBMs delve into the realm of Newton's method, delving deeper into the intricacies of optimization. By incorporating Newton's method, these GBMs unveil an additional layer of sophistication, harnessing the full power of the Hessian of the loss function.

While both first-order and second-order methods have their merits, recent studies, such as the work by Sigrist et al. [30], consistently demonstrate the superiority of second-order methods in the realm of GBMs. These methods, lever-

✉ Shixin Xu
shixin.xu@dukekunshan.edu.cn

Jiaqi Luo
jiaqi.luo@dukekunshan.edu.cn

Zihao Wei
zihao.wei@dukekunshan.edu.cn

Junkai Man
junkai.man@dukekunshan.edu.cn

¹ Data Science Research Center, Duke Kunshan University, No.8 Duke Ave., 215300 Kunshan, Jiangsu Province, China

aging the additional information provided by the Hessian, elevate the performance of GBMs, leading to unprecedented results. However, it is crucial to recognize that second-order methods come with a notable caveat. They rely on the Hessian of the loss function being positive for optimal performance. This requirement ensures stability and enables further optimization, setting the stage for GBMs to reach their full potential. Practical applications often present scenarios where not all losses adhere to strict convexity. A prime example is label-noise learning [12], where robust losses [23, 35, 37] are commonly employed. These robust losses introduce complexities, as their Hessians are not always positive and definite, deviating from the requirements of second-order GBMs. In contrast, first-order algorithms offer a versatile alternative, breaking free from the restrictions imposed by the Hessian's positivity requirement. They provide a flexible approach that can adapt to a wide range of objective functions, making them suitable for practical scenarios where strict convexity cannot be assumed.

In summary, achieving a balance between performance and objective generality has posed a challenge for GBMs. Second-order methods demonstrate superior performance but necessitate a positive Hessian, limiting their applicability in scenarios with non-convex losses. On the other hand, first-order algorithms offer adaptability, accommodating a broader range of objective functions. Striking the right balance becomes a critical consideration in leveraging the full potential of GBMs in various data science tasks.

Note that the Taylor expansion is only a local approximation of the given function, so we can limit the variables to a small range in which the approximation function is trustworthy. When the feasible region is a compact set, according to the extreme value theorem, the optimal solution can be achieved, whether it is convex or not. This provides one way to break through the dilemma of current GBMs by adding some constraints on the region where Taylor expansion is used. The idea of adding constraints is exactly the key concept of the Trust-region method. Trust-region method [36] defines a region around the current iterate and applies a quadratic model to approximate the objective function in this region. Benefiting from the constraint, Trust-region methods do not require a quadratic coefficient positive definite.

This paper introduces TRBoost, a groundbreaking gradient boosting machine that leverages the power of the Trust-region method. By formulating the learner generation as an optimization problem in the functional space, TRBoost overcomes the challenge of handling arbitrary differentiable losses without the need for a positive Hessian, setting it apart from first-order GBMs. At the same time, it retains the superior performance characteristic of second-order GBMs. One of TRBoost's remarkable features is its adaptive radius mechanism, which empowers the boosting algorithm to dynamically adjust the target values and the

number of learners. This adaptability enhances the flexibility and efficiency of the model, allowing it to automatically fine-tune its approach based on the specific task at hand. Theoretical analysis affirms the effectiveness of TRBoost, demonstrating that it shares the same convergence rate as previous algorithms. In empirical experiments, TRBoost proves its mettle by showcasing its competitive performance across various losses and learners. Additionally, this paper delves into the role of the Hessian, shedding light on its impact within the context of TRBoost.

The main contributions of this paper are summarized as follows:

- We propose a gradient boosting machine that is compatible with commonly used learners and loss functions. It can dynamically adjust the target values and assess the new learner in each iteration.
- The algorithm maintains a trade-off between performance and generality. It demonstrates comparable efficiency to Newton's method when the loss function is strictly convex. Moreover, when the Hessian is not positive definite, similar outcomes to those obtained with the first-order method can be achieved.
- We establish that when the loss function is MAE, TRBoost exhibits a convergence rate of $O(\frac{1}{T})$; for MSE loss, TRBoost demonstrates a quadratic rate of convergence, while for Logistic loss, it achieves a linear convergence rate.
- Both theoretical analysis and experimental evidence demonstrate that gradients primarily determine the results, while Hessians contribute to further improvement in both line search-based GBMs and TRBoost.

2 Backgrounds

2.1 Gradient boosting and related work

Gradient boosting is a machine learning technique that constructs a predictive model by combining an ensemble of weak learners in a sequential manner. The fundamental idea behind gradient boosting is to iteratively optimize a loss function by adding weak models, typically decision trees, to the ensemble. Each weak model is trained to correct the errors made by the previous models, with a focus on minimizing the gradient of the loss function. This approach results in a highly accurate and robust predictive model that can handle complex patterns and interactions in the data. The flexibility, scalability, and interpretability of gradient boosting make it a popular choice for solving challenging machine learning tasks [1, 21, 29].

Mathematically, for a given feature set X and a label set Y , supervised learning is to find an optimal function

F^* in a hypothesis space $\mathcal{F} = \{F|Y = F(X)\}$ by minimizing the objective function \mathcal{L} on a given training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ ($\mathbf{x}_i \in X, y_i \in Y$):

$$\min_{F \in \mathcal{F}} \mathcal{L}(F) = \frac{1}{n} \sum_{i=1}^n l(y_i, F(\mathbf{x}_i)), \tag{1}$$

where l is the loss function.

Boosting finds F^* in a stagewise way by sequentially adding a new learner f_t to the current estimator F_{t-1} , i.e., $F_t = F_{t-1} + f_t$. Gradient Boosting is inspired by numerical optimization and is one of the most successful boosting algorithms. It originates from the work of Freund and Schapire [13] and is later developed by Friedman [14, 15].

GBMs, being functional gradient-based techniques, offer a versatile framework that can leverage various optimization approaches to construct new boosting algorithms. In the literature, we find notable examples such as gradient descent and Newton’s method, which have been adopted to design first-order and second-order GBMs [8, 14, 15]. Expanding the horizon of possibilities, Nesterov’s accelerated descent and stochastic gradient descent have been utilized to create the Accelerated Gradient Boosting [2, 24] and Stochastic Gradient Boosting [16] algorithms, respectively. These approaches introduce new dimensions to the boosting framework, unlocking enhanced performance and efficiency. In the domain of probabilistic regression, losses in probability space have paved the way for the development of novel methods. Notably, NGBoost [11] has emerged as a powerful technique that tackles probabilistic regression tasks, offering a fresh perspective on handling uncertainty and capturing the nuances of probabilistic modeling.

When it comes to base learners, linear functions, splines, and tree models are among the commonly utilized choices [6, 7, 15, 28]. However, decision trees stand out as the preferred and most popular option. Tree-based optimizations have been extensively explored, with XGBoost [8] introducing a highly efficient parallel tree learning method that empowers Gradient Boosting Decision Trees (GBDT) to handle large-scale data. Building on this success, subsequent advancements such as LightGBM [22] and CatBoost [27] have further improved the efficiency and performance of GBDT in processing big data.

Convergence analysis has garnered significant attention among researchers, and numerous studies have confirmed the convergence properties of gradient boosting. Initial works by Bickel et al. [3] demonstrated sub-linear convergence rates for logistic loss, while Telgarsky [32] later established linear convergence for the same loss function. Furthermore, researchers have shown linear convergence rates for boosting with strongly convex losses [19], showcasing the theoretical foundations and stability of gradient boosting.

2.2 Trust-region method

Suppose the unconstrained optimization is

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}), \tag{2}$$

where f is the objective function and \mathbf{x} is the decision variables. We employ a quadratic (approximation) model,

$$f(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p}, \tag{3}$$

to characterize the properties of f around \mathbf{x}_k , where \mathbf{B}_k is a symmetric matrix. It is an approximation of the Hessian matrix and does not need to be positive definite. At each iteration, Trust-region solves the following subproblem

$$\begin{aligned} \min_{\mathbf{p} \in \mathbb{R}^n} m_k(\mathbf{p}) &= f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{B}_k \mathbf{p} \\ \text{s.t. } \|\mathbf{p}\| &\leq \Delta_k, \end{aligned} \tag{4}$$

where Δ_k is the trust region radius. The strategy for choosing Δ_k is critical since the radius reflects the confidence in $m_k(\mathbf{p})$ and determines the convergence of the algorithm. $\|\cdot\|$ is a norm that is usually defined to be the Euclidean norm. The following theorem provides the condition for problem (4) to process a global solution.

Theorem 1 *The vector \mathbf{p}^* is a global solution of (4) if and only if \mathbf{p}^* is feasible and there is a scalar $\lambda_k \geq 0$ such that the following conditions are satisfied:*

$$\begin{aligned} (\mathbf{B}_k + \lambda_k \mathbf{I}_k) \mathbf{p}^* &= -\nabla f(\mathbf{x}_k), \\ \lambda_k (\Delta_k - \|\mathbf{p}^*\|) &= 0, \\ (\mathbf{B}_k + \lambda_k \mathbf{I}_k) &\text{ is positive semi-definite.} \end{aligned}$$

Given a step \mathbf{p}_k , we define the ratio between the actual reduction and the predicted reduction

$$\rho_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{p}_k)}{m_k(\mathbf{0}) - m_k(\mathbf{p}_k)}, \tag{5}$$

which measures how well $m_k(\mathbf{p})$ approximates and helps to adjust the radius Δ_k . The following Algorithm 1 describes the standard Trust-region method, more details and the proof of Theorem 1 can be found in [26].

