Masked Aggregation Learning for Enhancing Distributed Gradient Boosting Decision Trees

Yuting Zha[1,2], Chao Lin[1(✉)], Xinyi Huang[3], and Dugang Liu[4]

[1] College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China
[2] Fujian Provincial Key Laboratory of Network Security and Cryptology
yutingzha@163.com, linchao91@fjnu.edu.cn
[3] College of Cyber Security, Jinan University, Guangzhou 510632, China
xyhuang81@gmail.com
[4] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
dugang.ldg@gmail.com

Abstract. Federated Gradient Boosting Decision Trees (GBDT) have gained popularity for enabling collaborative, privacy-preserving training across multiple distributed participants. However, existing federated GBDT frameworks require extensive communications for the creation of each subtree, as each participant trains data locally. These methods often involve complex secure multi-party computation or homomorphic encryption techniques that hinder training efficiency and suffer from low model accuracy in uneven data distributions. To address these issues, we propose a Masked Aggregation Learning (MAL) framework for federated GBDT. MAL combines distributed data preprocessing with centralized training, allowing participants to securely mask their data and share it with a central server for centralized training. Our approach includes constructing decision trees using histograms by discretizing continuous feature values into distinct buckets. During data preprocessing, we introduce a Secret Extremes Bucket Construction (SE-Bucket) method based on order-revealing encryption for unified bucketing and feature value obfuscation. Additionally, we propose an Erasable Label Mask generation algorithm that ensures label privacy while the masks do not impact model accuracy. MAL reduces the communication rounds from a linear relationship with the number of GBDT subtrees to a constant two rounds, independent of the number of subtrees, and maintains model accuracy regardless of how the data is distributed among the participants. Experimental results show that our method, MAL, achieves accuracy comparable to centralized training while being 500 to 600 times faster than state-of-the-art federated decision tree training solutions.

Keywords: Gradient Boosting Decision Trees · Federated Learning · Local Differential Privacy · Masked Aggregation Learning.

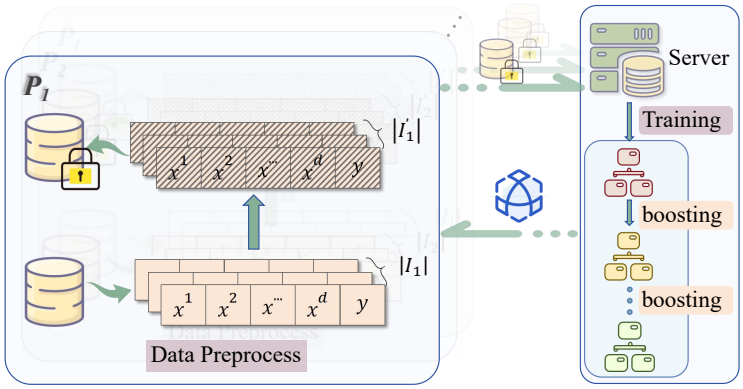


Fig. 1: The Framework of MAL.

1 Introduction

Gradient Boosting Decision Tree (GBDT) is a powerful ensemble learning method that has gained prominence with the growth of data and the demand for advanced analytics. High-quality data is critical to GBDT's performance and often originates from multiple sources in practice. However, traditional GBDT implementations rely on centralized data processing, which is impractical when dealing with multiple data sources due to privacy regulations like General Data Protection Regulation [19]. While GBDTs are highly effective, their application in protecting the privacy of the data from different sources poses challenges.

Recently, numerous studies have focus on distributed GBDT [7, 5, 21, 11, 22, 20], where federated learning [15, 14] has been widely appiled due to the ablility of collaborative training across distributed participants while protecting data privacy. Existing solutions allow users to train models locally and achieve multi-party collaborative training by sharing model parameters or gradients rather than raw data, but they still face the following insufficiencies.

1) **High Computation Costs.** Schemes [7, 5, 21] employ secure multi-party computation and homomorphic encryption to ensure privacy, but the substantial cryptographic computations impose nearly prohibitive computational burdens. 2) **Unstable Model Accuracy.** SimFL [11] uses locality-sensitive hashing to aggregate gradients from similar data across parties and create new trees with weighted gradient boosting. However, this approach still exhibits significant errors when dealing with uneven data distribution. 3) **Extensive communications.** Additionally, most federated GBDT methods [22, 20] keep the data on the participants, and the server needs to communicate with the participants continuously during the training process (to obtain the best split points, aggregate gradients, etc.). This leads to the training process of each subtree requiring multiple communications.

To address the above challenges, this paper employs a dual-mode approach combining distributed data preprocessing and centralized training (see Fig. 1).

Each participant processes their own data and then sends it to the server for centralized training. In this way, we can get rid of the multiple communications of training federated GBDT. However, the trade-off between privacy and performance remains a challenge. Thus, our main contributions are as follows.