In practice, $\hat{\rho}_1 = 0.25, \hat{\rho}_2 = 0.75, \gamma_1 = 0.25, \gamma_2 = 2$ are taken as default.

3 Trust-region gradient boosting

In this section, we begin by presenting the general formulation of the TRBoost algorithm. Subsequently, we focus on

Algorithm 1 Trust-region algorithm.

```

1: Given max radius  $\Delta_{max}$ , initial radius  $\Delta_0 \in (0, \Delta_{max})$ , initial point
    $x_0, k \leftarrow 0$ ;
2: Given parameters  $0 \leq \eta < \hat{\rho}_1 < \hat{\rho}_2 < 1, \gamma_1 < 1 < \gamma_2$ ;
3: while Stop condition not met do
4:   Obtain  $\mathbf{p}_k$  by solving (4);
5:   Evaluate  $\rho_k$  from (5);
6:   if  $\rho_k < \hat{\rho}_1$  then
7:      $\Delta_{k+1} = \gamma_1 \Delta_k$ ;
8:   else
9:     if  $\rho_k > \hat{\rho}_2$  and  $\|\mathbf{p}_k\| = \Delta_k$  then
10:       $\Delta_{k+1} = \min(\gamma_2 \Delta_k, \Delta_{max})$ ;
11:     else
12:       $\Delta_{k+1} = \Delta_k$ ;
13:     end if
14:   end if
15:   if  $\rho_k > \eta$  then
16:      $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ ;
17:   else
18:      $\mathbf{x}_{k+1} = \mathbf{x}_k$ ;
19:   end if
20: end while

```

the specific scenario where decision trees are employed as base learners.

3.1 General formulation

Objective Given a dataset with n samples and m features $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ ($\mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}$), we define the objective of iteration t as

$$\mathcal{L}^t = \frac{1}{n} \sum_{i=1}^n l(y_i, \hat{y}_i^{t-1} + f_t(\mathbf{x}_i)), \tag{6}$$

where $t \in [1, T]$ is the learner number, \hat{y}_i^{t-1} is the prediction from the previous $t - 1$ learners, f_t is a new weak learner and l is the loss function that does not need to be strictly convex.

Optimization and target values Trust-region method is employed to optimize the objective, which has the following formulation

$$\begin{aligned} \min_{f_t} \tilde{\mathcal{L}}^t &= \frac{1}{n} \sum_{i=1}^n (g_i^{t-1} f_t(\mathbf{x}_i) + \frac{1}{2} b_i^{t-1} f_t(\mathbf{x}_i)^2), \\ \text{s.t. } |f_t(\mathbf{x}_i)| &\leq \Delta_i^t, i = 1, 2, \dots, n, \end{aligned} \tag{7}$$

where $g_i^{t-1} = \frac{\partial l(y_i, \hat{y}_i^{t-1})}{\partial \hat{y}_i^{t-1}}$, b_i^{t-1} is the second derivative $\frac{\partial^2 l(y_i, \hat{y}_i^{t-1})}{\partial (\hat{y}_i^{t-1})^2}$ or some approximation.

Since each item $l(y_i, \hat{y}_i^{t-1} + f_t(\mathbf{x}_i))$ is independent, we can split (7) into following n one-dimensional constrained

optimization problems,

$$\begin{aligned} \min_{z_i^t} &g_i^{t-1} z_i^t + \frac{1}{2} b_i^{t-1} (z_i^t)^2 \\ \text{s.t. } &|z_i^t| \leq \Delta_i^t, \end{aligned} \tag{8}$$

where the solution z_i^t is the target that f_t needs to fit. They are easy to solve and each z_i^t has analytical expression, which is

$$z_i^t = \begin{cases} \text{sgn}(-g_i^{t-1}) \times \min(|\frac{g_i^{t-1}}{b_i^{t-1}}|, \Delta_i^t), & b_i^{t-1} > 0; \\ \text{sgn}(-g_i^{t-1}) \times \Delta_i^t, & b_i^{t-1} \leq 0. \end{cases} \tag{9}$$

The segmented form (9) can be rewritten in the following format

$$z_i^t = \frac{-g_i^{t-1}}{b_i^{t-1} + \mu_i^t}, \tag{10}$$

which matches Theorem 1. Here μ_i^t is a non-negative scalar related to Δ_i^t ensuring the positive denominator $b_i^{t-1} + \mu_i^t$. The relationship between μ_i^t and Δ_i^t is as follows:

- (a) If $|\frac{g_i^{t-1}}{b_i^{t-1}}| \geq \Delta_i^t$, then $z_i^t = \text{sgn}(-g_i^{t-1})|\frac{g_i^{t-1}}{b_i^{t-1}}| = \frac{-g_i^{t-1}}{b_i^{t-1}}$.
Let $\mu_i^t = 0$, then $z_i^t = \frac{-g_i^{t-1}}{b_i^{t-1} + \mu_i^t}$.
- (b) If $|\frac{g_i^{t-1}}{b_i^{t-1}}| < \Delta_i^t$ or $b_i^{t-1} \leq 0$, then $z_i^t = \text{sgn}(-g_i^{t-1})\Delta_i^t$.
Let $\mu_i^t = \frac{|g_i^{t-1}|}{\Delta_i^t} - b_i^{t-1}$, then $z_i^t = \frac{-g_i^{t-1}}{b_i^{t-1} + \mu_i^t}$.

As the number of instances increases, there will be a mass of μ_i^t that needs to be adjusted in each iteration, which will be time-consuming. Therefore, we replace different Δ_i^t with single value $\mu^t = \max\{\mu_i^t\}_{i=1}^n$ for simplicity and obtain the target value

$$z_i^t = \frac{-g_i^{t-1}}{b_i^{t-1} + \mu^t}. \tag{11}$$

Update strategy Let $\hat{z}_i^t = f_t(\mathbf{x}_i)$ be the output of the new learner, we provide two ratios to determine how to update μ^t . The first one is the same as (5), which is

$$\rho_1^t = \frac{\mathcal{L}^{t-1} - \mathcal{L}^t}{-\frac{1}{n} \sum_{i=1}^n [g_i^{t-1} \hat{z}_i^t + \frac{1}{2} b_i^{t-1} (\hat{z}_i^t)^2]}. \tag{12}$$

When $b_i^t = \frac{\partial^2 l(y_i, \hat{y}_i^{t-1})}{\partial (\hat{y}_i^{t-1})^2}$ and the loss function is L_2 loss, ρ_1^t is always 1, so that μ^t can not be updated. Therefore, the

second ratio is introduced based on the difference of \mathcal{L} to overcome the drawbacks, which is

$$\rho_2^t = \frac{\mathcal{L}^{t-1} - \mathcal{L}^t}{\frac{1}{n} \sum_{i=1}^n |\hat{z}_i^t|}. \tag{13}$$

Algorithm The detailed procedure of generic TRBoost is presented in Algorithm 2, which is on the basis of Algorithm 1. ϵ_1 , ϵ_2 , and γ are three constants that control the update of μ^t . If $\rho^t < \epsilon_1$, it means that the radius is too large such that the Taylor approximation is not good. On the other hand, although $\rho^t > \epsilon_2$ implies a good approximation, the sharp decrease in the objective \mathcal{L}^t may induce overfitting. Hence in both cases, we need to reduce the radius (enlarge μ^t) and let ρ^t be close to 1. The default values of ϵ_1 and ϵ_2 are 0.9 and 1.1 respectively. Besides, large γ will cause the target values (11) to drop to 0 fast and make the algorithm converge early, so we set the default value to be 1.01. Moreover, $\rho^t < \eta$ indicates that the corresponding weak learner contributes little to the decline of the objective function, thus it will not be added to the ensemble, which can reduce the model complexity and lower the risk of overfitting.

Algorithm 2 Generic trust-region gradient boosting.

- 1: Given initial constants $\mu^0 \geq 0$, initial tree F_0 ;
- 2: Given parameters $0 \leq \eta \leq \epsilon_1 < 1 < \epsilon_2, \gamma > 1$;
- 3: **while** Stop condition not met **do**
- 4: Calculate $z_i^t, i = 1, 2, \dots, n$ from (11);
- 5: Obtain f_i by fitting $\{x_i, z_i^t\}_{i=1}^n$;
- 6: Compute ρ^t using (12) or (13);
- 7: **if** $\rho^t < \epsilon_1$ or $\rho^t > \epsilon_2$ **then**
- 8: $\mu^{t+1} = \gamma \mu^t$;
- 9: **else**
- 10: $\mu^{t+1} = \mu^t$;
- 11: **end if**
- 12: **if** $\rho^t > \eta$ **then**
- 13: $F_{t+1} = F_t + f_i$;
- 14: **else**
- 15: $F_{t+1} = F_t$;
- 16: **end if**
- 17: **end while**

3.2 Trust-region boosting tree

Leaf values For a tree with a fixed structure, it can be written as a piecewise linear function $f_i(\mathbf{x}) = \sum_{j=1}^k C_j^t I(\mathbf{x} \in R_j^t)$, where $C_j^t \in \mathbb{R}$ is the value of leaf j and $R_j^t \subseteq \mathbb{R}^m$ is the corresponding region. Suppose the instances in R_j^t is $\{\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \dots, \mathbf{x}_{j_\ell}\}$, we have

$$|C_j^t| = |f_i(\mathbf{x}_s)| \leq \Delta_s^t, s = j_1, j_2, \dots, j_\ell.$$

Let $\Delta_j^t = \min\{\Delta_s^t\}_{s=j_\ell}^{j_1}$, then $|C_j^t| \leq \Delta_j^t$.