- We present a new framework (Masked Aggregation Learning, MAL) for collaborative GBDT training with data privacy protection in a distributed setting. It requires participants to process the features and labels of their data locally and send them to the central server. The server, upon receiving the processed data, can only use it for model training without obtaining the actual values of the data. Moreover, participants only need to send the data once before the model training begins.
- To obscure feature values and create a unified binning structure for uneven data nodes, we propose a new distributed bucketing method called Secret Extremes Bucket Construction (SE-Bucket). This method incurs minimal communication costs.
- We also propose an Erasable Label Mask generation algorithm to mask label values and ensure privacy. These masks are countervailed during the final centralized aggregation, which preserves model accuracy.
- Our experimental results show that MAL achieves the same level of accuracy as centralized training. With 32 participants, MAL is 500 to 600 times faster than the state-of-the-art federated solutions.

2 Preliminaries

2.1 Gradient Boosting Decision Trees (GBDTs)

GBDT is an ensemble learning method that builds models sequentially, with the aim of reducing errors by combining the strengths of multiple weak learners, typically decision trees [4]. The working principle of GBDT is boosting, where each new learner attempts to correct the prediction errors made by the previous learner. Formally, for a given dataset with n samples and d features $\mathcal{D} = \{(X_i, y_i)\}(|\mathcal{D}| = n, X_i \in \mathbb{R}^d, y_i \in \mathbb{R})$, GBDT minimizes the following regularized objective function $\tilde{\mathcal{L}} = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right)$, where l is a differentiable convex loss function, $\Omega(f) = \gamma T_l + \frac{1}{2}\lambda\|V\|^2$ is a regularization term to penalize the complexity of the model. γ and λ are hyper-parameters, T_l is the number of leaves, and V is the leaf weight. Each f_k corresponds to a decision tree.

Figure 2 illustrates an example of GBDT. In each tree, the input x is divided into a leaf node based on the intermediate nodes of the decision tree. The prediction result of each tree is the leaf weight of the leaf node where x falls, with the leaf weight being $V(I) = -\frac{\sum_{i \in I} g_i}{\sum_{i \in I} h_i + \lambda}$, where g_i and h_i represent the first-order and second-order gradients of the loss function, respectively. Then the final prediction is the sum of all individual tree predictions.

Each decision tree is created starting from the root node. For each intermediate node, let I_L and I_R be the sets of instances after splitting into the left

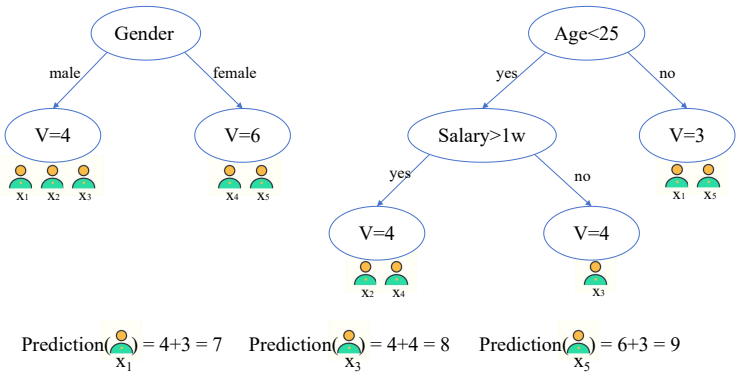


Fig. 2: An Example of GBDT.

and right child nodes, respectively, and $I = I_L + I_R$. The gain after splitting is $\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i+\lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i+\lambda}\right] - \gamma$.

2.2 Histogram Algorithm

In decision trees based on the histogram algorithm, the histogram algorithm divides the values of continuous features into discrete buckets and then computes histograms based on these buckets to perform node splitting. This method can reduce computational load and improve training efficiency. There are two ways to divide buckets: equal-width binning and equal-frequency binning.

Equal-width binning divides the data into several intervals of equal width. For example, when the input value from the range $[min, max]$ is divided into q intervals, the width of each interval is: $width = \frac{max-min}{q}$. Equal-frequency binning divides the data into several intervals with equal frequency, meaning each interval contains the same number of samples.

In LightGBM [8], the equal-width binning histogram algorithm is used to improve training efficiency. Our approach also utilizes the equal-width binning histogram algorithm. We find the maximum and minimum values for each feature and perform binning. Suppose that each feature is divided into q buckets. $G_k = \sum_{i\in B_k} g_i$ and $H_k = \sum_{i\in B_k} h_i$ are calculated for each bucket. The gradient histogram of a feature consists of the G_k and H_k of all buckets. Then, the optimal split point is selected based on the histogram. Let I_L and I_R be the sets of instances after splitting into the left and right child nodes, respectively, with $I = I_L + I_R$. The gain after splitting is

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{B_k\in I_L} G_k)^2}{\sum_{B_k\in I_L} H_k+\lambda} + \frac{(\sum_{B_k\in I_R} G_k)^2}{\sum_{B_k\in I_R} H_k+\lambda} - \frac{(\sum_{B_k\in I} G_k)^2}{\sum_{B_k\in I} H_k+\lambda}\right] - \gamma. \quad (1)$$

The leaf weight is computed using

Table 1: Summary of Notations.