Substituting $f_i(\mathbf{x})$ to (7) we have

$$\begin{aligned} \min_{C_1^t, \dots, C_k^t} & \frac{1}{n} \sum_{j=1}^k (C_j^t (\sum_{x_i \in R_j^t} g_i^{t-1}) + \frac{1}{2} (C_j^t)^2 (\sum_{x_i \in R_j^t} b_i^{t-1})) \\ \text{s.t. } & |C_j^t| \leq \Delta_j^t, j = 1, \dots, k. \end{aligned} \tag{14}$$

Since there is no intersection between different region R_j^t , problem (14) is equivalent to the following k models about $C_j^t, j = 1, 2, \dots, k$:

$$\begin{aligned} \min_{C_j^t} & \frac{1}{2} B_j^{t-1} (C_j^t)^2 + G_j^{t-1} C_j^t \\ \text{s.t. } & |C_j^t| \leq \Delta_j^t, \end{aligned} \tag{15}$$

where $G_j^{t-1} = \sum_{x_i \in R_j^t} g_i^{t-1}, B_j^{t-1} = \sum_{x_i \in R_j^t} b_i^{t-1}$.

Same as (11), the optimal leaf value C_j^t can be computed by

$$C_j^t = \frac{-G_j^{t-1}}{B_j^{t-1} + \mu_j^t}. \tag{16}$$

We use a variant of μ_j^t to reduce the influence of the number of instances on C_j^t , which is defined as $\mu_j^t = \alpha^t n_j^t + \beta^t$. Here n_j^t is the number of instances contained in leaf j , α^t and β^t are two constants needed to be updated like μ^t in Algorithm 2, i.e., $\alpha^{t+1} = \gamma \alpha^t, \beta^{t+1} = \gamma \beta^t$. Their default values are 0.1 and 10 respectively. And (16) now becomes

$$C_j^t = \frac{-G_j^{t-1}}{B_j^{t-1} + \alpha^t n_j^t + \beta^t}. \tag{17}$$

After getting the leaf value, we can calculate the corresponding optimal value by

$$\tilde{\mathcal{L}}_j^t = \frac{1}{2} B_j^{t-1} (C_j^t)^2 + G_j^{t-1} C_j^t. \tag{18}$$

Splitting rules Different functions such as *gini impurity* [5], *loss reduction* [8] and *variance gain* [22] can be applied to decide whether the leaf node should split. In our current implementation version, we choose loss reduction as the splitting function.

Suppose $R_P = R_L \cup R_R, R_L$, and R_R are the corresponding regions of the left and right nodes after the parent node R_P split. The loss reduction in our method is defined as follows:

$$\begin{aligned} \mathcal{L}_{split} = & \frac{1}{2} \left[\frac{G_L^2 B_L}{(B_L + \mu_L)^2} + \frac{G_R^2 B_R}{(B_R + \mu_R)^2} - \frac{G_P^2 B_P}{(B_P + \mu_P)^2} \right] \\ & + \left(\frac{-G_L^2}{B_L + \mu_L} + \frac{-G_R^2}{B_R + \mu_R} - \frac{-G_P^2}{B_P + \mu_P} \right) \end{aligned}$$

where $G_* = \sum_{x_i \in R_*} g_i$, $B_* = \sum_{x_i \in R_*} b_i$ and $\mu_* = \alpha n_* + \beta$, $* \in \{P, L, R\}$.

Comparison with XGBoost We conclude this subsection by conducting a comprehensive comparison between TRBoost and XGBoost, which highlights the superior performance of TRBoost.

- (1) **Approximation model:** The approximation model of XGBoost at step t is

$$\tilde{\mathcal{L}}_{XGB}^t = \sum_{i=1}^n (g_i^{t-1} f_t(\mathbf{x}_i) + \frac{1}{2} h_i^{t-1} f_t(\mathbf{x}_i)^2) + \Omega(f_t).$$

This model is unconstrained and requires $h_i^{t-1} > 0$ to ensure that the minimization problem can be solved. In contrast, the bounded feasible region in TRBoost makes the approximation model able to be solved whatever the sign of the quadratic term is.

- (2) **Leaf value:** Although the leaf value of both methods can be written as $C = \frac{-G}{H+\mu}$, the C in TRBoost is more flexible. H does not need to be strictly positive and μ can be updated during iterations.
- (3) **Boosting strategy:** In each iteration, XGBoost adds all new learners to the ensemble model, while TRBoost adds selectively.

Since our method and Newton’s method both belong to the quadratic model, the proof process is similar. We follow the proof ideas of Sun et al. [31] and use the same notations.

We present the main results below and the supplemental materials are provided in Appendix A and B. For the proofs and more details of the introduced theorems, we refer interested readers to the original paper [31].

Theorem 2 [Convergence rate of TRBoost] *When the loss is the mean absolute error (MAE), TRBoost has $O(\frac{1}{T})$ rate; when the loss is MSE, TRBoost has a quadratic rate and a linear rate is achieved when the loss is Logistic loss.*

4 Experimental results

4.1 Experiments setup

Datasets

We conduct comparisons using 12 datasets obtained from the UCI Machine Learning Repository [10] and OpenML [33]. The details of these datasets can be found in Table 1. To ensure fair evaluations, we randomly partition each dataset, reserving 80% of the instances for training and the remaining 20% for testing. Within the training data, we further hold out 20% as a validation set for parameter selection using the grid search method. Once the best parameters are determined, we

Table 1 Datasets used in experiments

Datasets	#Ins./#Feat.	Task/Loss	Metric
Adult	32561/14	Clf/Log	AUC&F1
German	1000/20	Clf/Log	AUC&F1
Electricity	45312/8	Clf/Log	AUC&F1
Sonar	208/60	Clf/Log	AUC&F1
Credit	1000/20	Clf/Log	AUC&F1
Spam	4601/57	Clf/Log	AUC&F1
California	20634/8	Reg/ L_2	Loss
Concrete	1030/8	Reg/ L_2	Loss
Pol	15000/49	Reg/ L_2	Loss
Kin8nm	8192/8	Reg/ L_2	Loss
CPU-act	8192/22	Reg/ L_2	Loss
Wine quality	6497/11	Reg/ L_2	Loss

#Ins. means the instance number of a dataset, and #Feat. represents the feature number. Clf means the task type is classification, and Reg stands for regression

retrain the models and evaluated their performance on the test set. This entire process is repeated 5 times for each dataset to account for variability.

Loss functions and evaluation metric

Two commonly used convex losses are employed for experiments, namely *Log Loss* in classification and *Squared Error (L_2 Loss)* in regression. For the binary classification task, *AUC* [9] and *F1-score* are chosen as the evaluation metric. For the regression task, we utilize the loss functions to assess the performance of different methods. In particular, we select $\frac{\partial^2 l(y_i, \hat{y}_i^{t-1})}{\partial (\hat{y}_i^{t-1})^2}$ as b_i^{t-1} in (11).

Implementation details We choose five widely used tree methods to make comparisons and employ the official open-source implementations for these models^{1 2 3}. Based on the gradient information used, GBDT belongs to the first-order algorithms category, while XGBoost and LightGBM fall under the second-order algorithms category. In TRBoost⁴, we choose Decision Tree, Linear Regression, and Cubic Spline [20] as the base learners. We implement the decision tree model ourselves using Python, while the other two models are implemented using the Scikit-Learn library. No special preprocessing is performed on categorical features. All experiments are conducted on a workstation running Ubuntu 20.04, equipped with an Intel Core i9-10900X CPU and 128GB of memory.

¹ XGBoost: <https://xgboost.readthedocs.io/en/stable/>

² LightGBM: <https://lightgbm.readthedocs.io/en/latest/>

³ GBDT & Random Forest & Decision Tree: <https://scikit-learn.org/stable/index.html>

⁴ <https://github.com/Luojiaqimath/TRBoost>

4.2 General comparisons

Numerical results of different algorithms with optimal parameters are presented in Fig. 1 and Tables 2 and 3. The row named TRB-Tree denotes that the base learner is the decision tree and the approximation ratio is ρ_1^t , TRB-Tree(D) means the ratio is ρ_2^t , and the last two rows represent the model whose base learner is Linear Regression and Spline respectively.

In summary, TRBoost demonstrates remarkable effectiveness, particularly in terms of average rank, which sets it apart as a standout performer among boosting methods. On average, TRBoost outperforms other methods, establishing its superiority in terms of overall performance.

Table 2 presents the results of different algorithms, highlighting the consistent performance of TRBoost, which consistently secures a position within the top 2 performers. While it may not consistently achieve the absolute best results, TRBoost demonstrates comparable performance to other boosting methods, particularly in terms of AUC. However, where it truly excels is in terms of the F1-score, where it outperforms other approaches, providing a clear advantage.