Notation	Description
P_j	the j-th participant
I_j	participant P_j's dataset
s_i^j	the i-th sample in I_j
X_i^j	feature vector of the i-th sample in I_j
y_i^j	label value of the i-th sample in I_j
x^i	value of the i-th feature
d	the number of features
q	the number of buckets for each feature
Q	the total number of buckets
B_k	the k-th bucket or bin
B_k^i	the k-th bucket of the i-th feature
G_k	first-order gradient aggregation of B_k
H_k	second-order gradient aggregation of B_k
N_k	the number of samples in B_k
N_k^j	the number of samples in B_k of I_j
max_j^i	local maximum value of the i-th feature in I_j
min_j^i	local minimum value of the i-th feature in I_j
max^i	global maximum value of the i-th feature
min^i	global minimum value of the i-th feature

$$V(I) = -\frac{\sum_{B_k \in I} G_k}{\sum_{B_k \in I} H_k + \lambda}. \tag{2}$$

2.3 Order-Revealing Encryption (ORE)

ORE [10, 12, 17] is an encryption primitive that allows ciphertexts to be ordered through a computable comparison function without accessing the plaintext itself. The key generation algorithm takes the security parameter λ as input and outputs the system parameter par, the master secret key msk, and the comparison key ck. The encryption algorithm takes msk and a message m as inputs and outputs the ciphertext $c \leftarrow ORE.Enc(msk, m)$. The comparison algorithm, given ck and two ciphertexts c and c', outputs a flag $b \leftarrow ORE.Cmp(ck, c, c')$. Peng et al. [17] proposed a new and efficient ORE scheme, which only requires about 79 milliseconds of time overhead to to compare two 64-bit plaintexts in the cipher setting. In this paper, we use this ORE scheme to enable each participant to obtain the global maximum and minimum values for each feature without exposing the values of each feature in their own dataset.

2.4 Differential Privacy

Differential Privacy [6] is a technique for protecting data privacy that ensures the participation of individual data points in a dataset does not significantly affect

the results of data analysis. In other words, even if any single entry in the dataset is removed or replaced, the analysis results will not show noticeable changes. This effectively prevents malicious attackers from inferring information about a specific individual through database queries. Local Differential Privacy (LDP) is a form of differential privacy that focuses on protecting each participant's data privacy before data collection. Local differential privacy ensures that each individual's data remains private after local processing. Even if an attacker knows all other users' data, they cannot infer an individual's original data by analyzing the processed data.

Definition 1. *(ε-Local Differential Privacy) The function $\mathcal{S}$ is said to provide ε-local differential privacy if, for any two sample x_1 and x_2 that differ by only one element, the function $\mathcal{S}$ satisfies*

$$Pr[\mathcal{S}(x_1) \in y] \leq e^{\varepsilon} \cdot Pr[\mathcal{S}(x_2) \in y], \tag{3}$$

where all $y \subseteq$ Range $(\mathcal{S})$, ε is the privacy budget used to quantify the degree of differential privacy. The smaller ε, the stronger the protection of privacy.

FederBoost [18] introduced a variant of DP to incorporate noise into bucket construction. This method allows for a certain probability that samples intended for the k-th bucket may be assigned to other buckets. Inspired by Tian et al. [18], in this paper, we introduce perturbations to the bucketing results to achieve ε-local differential privacy.

3 Masked Aggregation Learning

We assume that there are l participants $P_1, P_2, ..., P_l$, each holding datasets $I_1, I_2, ..., I_l$, where $I_j = \{(X_i^j, y_i^j)\}$, $(X_i^j \in \mathbb{R}^d, y_i^j \in \mathbb{R}, j \in [1, l])$. We focus on the case of partitioned data, where each party holds datasets with the same feature set. In this work, we do not consider poisoning attacks which can be mitigated using the existing technologies [3, 16, 13]. We assume that each participant is motivated to train a better model, but they also want to peek at the data of others. Each participant holds the same master secret key msk. Additionally, we set up a third-party comparer, which does not hold msk but has the comparison key ck. The participants do not collude with the comparer. Table 1 summarizes the commonly used symbols.

3.1 An Overview of MAL

MAL consists of two phases: preprocessing and training. In the preprocessing phase, to protect feature values from being leaked, we propose Secret Extremes Bucket Construction for feature value obfuscation. To maintain model accuracy while ensuring privacy, we introduce an Erasable Label Mask generation algorithm that ensures the sum of masks added to labels of samples within the same bucket is zero. During the training phase, parties send the obfuscated data to

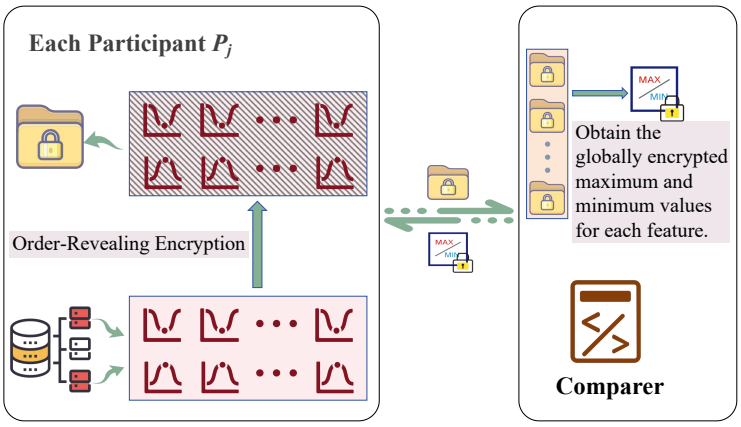


Fig. 3: Obtaining the Global Extreme for Each Feature.

the server, which performs centralized training and then broadcasts the trained model parameters to the other participants. The entire process of MAL only requires one round of communication at each of the two phases.

3.2 The Preprocessing Stage

The primary purpose of the preprocessing stage is to transform the feature values and the label values of each sample, and minimize the loss of precision. As described in the Section 2, each decision tree consists of internal nodes (i.e., split points) and leaf nodes. The split points are selected based on the split gain of the features, which is calculated using Eq. (1) and depends only on G and H. The leaf weights of the decision tree are calculated using Eq. (2), which also depends solely on G and H. Therefore, as long as we ensure that the transformation of feature values and label values does not alter the values of G and H, we can achieve data transformation without impacting model accuracy.

3.3 Secret Extremes Bucket (SE-Bucket)

Algorithm 1 describes the entire process of the SE-Bucket. The MAL uses the histogram algorithm to construct decision trees. To ensure that each party's binning structure is consistent with the global binning structure, we need to obtain the global maximum and minimum values for each feature (lines 2-8). Figure 3 illustrates the process of obtaining the global maximum and minimum values. Each participant P_j finds the maximum value max_j^i and the minimum value min_j^i for each feature f_i locally. P_j encrypts max_j^i and min_j^i using msk (lines 4-5). Each participant sends the encrypted values $c(max_j^i)$ and $c(min_j^i)$ to comparer. Comparer uses ck to compare and obtain the encrypted global extremes $c(max^i)$ and $c(min^i)$, and then returns the results to participants (lines

Algorithm 1: Secret Extremes Bucket

Input: each P_j inputs n_j samples and each sample has all d features $X = \{x^1, x^2, \ldots, x^d\}$

Output: each P_j outputs n_j samples, with each sample having d features $X' = \{x^{1\prime}, x^{2\prime}, \ldots, x^{d\prime}\}$

1 **for** $i = 1 \to d$ **do**
2 **for** $j = 1 \to l$ **do**
3 $max_j^i, min_j^i \leftarrow P_j$;
4 $c(max_j^i) \leftarrow$ ORE.Enc(msk, max_j^i);
5 $c(min_j^i) \leftarrow$ ORE.Enc(msk, min_j^i);
6 P_j sends $c(max_j^i)$ and $c(min_j^i)$ to Comparer;
7 $c(max^i) \leftarrow max\{c(max_1^i), \ldots, c(max_l^i)\}$;
8 $c(min^i) \leftarrow min\{c(min_1^i), \ldots, c(min_l^i)\}$;
9 **for** $j = 1 \to l$ **do**
10 **for** $k = 0 \to q$ **do**
11 $b_k = min^i + \frac{k \times (max^i - min^i)}{q}$;
12 **for** *each sample* **do**
13 **if** $b_{k-1} \leq x^i \leq b_k$ **then**
14 P_j moves it to B_k^i;
15 P_j moves it to another bucket with probability $\frac{q-1}{e^\varepsilon + q - 1}$;
16 **for** $k = 1 \to q$ **do**
17 **if** $X \in I_j$ *belongs to the* B_k^i **then**
18 $x^i = min^i + \frac{(k - 1/2) \times (max^i - min^i)}{q}$;

7-8). Participants decrypt the values using msk to obtain the global extremes max^i and min^i of each feature. Determine the global bucket structure by finding the quantiles based on the global extremes for each feature (lines 10-11).

Based on the above binning results, a sample is assigned to the k-th bucket according to the value of i-feature x^i. Its feature value x^i is then transformed to the median value of the k-th bucket (line 18). The central server only knows the boundaries of each bucket but does not have access to the ground truth data within the buckets. However, it can infer the range of feature values belonging to that bucket (i.e. the bucket boundaries). Therefore, we introduce noise perturbation to the bucketing results. For a sample originally assigned to the k-th bucket, it has a probability $\frac{e^\varepsilon}{e^\varepsilon + q - 1}$ of remaining in the k-th bucket, while the probability of being assigned to each of the remaining $q - 1$ buckets is $\frac{1}{e^\varepsilon + q - 1}$.

3.4 Erasable Label Mask (ELM)

To protect privacy, we add a mask to the label value of each individual sample. The masks are designed to self-cancel within each bucket, thus not affecting the gradient aggregation results (G and H) to maintain the accuracy of the model.