In the realm of regression problems, TRBoost takes the lead, outperforming not only XGBoost and GBDT but also formidable competitors like LightGBM and Random Forest. While it may fall short of the top spot on the Pol dataset, its exceptional performance across other datasets speaks volumes about its capabilities.

Among our four boosting machines, it becomes evident that tree-based learners hold a distinct advantage when working with tabular data. While other learners may shine in specific cases, they face limitations imposed by the datasets at hand. Consider Linear Regression, for instance. It boasts impressive speed but struggles when confronted with complex data distributions. On the other hand, Spline learners yield superior outcomes, but their calculation time increases in proportion to both the feature dimension and data size. In practice, when we possess prior knowledge about the data, such as its approximate distribution, leveraging a learner based on these insights allows us to achieve satisfactory results with a minimal number of estimators, significantly reducing computational costs. However, in the absence of such priors, the tree-based learner remains the optimal choice [4, 18], providing a reliable and versatile approach.

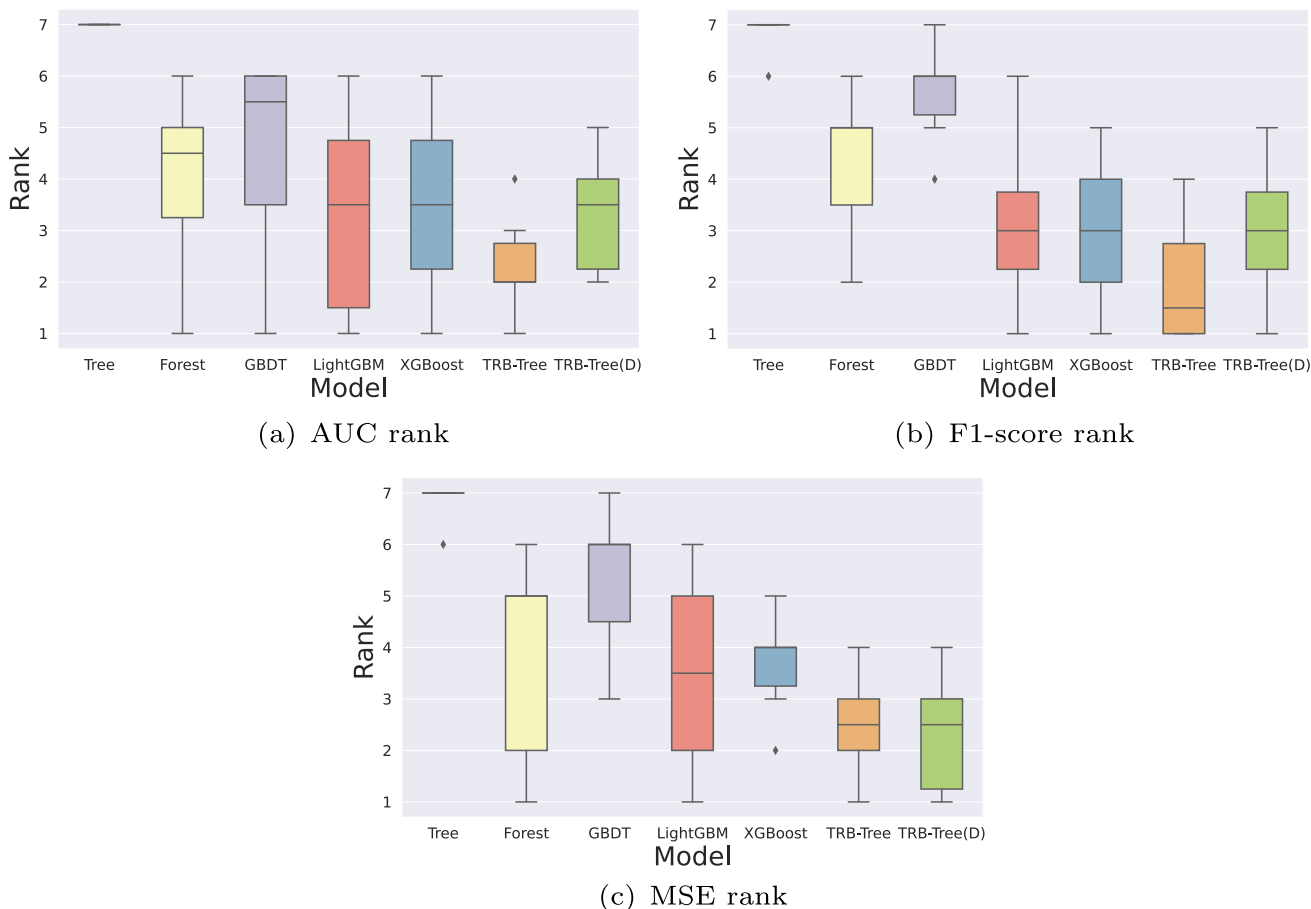


Fig. 1 Rank values of different models on 12 datasets. Fig. (a) and (b) are for classification tasks and Fig. (c) is for regression tasks

Table 2 Mean results of different models on classification tasks

Model	Metric	Adult	German	Electricity	Sonar	Credit	Spam
Decision Tree	AUC	89.02	67.06	86.02	77.29	65.81	93.27
	F1-score	63.51	41.50	70.36	73.37	78.55	88.63
Random Forest	AUC	90.19	76.12	96.68	94.63	74.54	98.62
	F1-score	67.46	<u>50.14</u>	88.31	82.56	84.05	93.87
GBDT	AUC	92.40	76.25	95.08	92.13	<u>74.91</u>	98.24
	F1-score	70.77	40.20	85.88	81.35	83.68	93.36
LightGBM	AUC	92.70	75.91	97.30	94.33	73.24	98.84
	F1-score	<u>71.20</u>	49.10	90.04	85.22	82.00	94.46
XGBoost	AUC	<u>92.69</u>	75.07	97.70	93.93	74.90	98.64
	F1-score	70.92	46.12	90.86	<u>85.33</u>	<u>84.16</u>	93.99
TRB-Tree	AUC	<u>92.69</u>	<u>76.99</u>	<u>97.41</u>	<u>94.36</u>	74.44	<u>98.72</u>
	F1-score	71.51	52.74	<u>90.10</u>	84.74	84.55	94.06
TRB-Tree(D)	AUC	92.65	76.42	97.07	94.59	73.27	98.68
	F1-score	71.08	49.34	89.43	86.77	83.34	<u>94.19</u>
TRB-LR	AUC	83.76	77.25	80.81	78.04	68.24	95.86
	F1-score	31.52	41.43	65.21	64.79	82.63	84.06
TRB-Spline	AUC	90.48	76.02	86.26	75.71	76.01	97.11
	F1-score	57.91	43.94	73.35	66.02	83.76	89.74

The top results for each dataset are in **bold**, we also underline the second-best results

4.3 Label-noise learning

Label noise is a common issue that frequently arises in classification and regression datasets, posing a significant challenge to the performance of models trained on such data. To mitigate this problem, noise-robust loss functions are commonly employed. In this section, we show that TRBoost can employ robust loss to obtain better performance when datasets have noise.

4.3.1 Classification

Recent research [17] has theoretically shown that mean absolute error (MAE) can be robust to noisy labels for classification tasks under some assumptions. However, a drawback

of MAE is its lack of positive definiteness, which renders them incompatible with second-order GBDT. Here, we demonstrate that MAE can be applied to TRBoost.

The MAE loss for binary classification can be calculated as

$$l(y, \hat{y}) = |y - \text{Sigmoid}(\hat{y})|, \tag{19}$$

where $y \in \{0, 1\}$ is the label, \hat{y} is the prediction of previous estimators, $p = \text{Sigmoid}(\hat{y}) = \frac{1}{1+e^{(-\hat{y})}}$ represents the predicted probability of the positive class. The gradient and the Hessian are $p(1-p)(1-2y)$ and $p(1-p)(1-2p)(1-2y)$, respectively. We can see that the Hessian is negative when $p < 0.5, y = 1$ or $p > 0.5, y = 0$, hence second-order boosting method can not use this loss.

Table 3 Mean results of different models on regression tasks

Model	California	Concrete	Pol	Kin8nm	CPU-act	Wine quality
Decision Tree	0.2815	44.60	48.88	0.0470	12.44	0.3394
Random Forest	0.1477	23.97	22.62	0.0196	6.06	0.1597
GBDT	0.1939	18.54	78.00	0.0251	5.91	0.1735
LightGBM	0.1673	19.27	<u>22.89</u>	0.0155	6.18	<u>0.1438</u>
XGBoost	0.1630	18.87	26.53	0.0177	<u>5.42</u>	0.1475
TRB-Tree	0.1646	<u>17.54</u>	26.22	<u>0.0159</u>	5.54	0.1407
TRB-Tree(D)	<u>0.1622</u>	17.43	26.44	0.0167	5.15	0.1442
TRB-LR	0.3876	103.43	935.92	0.0410	94.08	0.2749
TRB-Spline	0.3115	33.75	500.11	0.0392	8.82	0.2190

The top results for each dataset are in **bold**, we also underline the second-best results

Table 4 Mean AUC results of five models with different losses

Fraction	GBDT	LightGBM	XGBoost	TRB-CE	TRB-MAE
0.0	<u>98.56</u>	99.03	98.09	98.54	98.21
0.1	93.37	92.59	93.45	93.82	<u>93.77</u>
0.2	<u>88.86</u>	86.61	88.51	88.80	89.38
0.3	85.68	84.75	85.39	<u>85.66</u>	85.51
0.4	76.25	78.47	76.94	77.60	<u>78.43</u>
0.5	66.58	67.23	<u>69.51</u>	68.87	71.78
0.6	<u>64.75</u>	63.63	62.78	64.67	69.78

We embark on an experiment to unveil the unrivaled superiority of TRBoost when it comes to conquering the formidable realm of label noise learning problems. Our test dataset is meticulously crafted using the powerful *make classification* function within Scikit-Learn, where a fraction of the samples had their classes assigned randomly. By manipulating this fraction, we can introduce varying degrees of noisy labels, thereby intensifying the challenge.