Algorithm 2: Erasable Label Mask Generation

Input: each P_j inputs n_j samples and each sample has all a label y
Output: each P_j inputs n_j samples and each sample has all a label y'

for $j = 1 \to l$ **do**
  `// for each party` P_j
  Set unknowns $X = [x_1, x_2, \ldots, x_{Q+1}]$;
  for $k = 1 \to Q$ **do**
    `// for each bucket` B_k
    $a_1 \leftarrow 0$, $a_2 \leftarrow 0$, $\ldots$, $a_{Q+1} \leftarrow 0$;
    for $i = 1 \to |I_j|$ **do**
      `// for each simple` s_i^j
      Randomly generating a $(Q+1)$-dimensional vector $[a_1^i, a_2^i, \ldots, a_{Q+1}^i]$;
      if s_i^j *belongs to the* B_k **then**
        $a_1 \leftarrow a_1 + a_1^i$, $a_2 \leftarrow a_2 + a_2^i$, $\ldots$, $a_{Q+1} \leftarrow a_{Q+1} + a_{Q+1}^i$;
    Homogeneous linear equation $a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_{Q+1} \cdot x_{Q+1} = 0$;
  Solving the system of homogeneous linear equations;
  for $i = 1 \to |I_j|$ **do**
    $mask_i = a_1^i \cdot x_1 + a_2^i \cdot x_2 + \ldots + a_{Q+1}^i \cdot x_{Q+1}$;
    $y_i' = y_i + mask_i$;

Additionally, compared to other secure aggregation schemes (multi-party mask negotiation) [2], our ELM does not need to consider the issue of other users dropping out due to its self-cancelling mask characteristic. The Erasable Label Mask generation process is detailed in Algorithm 2. Samples can be divided into q buckets based on any feature. Assuming the data has d-dimensional features, there are a total of $Q = q \times d$ buckets. To eliminate the noise interference caused by the mask, it is necessary to ensure that within each bucket satisfies

$$\sum_{i=1}^{N_k} mask_i = 0, \tag{4}$$

where N_k is the number of samples in B_k. Evidently, as long as participants can ensure within each bucket satisfies Eq. (5), then Eq. (4) is guaranteed to hold.

$$\sum_{i=1}^{N_k^j} mask_i = 0. \tag{5}$$

Let there be unknowns $X = [x_1, x_2, \ldots, x_{Q+1}]^\top$. Each sample s_i randomly generates a $(Q+1)$-dimensional vector $A_i = [a_1^i, a_2^i, \ldots, a_{Q+1}^i]^\top$. The mask value is then

$$mask_i = a_1^i \cdot x_1 + a_2^i \cdot x_2 + \ldots + a_{Q+1}^i \cdot x_{Q+1}. \tag{6}$$

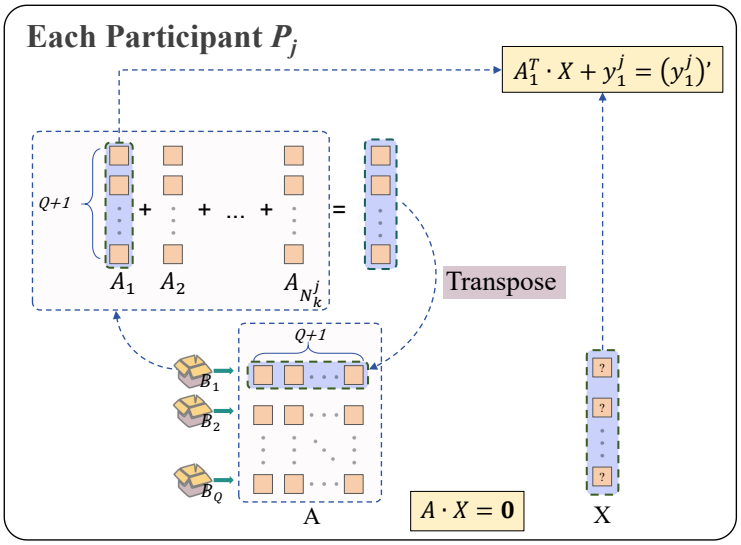


Fig. 4: Erasable Label Mask Generation.

Each participant must ensure that each bucket satisfies Eq. (5). We transform this problem into solving a $Q \times (Q+1)$ homogeneous matrix equation. Therefore, within each bucket satisfies Eq. (7) (lines 3-9).

There are Q homogeneous linear equations in the form of Eq. (7), forming a system of homogeneous linear equations. When the number of unknowns is greater than the number of equations, the system must have non-trivial solutions [9]. Solving this system and substituting the obtained values of X into Eq. (6), we calculate the mask value added to the label of each sample. Finally, the mask is added to the label of each sample (lines 11-13). The entire process of adding a mask to each label value is visualized in Fig. 4.

$$\begin{aligned}\sum_{i=1}^{N_k^j}\left(a_1^i x_1 + \ldots + a_{Q+1}^i x_{Q+1}\right) &= \sum_{i=1}^{N_k^j} a_1^i x_1 + \ldots + \sum_{i=1}^{N_k^j} a_{Q+1}^i x_{Q+1} \\ &= a_1 x_1 + \ldots + a_{Q+1} x_{Q+1} = 0.\end{aligned} \tag{7}$$