To gauge the true prowess of TRBoost, we compare its performance against other boosting models, all utilizing the cross-entropy (CE) loss function. However, TRBoost stands apart by employing the MAE loss. The astounding results are meticulously presented in Table 4, with the accompanying visual masterpiece, Fig. 2, adding a touch of elegance. As the fraction of noisy labels escalated, signifying an increasingly complex landscape, the remarkable benefits of TRB-MAE become more pronounced, leaving no doubt about its excellent capability to tackle label noise head-on.

4.3.2 Regression

In our pursuit of evaluating the effectiveness of TRBoost in label-noise regression tasks, we explore its robustness by introducing strong outliers. To achieve this, we utilize the ver-

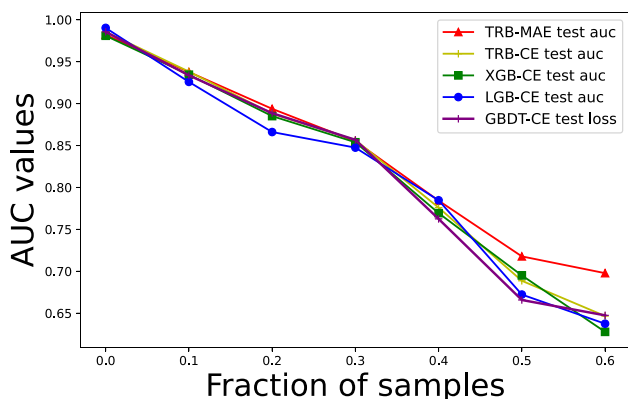


Fig. 2 AUC curves of five models with different loss functions. Red line: TRBoost with MAE loss; Yellow line: TRBoost with CE loss; Green line: XGBoost with CE loss; Blue line: LightGBM with CE loss; Purple line: GBDT with CE loss

satile *make regression* function in Scikit-Learn, which injects a significant 10% of strong outliers into the training data. To assess the performance of different models, we employ two non-strictly convex loss functions: the *Absolute Loss* (L_1 Loss) and the *Huber Loss*.

In the insightful Table 5, a fascinating trend emerges. When the loss function lacks strict convexity, models like GBDT and TRBoost remain applicable and deliver commendable results. However, second-order GBMs fail to meet the mark in such scenarios. Furthermore, Fig. 3(b) illustrates that the inclusion of a nonzero quadratic term in the loss function makes TRBoost superior to GBDT in terms of convergence rate and performance.

5 Discussion

5.1 Impact of hessian

In this session, we discuss the impact of second-order approximation in Gradient Boosting, suppose the step of first-order GBMs, second-order GBMs, and TRBoost, is $-\nu g$, $\frac{-\nu g}{h}$, and $\frac{-g}{h+\mu}$, respectively.

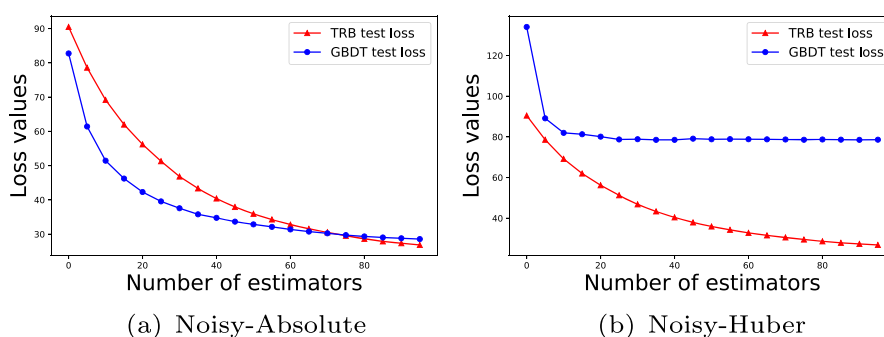
Figure 4 gives two loss curves of datasets previously used. It can be seen that the loss curves of XGBoost, LightGBM, and TRBoost converge faster and better than those of GBDT. There are two reasons for this, one is that the Hessian h , which is a function of the predicted probability p , is less than 1 and monotonically decreasing, hence it makes $|\frac{-\nu g}{h}|$ greater than $|\nu g|$ and accelerates the decline of the objective function. The other is that $|\frac{\nu g}{h}| > |\nu g|$ makes the probability $p = \psi(\frac{\nu g}{h})$ predicted by the second-order method closer to 0 or 1 than the first-order method, so the corresponding objec-

Table 5 Results on noisy regression tasks

Dataset	GBDT	LightGBM	XGBoost	TRB-Tree
Noisy-Absolute	28.6531	-	-	27.1661
Noisy-Huber	37.8729	-	-	26.4561

The top results for each dataset are in **bold**

Fig. 3 Test curves for GBDT and TRBoost-Tree on regression tasks with different losses. Red curve: TRBoost; Purple line: GBDT



tive function is smaller. However, while their losses are quite different, the AUC results of these methods are closer. This is because the AUC score does not depend on the probability, the rank of the prediction is more important. For regression tasks, $h = 2$ shrinks the predicted value $\frac{-vg}{h}$, hence the second-order algorithm converges slowly. In Fig. 4(b), we can see that XGBoost is indeed the slowest converge because $v < 1$ further narrows its predictions. LightGBM, being optimized based on XGBoost, is faster in comparison. TRBoost has similar convergence to GBDT because a suitable μ can make $|\frac{-g}{h+\mu}| \approx |-vg| > |\frac{-vg}{h}|$.

Based on the above discussion, we claim that gradient determines the targets and hessian can further improve the results. The learning rate is to reduce the size of the predicted values and find a better solution. However, combined with the expression of TRBoost, the learning rate seems unnecessary since we can always decrease the predicted value by shrinking the target value. Hence, $\frac{-g}{h+\mu}$ is a general target value that combines the advantages of both $-g$ and $\frac{-g}{h}$ and the adaptive adjustment mechanism of μ makes it more flexible.

We further verify the assertion by conducting some experiments with TRBoost-Tree. We take the regression task as an example and apply the Concrete data for the test. For all

results, the only difference is the choice of B_j^{t-1} and α^t in (17). $B_j^{t-1} = 0$ means that we use the first-order approximation to the objective.

From Fig. 5, we can see that when $\alpha = 1$, the second-order approximation is better than the first-order method in spite of convergence and results. That is because when $B_j^{t-1} = 0$, a large leaf value C_j^t leads to a sharp decline in the training loss and causes overfitting. If $B_j^{t-1} > 0$, it shrinks the leaf value and plays a regular role. When $\alpha = 10$, the shrinkage effect of B_j^{t-1} disappears hence both approximations do nice jobs, and the results are better than those with $\alpha = 1$. Combining the above discussions with $\lim_{C \rightarrow 0} \frac{\frac{1}{2}BC^2 + GC}{GC} = 1$, we conclude that the second-order approximation is rewarding only when α is small, i.e. the radius of trust region is large.

In numerical optimization, small steps tend to find better points but require more iterations. On the contrary, large steps can make the objective converge faster, but they may miss the optimal point. Hence in practice, we suggest using big α which means small steps to find a better model. At this time, the performance of algorithms does not rely much on the order of expansion of the objective. This also proves the generality of our method.

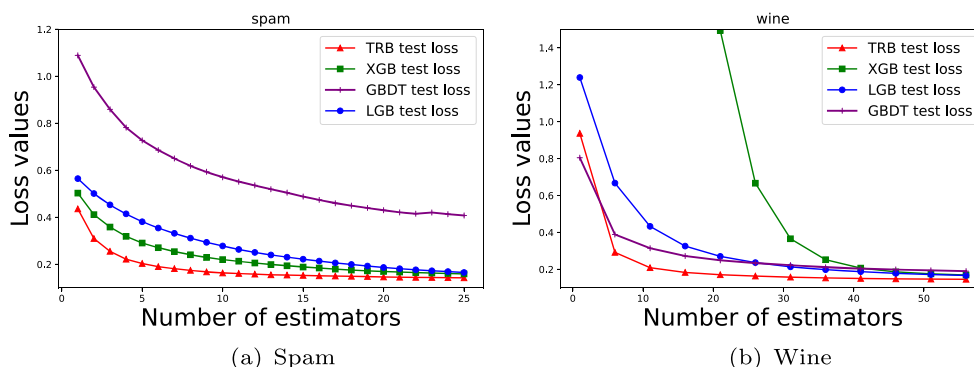


Fig. 4 Test loss curves for GBDT, LightGBM, XGBoost, and TRBoost-Tree on different classification tasks. Red curve: TRBoost; Green curve: XGBoost; Blue curve: LightGBM; Purple line: GBDT