3.5 The Training Stage

The entire process of the training phase is shown in algorithm 3. All participants send their transformed data and quantiles for each feature to the central server. After collecting all the data, the central server buckets the data based on the quantiles, constructs a histogram, and trains a GBDT model. The central server selects the optimal split points for each decision tree according to Eq. (1) (lines 5-11). To ensure that the added mask values on the labels cancel each other out

Algorithm 3: The Training Stage

Input: Instance set I'
Output: The GBDT model
All ordinary participants send their transformed datasets to the active participant P_l;
for $t = 1 \to T$ **do**
  Update g and h of all training instances on loss l;
  $gain \leftarrow 0$;
  for $i = 1 \to d$ **do**
    // for each feature
    $G_L \leftarrow 0$, $G_R \leftarrow 0$;
    $G \leftarrow \sum_{k=0}^{q} G_k$, $H \leftarrow \sum_{k=0}^{q} H_k$;
    for $k = 1 \to q$ **do**
      // for each bucket B_k^i
      $G_L \leftarrow G_L + G_i$, $H_L \leftarrow H_L + H_i$;
      $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$;
      $gain \leftarrow max(gain, \frac{G_R^2}{H_R^2+\lambda} + \frac{G_L^2}{H_L^2+\lambda} - \frac{G^2}{H^2+\lambda})$;
  Split with the max gain;
  $V(I) = -\frac{\sum_{i \in I} G_i}{\sum_{i \in I} H_i + \lambda}$;
  Send the split point and leaf weights to the other participants;

and do not affect the leaf values, we set the depth of each decision tree to 1. The central server sends the split points of each tree and the leaf weights of all leaf nodes of each tree to the other participants.

3.6 Security Analysis

Each participant sends the encrypted feature local extremes to the comparer. The comparer receives the ciphertexts but cannot access the plaintext without the master key. Conversely, although each participant has the master key, they cannot access other participants' feature local extremes without the ciphers. This protects the privacy of all parties' feature local extremes.

The feature values is converted to the median of the bucket it belongs to, which obfuscates each other so that they cannot be distinguished. Furthermore, we add noise to the feature values to ensuring local differential privacy.

Corollary 1. *Our feature value obfuscation mechanism satisfies ε-local differential privacy.*

Proof. For any two samples s_1 and s_2, which possess feature vectors X_1 and X_2, respectively, and differ by only one feature value x^j, we have

$$\begin{aligned}\frac{Pr[\mathcal{S}(X_1)=X']}{Pr\left[\mathcal{S}\left(X_2\right)=X'\right]}&=\prod_i \frac{Pr\left[\mathcal{S}\left(x_1^i\right)=x^{i\prime}\right]}{Pr\left[\mathcal{S}\left(x_2^i\right)=x^{i\prime}\right]}\\ =&\frac{Pr\left[\mathcal{S}\left(x_1^j\right)=x^{j\prime}\right]}{Pr\left[\mathcal{S}\left(x_2^j\right)=x^{j\prime}\right]}\leq\frac{e^\varepsilon/(e^\varepsilon+q-1)}{1/(e^\varepsilon+q-1)}=e^\varepsilon,\end{aligned} \tag{8}$$

where $Pr[\mathcal{S}(X)=X']$ denotes the probability that a sample s has its feature value obfuscated to X'.

We mask the label values, ensuring that the sum of the masks within the same bucket equals zero. The masks are determined using a matrix equation, which has infinitely many solutions. This guarantees that for any given output, there are infinitely many possible inputs that produce the same output. Thus, the server cannot obtain the original label values.

4 Implementation and Experiments

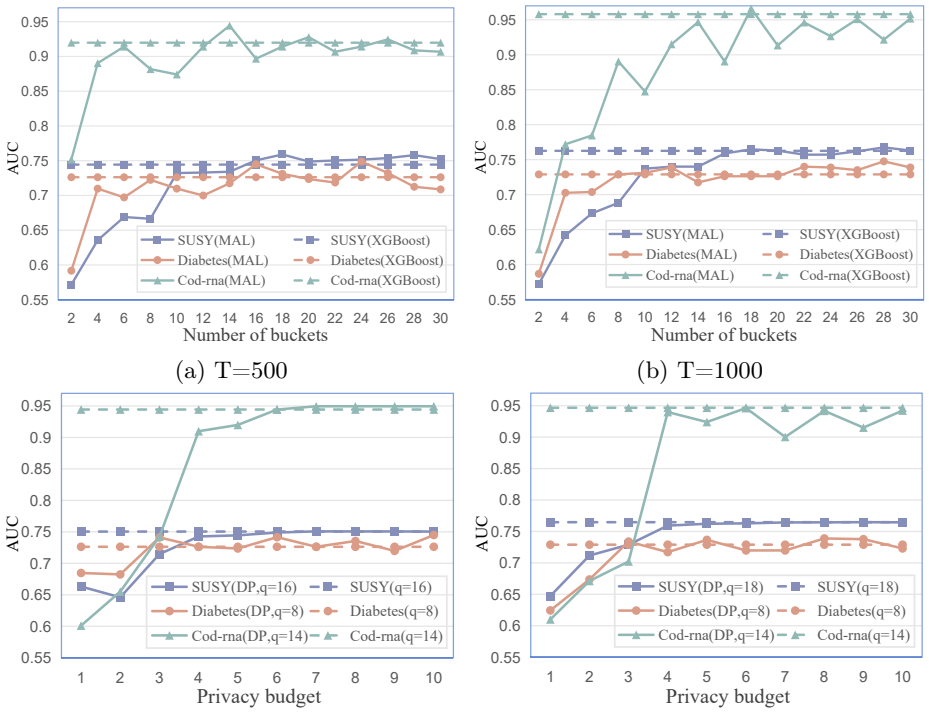
In this section, we demonstrate the effectiveness and efficiency of MAL, and provide experiments on several publicly available datasets from the LIBSVM website[5]. **SUSY**: It contains 3,000,000 samples and 18 features. We randomly selected 289,330 samples. **Diabetes**: It contains 768 samples, each with 8 features. **Cod-rna**: It contains 59,535 training samples and 271,617 test samples, each sample having 8 features.

We implemented MAL based on the XGBoost[6]. We conducted experiments on a machine with an Intel(R) Core(TM) i7-10700 CPU at 2.90GHz and 16GB RAM. For the datasets SUSY and Diabetes, we randomly selected 80% samples for training and the remaining for testing. We evaluate our new framework based on utility and efficiency. Since all participants send disguised data to the central server for centralized training, changing the number of participants does not affect the utility of MAL. Therefore, when evaluating utility, we only consider the impact of different numbers of buckets q and privacy budgets ε on accuracy, with the number of trees T set to 500 and 1000, respectively. When evaluating efficiency, we account for different numbers of participants. All experiments are repeated 10 times, and the average is taken.

4.1 Utility

We use AUC (Area Under the ROC Curve) as the criterion for model utility, which provides a better reflection of the model ability to distinguish between different classes. We first evaluate the impact of different bucket numbers on

[5] https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/
[6] https://github.com/dmlc/xgboost



(a) T=500 (b) T=1000

(c) T=500 (d) T=1000

Fig. 5: Utility of MAL.

the utility of MAL without considering differential privacy. Figures 5a and 5b show how AUC varies with the number of buckets q for $T = 500$ and $T = 1000$, respectively. For SUSY, MAL achieves the best AUC of 75.89% at $q = 18$ and 76.75% at $q = 28$ for $T = 500$ and $T = 1000$, respectively, while the AUC achieved by XGBoost is 74.43% and 76.26%. For Diabetes, MAL reaches the best AUC of 74.49% at $q = 16$ and 74.76% at $q = 28$, while the AUC achieved by XGBoost is 72.62% and 72.89%. For Cod-rna, MAL achieves the best AUC of 92.77% at $q = 20$ and 96.61% at $q = 18$, while the AUC achieved by XGBoost is 91.97% and 95.80%. Overall, when $q \geq 14$, MAL achieves the same level of accuracy as XGBoost. Because the bucketing operation can somewhat mitigate overfitting, MAL's performance even surpasses XGBoost's at specific q.

Next, we fix the optimal q for each dataset and observe the utility of MAL under different privacy budgets. Figures 5c and 5d show the results for $T = 500$ and $T = 1000$, respectively. When $\varepsilon = 4$, each dataset achieves accuracy very close to that obtained without considering differential privacy. Table 2 shows a comparison of the utility of MAL with $\varepsilon = 4$ against MAL without privacy concerns. Figure 5 shows that MAL achieves almost the same level of accuracy as XGBoost.

Table 2: Utility of MAL with $\varepsilon = 4$.

Datasets	T=500		T=1000	
	MAL	MAL(DP)	MAL	MAL(DP)
SUSY	75.04	74.26	76.47	75.92
Diabetes	72.62	72.62	72.89	71.71
Cod-rna	90.98	94.42	93.99	94.67

Table 3: Efficiency of MAL & XGBoost.

Datasets	XGBoost	MAL						
	time(ms)	time(ms)	comp(ms)		comm(KB)		latency(ms)	
			prep	train	prep	train	prep	train
SUSY	3853.565	5674.162	1087.863	3815.308	1.547	904.192	200.634	570.357
Diabetes	28.996	602.684	173.75	26.976	0.687	4.096	200.281	201.677
Cod-rna	1855.945	2597.845	237.895	1863.618	0.692	234.496	200.283	296.049

4.2 Efficiency

We spawn 10 processes, each representing a participant, to run the experiment. For communication overhead, to simulate a wide-area network, we limit each process's network bandwidth to 20Mbit/s and add a 100ms delay for each link connection. All evaluations about ORE were performed with the bilinear pairing parameter set "d159.param" under the security parameter $\lambda = 80bits$. We present the computation time overhead and communication cost for the preprocessing and training stages in table 3.

For communication costs, MAL needs to share global extremes during the preprocessing stage, share data that satisfies differential privacy before training, and share the parameters of the GBDT model after training is complete. For communication time, during the preprocessing stage, MAL only needs to communicate when obtaining the global extremes of features, resulting in minimal communication cost. The communication time is mainly due to network bandwidth delay. Even during the training phase, MAL only requires two communication steps, which constitute one round trip, with the total communication time being less than 1 second. We observe that the total time overhead of MAL is very close to that of XGBoost.