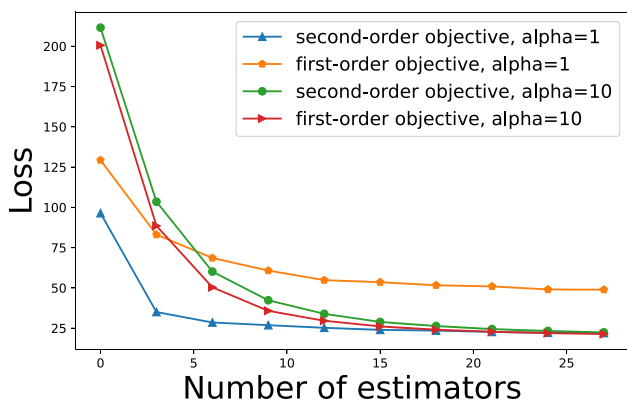
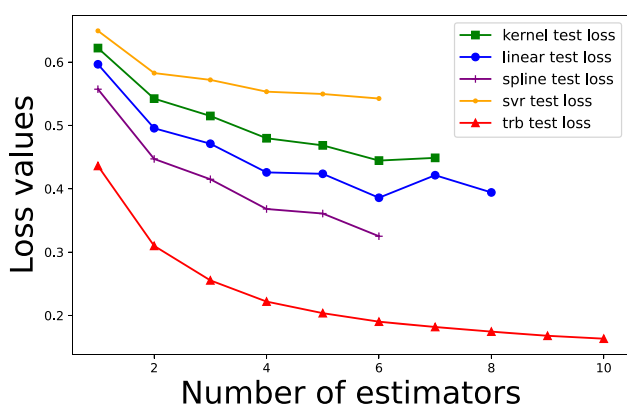


Fig. 5 Loss curves of Taylor expansion of different orders. The orange line represents the first-order approximation result of $\alpha = 1$, the blue line represents the second-order approximation result of $\alpha = 1$, the red line represents the first-order approximation result of $\alpha = 10$, and the green line represents the second-order approximation result of $\alpha = 10$

5.2 Impact of different learners

To delve into the influence of the initial learner choice, we conduct experiments using various models as base learners, including SVR, Gaussian Process Regression, Linear Regression, Spline Regression, and Kernel Ridge Regression. Figure 6 displays the convergence curves of these different learners, where the stopping criterion is determined based on the validation set. The results demonstrate that all the learners can achieve convergence, albeit at varying rates. Notably, the tree method stands out as the most effective option. The discrepancy in performance can be attributed to the discrete nature of tabular datasets and the non-smooth characteristics of the target function during regression iter-



(a) Test Loss

Fig. 6 Test loss curves for different learners on Spam dataset. Green curve: Kernel Ridge; Purple curve: Spline; Orange curve: SVR; Blue line: Linear; Red line: Tree

ation. In contrast, other models, being continuous, tend to favor excessively smooth solutions, potentially hindering their performance in regression tasks [18].

6 Conclusions and future work

In conclusion, TRBoost revolutionizes gradient boosting machines by harnessing the power of the Trust-region method, effectively addressing the challenges of generality and performance head-on. With its remarkable ability to handle a wide range of commonly used losses, maintain second-order superiority, and seamlessly integrate an adaptive radius mechanism, TRBoost emerges as an unstoppable force in the realm of machine learning. We have both theoretically and numerically demonstrated that our algorithm exhibits faster convergence when dealing with strictly convex losses, surpassing first-order approaches and achieving comparable performance to second-order algorithms. Even when faced with a non-positive definite Hessian, TRBoost continues to deliver outstanding results, showcasing its resilience and adaptability. Moreover, through meticulous analysis and a series of carefully designed experiments, we have confirmed that our approach remains unaffected by the degree of expansion of the loss functions. These compelling findings further solidify TRBoost’s position as a competitive and versatile choice for tackling a wide array of data science tasks.

The path forward is brimming with exciting possibilities for further exploration. Firstly, the current implementation of TRBoost serves as a solid foundation, yet there is ample room for improvement. By leveraging more advanced technologies [34] and harnessing the capabilities of superior learners [8, 22, 27], we can unlock even greater performance potential. Additionally, given the convex nature of commonly used loss functions, we can leverage this property to design models that not only converge faster but also uphold their accuracy. This opens up the opportunity to employ fewer estimators, paving the way for more potent models such as neural networks to serve as the base learners. While this endeavor is meaningful, it does come with its own set of challenges, which makes it all the more intriguing and rewarding.

Author Contributions **Jiaqi Luo**: Methodology, Software, Writing—original draft, Writing – review & editing. **Zihao Wei**: Software, Writing – original draft. **Junkai Man**: Software, Writing – original draft. **Shixin Xu**: Conceptualization, Methodology, Writing – original draft, Writing – review & editing.

Funding This work was partially supported by the National Natural Science Foundation of China no. 12071190.

Code Availability All the datasets and codes are made publicly available on GitHub at <https://github.com/Luojiaqimath/TRBoost>.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A: Proof of theorem 2

A.1 One-instance example

Denote the loss at the current iteration by $l = l^t(y, F)$ and that at the next iteration by $l^+ = l^{t+1}(y, F + f)$. Suppose the steps of gradient descent GBMs, Newton’s GBMs, and TRBoost, are $-vg$, $\frac{-vg}{h}$, and $\frac{-g}{h+\mu}$, respectively. v is the learning rate and is usually less than 1 to avoid overfitting.

For regression tasks with MAE loss, since the Hessian $h = 0$, we have $l^+ = l + gf$. Let $\mu = \frac{1}{v}$, then TRBoost degenerate into gradient descent and has the same convergence as the gradient descent algorithm, which is $O(\frac{1}{T})$.

Proof Since $g = 1$ or -1 , substituting $f = -vg$ to $l^+ = l + gf$, we have

$$l^+ = l - vg^2 = l - v < l. \tag{A1}$$

Because l is monotonically decreasing and $l > 0$, v is a constant, thus there exists a constant $0 < c < 1$ such that

$$c(l^t)^2 < v, \forall t. \tag{A2}$$

Hence we have

$$l^+ = l - v < l - cl^2. \tag{A3}$$

According to the Theorem 3 in Appendix B, the $O(\frac{1}{T})$ convergence rate is obtained. \square

For MSE loss, it is a quadratic function. Hence just let $v = 1$ and $\mu = 0$, then follow the standard argument in numerical optimization book [26], both Newton’s method and trust-region method have the **quadratic convergence**.

We now discuss the convergence when the Hessian is not constant. We prove that the trust-region method has a **linear convergence rate** like Newton’s method with fewer constraints.

Proof By the Mean Value Theorem, we have

$$l^+ = l + gf + \frac{1}{2}h_\xi f^2, \tag{A4}$$

$\xi \in [0, f]$ and h_ξ denotes the Hessian at ξ . Substituting $f = \frac{-g}{h+\mu}$ to (A4), we have

$$l^+ = l - \frac{g^2}{h + \mu} + \frac{1}{2}h_\xi \frac{g^2}{(h + \mu)^2}. \tag{A5}$$

In paper [31], it needs $f \geq 0$ to ensure $\frac{h_\xi}{h} \leq 1$. But in our method, we do not need this constraint. Since the Hessian $h = 4p(1 - p)$ (Definition 1) is bounded by the predicted probability p and a proper μ (for example $\mu = 1$) can always make $\frac{h_\xi}{h+\mu} \leq 1$.

By applying Theorem 7 we have

$$\begin{aligned} l^+ &= l - \frac{g^2}{h + \mu} + \frac{1}{2} \frac{h_\xi}{h + \mu} \frac{g^2}{h + \mu} \\ &\leq l - \frac{1}{2} \frac{g^2}{h + \mu} \\ &= l - \frac{1}{2} \frac{h}{h + \mu} \frac{g^2}{h} \\ &\leq l - \frac{1}{2} \frac{h}{h + \mu} l \\ &= \left(1 - \frac{1}{2} \frac{h}{h + \mu}\right) l. \end{aligned} \tag{A6}$$

For any μ that satisfies the inequality $\frac{h_\xi}{h+\mu} \leq 1$ and the Assumption 2, $0 < (1 - \frac{1}{2} \frac{h}{h+\mu}) < 1$ always holds, hence we obtain the **linear convergence** according to Theorem 4. \square

A.2 Tree model case

We omit the proof for regression tasks with different loss functions since it is the same as the One-instance case.

Proof Define the total loss $L(f) = \sum_{i=1}^n l(y_i, F_i + f_i)$. Write it in matrix form and apply the Mean Value Theorem, we have

$$L(f) = L(0) + \mathbf{g}^T \mathbf{f} + \frac{1}{2} \mathbf{f}^T \mathbf{H}_\xi \mathbf{f}, \tag{A7}$$

where $\mathbf{g} = (g_1, \dots, g_n)^T$, $\mathbf{H} = \text{diag}(h_1, \dots, h_n)$ denote the instance wise gradient and Hessian, respectively. In a decision tree, instances falling into the same leaf must share a common fitted value. We can thus write down $\mathbf{f} = \mathbf{V}\mathbf{s}$, where $\mathbf{s} = (s_1, \dots, s_j, \dots, s_J)^T \in \mathbb{R}^J$ are the fitted values on the J leaves and $\mathbf{V} \in \mathbb{R}^{n \times J}$ is a projection matrix $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_j, \dots, \mathbf{v}_J]$, where $\mathbf{v}_{j,i} = 1$ if the j th leaf contains the i th instance and 0 otherwise. Substituting $\mathbf{f} = \mathbf{V}\mathbf{s}$ to $L(f)$ and reload the notation $L(\cdot)$ for \mathbf{s} we have

$$L(\mathbf{s}) = L(0) + (\mathbf{g}^T \mathbf{V}) \mathbf{s} + \frac{1}{2} \mathbf{s}^T (\mathbf{V}^T \mathbf{H}_\xi \mathbf{V}) \mathbf{s}. \tag{A8}$$

Denote $L(\mathbf{s})$ by L^+ and $L(0)$ by L and replace \mathbf{f} with the trust-region step

$$\begin{aligned} \mathbf{f} &= -\mathbf{V}\mathbf{s} \\ &= -\mathbf{V}(\mathbf{V}^T\mathbf{H}\mathbf{V} + \mu\mathbf{I})^{-1}\mathbf{V}^T\mathbf{g}, \end{aligned} \tag{A9}$$

then we have

$$\begin{aligned} L^+ &= L - (\mathbf{V}^T\mathbf{g})^T\hat{\mathbf{H}}^{-1}\mathbf{V}^T\mathbf{g} \\ &\quad + \frac{1}{2}(\mathbf{V}^T\mathbf{g})^T\hat{\mathbf{H}}^{-1}(\mathbf{V}^T\mathbf{H}_\xi\mathbf{V})\hat{\mathbf{H}}^{-1}\mathbf{V}^T\mathbf{g}, \end{aligned} \tag{A10}$$

where $\hat{\mathbf{H}} = \mathbf{V}^T\mathbf{H}\mathbf{V} + \mu\mathbf{I}$, \mathbf{I} is the identity matrix.

Using the Lemmas 4, 5, and Corollary 1 in the Appendix, we obtain the following inequality

$$\begin{aligned} L^+ &\leq L - (\mathbf{V}^T\mathbf{g})^T\hat{\mathbf{H}}^{-1}\mathbf{V}^T\mathbf{g} \\ &\quad + \frac{\tau}{2}(\mathbf{V}^T\mathbf{g})^T\hat{\mathbf{H}}^{-1}\mathbf{V}^T\mathbf{g} \quad (\text{Lemma 4}) \\ &= L - (1 - \frac{\tau}{2})(\mathbf{V}^T\mathbf{g})^T\hat{\mathbf{H}}^{-1}\mathbf{V}^T\mathbf{g} \\ &\leq L - \lambda\gamma_*^2(1 - \frac{\tau}{2})\mathbf{g}^T\mathbf{H}^{-1}\mathbf{g} \quad (\text{Lemma 5}) \\ &\leq L - \lambda\gamma_*^2(1 - \frac{\tau}{2})L \quad (\text{Corollary 1}) \\ &= (1 - \lambda\gamma_*^2(1 - \frac{\tau}{2}))L. \end{aligned} \tag{A11}$$

Since τ and γ_* are constants, $0 < \lambda \leq \frac{h^{min}}{n(h^{max} + \mu)}$, a proper λ can make $0 < \lambda\gamma_*^2(1 - \frac{\tau}{2}) < 1$. According to the Theorem 4 in the Appendix, the linear rate is achieved. \square

Appendix B: Related theory

B.1 Existing theoretical results

In this subsection, we introduce some related results proposed in the paper [31].

Definition 1 [Logistic loss, gradient, Hessian] The instance wise loss $l(\cdot, \cdot)$ is defined as

$$l(y, F) = y\log(\frac{1}{p}) + (1 - y)\log(\frac{1}{1 - p}), \tag{B12}$$

where $y \in \{0, 1\}$ and the probability estimate $p \in [0, 1]$ is computed by a sigmoid-like function on F

$$p = \psi(F) = \frac{e^F}{e^F + e^{-F}}. \tag{B13}$$

And the gradient and Hessian of $l(\cdot, F)$ are given as

$$g(F) = 2(p - y), h(F) = 4p(1 - p). \tag{B14}$$

Assumption 1 [Clapping on p_i] In order to avoid the numerical stability problems, we do a clapping on the output probability $p_i, i = 1, \dots, n$:

$$p_i = \begin{cases} 1 - \rho, & (y_i = 0) \wedge (p_i > 1 - \rho) \\ \rho, & (y_i = 1) \wedge (p_i < \rho) \\ p_i, & \text{otherwise} \end{cases}$$

for some small constant ρ .

Theorem 3 [Recurrence to $O(\frac{1}{T})$ rate] If a sequence $\epsilon_0 \geq \dots \geq \epsilon_{t-1} \geq \epsilon_t \geq \dots \geq \epsilon_T > 0$ has the recurrence relation $\epsilon_t \leq \epsilon_{t-1} - c\epsilon_{t-1}^2$ for a small constant $c > 0$, then we have $\frac{1}{T}$ convergence rate: $\epsilon_T \leq \frac{\epsilon_0}{1 + c\epsilon_0 T}$.

Theorem 4 [Recurrence to linear rate] If a sequence $\epsilon_0 \geq \dots \geq \epsilon_{t-1} \geq \epsilon_t \geq \dots \geq \epsilon_T > 0$ has the recurrence relation $\epsilon_t \leq c\epsilon_{t-1}$ for a small constant $0 < c < 1$, then we have linear convergence rate: $\epsilon_T \leq \epsilon_0 c^T$.

Theorem 5 [Bounded Newton Step] With the value clapping on $p_i, i = 1, \dots, n$, the Newton step $-\frac{\bar{g}_j}{\bar{h}_j}$ is bounded such that $|\bar{g}_j/\bar{h}_j| \leq 1/(2\rho)$. Here $\bar{g}_j = \sum_{i \in I_j} g_i$ and $\bar{h}_j = \sum_{i \in I_j} h_i$ denote the sum of gradient and Hessian in j th tree node.

Theorem 6 [Convergence rate] For GBMs, it has $O(\frac{1}{T})$ rate when using gradient descent, while a linear rate is achieved when using Newton descent.

Theorem 7 [Comparison] Let g, h , and l be the shorthand for gradient, Hessian, and loss, respectively. Then $\forall p$ (and thus $\forall F$), the inequality $\frac{g^2}{h} \geq l$ always holds.

Corollary 1 Connect Newton's objective reduction and loss reduction using Theorem 7, we have

$$\mathbf{g}^T\mathbf{H}^{-1}\mathbf{g} \geq L. \tag{B15}$$

Lemma 1 [Weak Learnability I] For a set of n instances with labels $\{0, 1\}^n$ and non-negative weights $\{w_1, \dots, w_n\}$, there exists a J -leaf classification tree (i.e., outputting $\{0, 1\}$ at each leaf) such that the weighted error rate at each leaf is strictly less than $\frac{1}{2}$ by at least $\delta > 0$.

Lemma 2 [Weak Learnability II] Let $\mathbf{g} = (g_1, \dots, g_n)^T$, $\mathbf{H} = \text{diag}(h_1, \dots, h_n)$. If the weak Learnability Assumption 1 holds, then there exists a J -leaf regression tree whose projection matrix $\mathbf{V} \in \mathbb{R}^{n \times J}$ satisfies

$$(\mathbf{V}^T\mathbf{g})^T(\mathbf{V}^T\mathbf{H}\mathbf{V})(\mathbf{V}^T\mathbf{g}) \geq \gamma_*^2\mathbf{g}^T\mathbf{H}^{-1}\mathbf{g}, \tag{B16}$$

where $\gamma_*^2 = \frac{4\delta^2\rho}{n}$.

Lemma 3 [Change of Hessian] For the current $F \in \mathbb{R}$ and a step $f \in \mathbb{R}$, the change of the Hessian can be bounded in terms of step size $|f|$:

$$\frac{h(F + f)}{h(f)} \leq e^{2|f|}. \tag{B17}$$

B.2 New theoretical results proposed in this paper

Assumption 2 [Boundedness of μ] In the Algorithm 2, we can see that μ^t is monotonically increasing, and it will become infinite as the number of iterations increase, which makes the step \mathbf{p}_k go to 0. In order to avoid this situation, in practice we assume μ^t is bounded, i.e., $\mu^t \in [\mu^0, \mu^{\max}]$, $\forall t$.

Theorem 8 [Boundedness of Hessian] Since μ and Hessian h are bounded, there exists a constant λ that satisfies

$$\frac{1}{\sum_{j \in I_j} (h_j + \mu)} \geq \lambda \frac{1}{\sum_{j \in I_j} h_j}, \forall j = 1, \dots, J. \tag{B18}$$

Proof Let h^{\max} and h^{\min} be the maximum and minimum values of the Hessian respectively. The minimum value on the left side of the inequality is $\frac{1}{n(h^{\max} + \mu)}$, and the maximum value on the right side is $\frac{1}{h^{\min}}$. Just let

$$\lambda \leq \frac{h^{\min}}{n(h^{\max} + \mu)}, \tag{19}$$

then the inequality always holds. □

Lemma 4 [Node wise Hessian] Based on Assumption 1, we have

$$(\mathbf{V}^T \mathbf{H}_\xi \mathbf{V}) \hat{\mathbf{H}}^{-1} \leq \tau \mathbf{I}. \tag{B20}$$

Proof For the j th leaf ($j = 1, \dots, J$), the node wise Hessian before update is

$$\bar{h}_j = \sum_{i \in I_j} (h(F_i) + \mu) \tag{B21}$$

while the mean value is

$$\bar{h}_{\xi,j} = \sum_{i \in I_j} h(F_i + \eta_j), \tag{B22}$$

where η_j lies between 0 and the trust-region step $\frac{\sum_{i \in I_j} g(F_i)}{\sum_{i \in I_j} (h(F_i) + \mu)}$. Applying Lemma 3, we have

$$\begin{aligned} \frac{\bar{h}_{\xi,j}}{\bar{h}_j} &= \frac{\sum_{i \in I_j} h(F_i + \eta_j)}{\sum_{i \in I_j} (h(F_i) + \mu)} \\ &\leq \frac{\sum_{i \in I_j} h(F_i + \eta_j)}{\sum_{i \in I_j} h(F_i)} \\ &\leq \frac{\sum_{i \in I_j} h(F_i) \cdot e^{2\eta_j}}{\sum_{i \in I_j} h(F_i)} \\ &= \exp^{2\eta_j}. \end{aligned} \tag{B23}$$

According to Theorem 5, we have

$$|\eta_j| \leq \frac{\sum_{i \in I_j} g(F_i)}{\sum_{i \in I_j} (h(F_i) + \mu)} \leq \frac{\sum_{i \in I_j} g(F_i)}{\sum_{i \in I_j} h(F_i)} \leq \frac{1}{2\rho}. \tag{B24}$$

Rewriting the matrix form, we have

$$(\mathbf{V}^T \mathbf{H}_\xi \mathbf{V}) \hat{\mathbf{H}}^{-1} \leq \tau \mathbf{I}, \tag{B25}$$

where $\tau = e^{\frac{1}{2\rho}}$. □

Lemma 5 [Weak Learnability III] Let $\mathbf{g} = (g_1, \dots, g_n)^T$, $\mathbf{H} = \text{diag}(h_1, \dots, h_n)$, $\hat{\mathbf{H}} = \mathbf{V}^T \mathbf{H} \mathbf{V} + \mu \mathbf{I}$. If the weak Learnability Assumption 2 holds, then there exists a J -leaf regression tree whose projection matrix $\mathbf{V} \in \mathbb{R}^{n \times J}$ satisfies

$$(\mathbf{V}^T \mathbf{g})^T \hat{\mathbf{H}}^{-1} (\mathbf{V}^T \mathbf{g}) \geq \gamma \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} \tag{B26}$$

for some constant $\gamma > 0$.

Proof For each leaf j , according to Lemma 2, we have

$$\frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i} \geq \gamma_*^2 \sum_{i \in I_j} \frac{g_i^2}{h_i} \tag{B27}$$

By Theorem 8, for some λ , we obtain

$$\frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} (h_i + \mu)} \geq \lambda \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i} \geq \lambda \gamma_*^2 \sum_{i \in I_j} \frac{g_i^2}{h_i}. \tag{B28}$$

Rewriting in the matrix form, we get the final result

$$(\mathbf{V}^T \mathbf{g})^T \hat{\mathbf{H}}^{-1} (\mathbf{V}^T \mathbf{g}) \geq \lambda \gamma_*^2 \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g}. \tag{B29}$$

□

References

1. Ait Hammou B, Ait Lahcen A, Mouline S (2019) A distributed group recommendation system based on extreme gradient boosting and big data technologies. *Appl Intell* 49:4128–4149
2. Biau G, Cadre B, Rouvière L (2019) Accelerated gradient boosting. *Mach Learn* 108(6):971–992
3. Bickel PJ, Ritov Y, Zakai A et al (2006) Some theory for generalized boosting algorithms. *Journal of Machine Learning Research* 7(5)
4. Borisov V, Leemann T, Seßler K et al (2022) Deep neural networks and tabular data: a survey. *IEEE Transactions on Neural Networks and Learning Systems*
5. Breiman L, Friedman JH, Olshen RA et al (2017) *Classification and regression trees*. Routledge
6. Buehlmann P (2006) Boosting for high-dimensional linear models. *The Annals of Statistics* 34(2):559–583
7. Bühlmann P, Yu B (2003) Boosting with the l_2 loss: regression and classification. *J Am Stat Assoc* 98(462):324–339
8. Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp 785–794
9. Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: *Proceedings of the 23rd international conference on Machine learning*, pp 233–240
10. Dua D, Graff C (2017) UCI machine learning repository. <http://archive.ics.uci.edu/ml>
11. Duan T, Anand A, Ding DY et al (2020) Ngboost: Natural gradient boosting for probabilistic prediction. In: *International conference on machine learning*, PMLR, pp 2690–2700
12. Frénay B, Verleysen M (2013) Classification in the presence of label noise: a survey. *IEEE Trans Neural Netw Learn Syst* 25(5):845–869
13. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 55(1):119–139
14. Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics* 28(2):337–407
15. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp 1189–1232
16. Friedman JH (2002) Stochastic gradient boosting. *Comput Stat Data Anal* 38(4):367–378
17. Ghosh A, Kumar H, Sastry PS (2017) Robust loss functions under label noise for deep neural networks. In: *Proceedings of the AAAI conference on artificial intelligence*
18. Grinsztajn L, Oyallon E, Varoquaux G (2022) Why do tree-based models still outperform deep learning on typical tabular data. *Adv Neural Info Process Syst* 35:507–520
19. Grubb A, Bagnell JA (2011) Generalized boosting algorithms for convex optimization. In: *Proceedings of the 28th international conference on machine learning*, pp 1209–1216
20. Hastie T, Tibshirani R, Friedman JH et al (2009) *Elements of statistical learning: data mining, inference, and prediction*, vol 2. Springer, New York
21. Huang H, Jia R, Shi X et al (2021) Feature selection and hyper parameters optimization for short-term wind power forecast. *Appl Intel* 1–19
22. Ke G, Meng Q, Finley T et al (2017) Lightgbm: a highly efficient gradient boosting decision tree. *Adv Neural Info Process Syst* 30
23. Liu Y, Guo H (2020) Peer loss functions: learning from noisy labels without knowing noise rates. In: *International conference on machine learning*, PMLR, pp 6226–6236
24. Lu H, Karimireddy SP, Ponomareva N et al (2020) Accelerating gradient boosting machines. In: *International conference on artificial intelligence and statistics*, PMLR, pp 516–526
25. Natekin A, Knoll A (2013) Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics* 7:21
26. Nocedal J, Wright SJ (1999) *Numerical optimization*. Springer, New York
27. Prokhorenkova L, Gusev G, Vorobev A et al (2018) Catboost: unbiased boosting with categorical features. *Adv Neural Info Process Syst* 31
28. Schmid M, Hothorn T (2008) Boosting additive models using component-wise p-splines. *Comput Stat Data Anal* 53(2):298–311
29. Shrivastav LK, Jha SK (2021) A gradient boosting machine learning approach in modeling the impact of temperature and humidity on the transmission rate of covid-19 in india. *Appl Intell* 51:2727–2739
30. Sigrist F (2021) Gradient and newton boosting for classification and regression. *Expert Syst Appl* 167:114080
31. Sun P, Zhang T, Zhou J (2014) A convergence rate analysis for logitboost, mart and their variant. In: *International conference on machine learning*, PMLR, pp 1251–1259
32. Telgarsky M, Singer Y (2012) A primal-dual convergence analysis of boosting. *J Mach Learn Res* 13(3)
33. Vanschoren J, Van Rijn JN, Bischl B et al (2014) Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter* 15(2):49–60
34. Verbraeken J, Wolting M, Katzy J et al (2020) A survey on distributed machine learning. *Acm Comput Surv* 53(2):1–33
35. Wang Y, Ma X, Chen Z et al (2019) Symmetric cross entropy for robust learning with noisy labels. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 322–330
36. Yuan Y (2000) A review of trust region algorithms for optimization. In: *ICIAM*, pp 271–282
37. Zhang Z, Sabuncu M (2018) Generalized cross entropy loss for training deep neural networks with noisy labels. *Adv Neural Info Process Syst* 31

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Jiaqi Luo is a research scientist currently working at the Data Science Research Center at Duke Kunshan University. He earned both his B.S. degree in Mathematics and his Ph.D. degree in Computational Mathematics from Soochow University, graduating in 2015 and 2020, respectively. His research interests include machine learning, scientific computing, and nonlinear optimization.



Zihao Wei holds a B.S. degree in Data Science from Duke Kunshan University and is currently embarking on his journey towards an MEng degree in Financial Technology at Duke University. His research lies at the intersection of algorithms and machine learning, where he focuses on developing data-driven optimization frameworks and novel machine learning methodologies with a strong emphasis on their real-world applications in the financial domain.



Junkai Man holds a bachelor's degree in Data Science from Duke Kunshan University and is pursuing a master's degree in Computer Science at Brown University. His research mainly focuses on machine learning applications, software engineering, and multimedia.



Shixin Xu is an Assistant Professor of Mathematics at Duke Kunshan University. His research interests are machine learning and data-driven model for diseases and multiscale modeling of complex fluids. Xu has a B.Sc. in mathematics (honors) from Ocean University of China and a Ph.D. in mathematics from the University of Science and Technology China. From 2013 to 2017, he held postdoctoral positions at the National University of Singapore, the University of Notre Dame, the University of California, Riverside, and the Fields Institute for Research in Mathematical Sciences, Canada.