The above indicates that the additional overhead incurred by MAL due to privacy protection in a distributed setting is minimal. Despite MAL operating under distributed data sharing conditions, its overall time overhead remains impressive due to the minimal number of communications.

We compared MAL with FederBoost [18], a state-of-the-art solution which is 4668 times faster than scheme [1] for horizontal federated GBDT. They used 32 processes to represent participants, and randomly selected 193,333 samples from SUSY for training and trained 60 trees. We trained MAL under the same

settings. The results indicate that MAL has a runtime of approximately 5.23 seconds, achieving a 593-fold speedup compared to FederBoost [18], which requires 3103.24 seconds.

5 Conclusion

To enable multi-party collaborative training of Gradient Boosting Decision Tree (GBDT) with privacy protection in distributed scenarios, federated GBDT is widely used. However, frequent communication leads to inefficiency. This paper proposes the Masked Aggregation Learning (MAL) framework in a horizontal federated setting, which combines distributed data preprocessing and centralized training to significantly reduce communication rounds to a constant. The Secret Extremes Bucket Construction algorithm, based on order-preserving encryption, is introduced for feature bucketing and feature value obfuscation. Additionally, the data is further protected through an Erasable Label Mask generation algorithm, which eliminates masks in each bucket during centralized training to achieve optimal model performance. We also prove that MAL satisfies Local Differential Privacy. Experiments on multiple datasets show that MAL is competitive in both privacy and efficiency. Future work will extend MAL from horizontally distributed scenarios to more complex settings, such as vertical distribution and migration.

References

1. Abspoel, M., Escudero, D., Volgushev, N.: Secure training of decision trees with continuous attributes. Cryptology ePrint Archive, Paper 2020/1130 (2020), `https://eprint.iacr.org/2020/1130`
2. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)
3. Cao, X., Jia, J., Gong, N.Z.: Provably secure federated learning against malicious clients. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 6885–6893 (2021)
4. 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)
5. Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., Yang, Q.: Secureboost: A lossless federated learning framework. IEEE intelligent systems **36**(6), 87–98 (2021)
6. Dwork, C.: Differential privacy: A survey of results. In: International conference on theory and applications of models of computation. pp. 1–19. Springer (2008)
7. Feng, Z., Xiong, H., Song, C., Yang, S., Zhao, B., Wang, L., Chen, Z., Yang, S., Liu, L., Huan, J.: Securegbm: Secure multi-party gradient boosting. In: 2019 IEEE international conference on big data (big data). pp. 1312–1321. IEEE (2019)

8. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30** (2017)
9. Ladyzhenskaia, O.A., Solonnikov, V.A., Ural'tseva, N.N.: Linear and quasi-linear equations of parabolic type, vol. 23. American Mathematical Soc. (1968)
10. Lewi, K., Wu, D.J.: Order-revealing encryption: New constructions, applications, and lower bounds. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1167–1178 (2016)
11. Li, Q., Wen, Z., He, B.: Practical federated gradient boosting decision trees. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 4642–4649 (2020)
12. Liu, Z., Lv, S., Li, J., Huang, Y., Guo, L., Yuan, Y., Dong, C.: Encodeore: reducing leakage and preserving practicality in order-revealing encryption. IEEE Transactions on Dependable and Secure Computing **19**(3), 1579–1591 (2020)
13. Liu, Z., He, W., Chang, C.H., Ye, J., Li, H., Li, X.: Spfl: A self-purified federated learning method against poisoning attacks. IEEE Transactions on Information Forensics and Security (2024)
14. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
15. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. arXiv preprint arXiv:1602.05629 **2**(2) (2016)
16. Mozaffari, H., Shejwalkar, V., Houmansadr, A.: Every vote counts:{Ranking-Based} training of federated learning to resist poisoning attacks. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 1721–1738 (2023)
17. Peng, C., Chen, R., Wang, Y., He, D., Huang, X.: Parameter-hiding order-revealing encryption without pairings. In: IACR International Conference on Public-Key Cryptography. pp. 227–256. Springer (2024)
18. Tian, Z., Zhang, R., Hou, X., Lyu, L., Zhang, T., Liu, J., Ren, K.: Federboost: Private federated learning for gbdt. IEEE Transactions on Dependable and Secure Computing (2023)
19. Voigt, P., Von dem Bussche, A.: The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing **10**(3152676), 10–5555 (2017)
20. Wu, X., Huang, H., Ding, Y., Wang, H., Wang, Y., Xu, Q.: Fednp: Towards non-iid federated learning via federated neural propagation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 10399–10407 (2023)
21. Wu, Y., Cai, S., Xiao, X., Chen, G., Ooi, B.C.: Privacy preserving vertical federated learning for tree-based models. arXiv preprint arXiv:2008.06170 (2020)
22. Zhang, J., Hua, Y., Wang, H., Song, T., Xue, Z., Ma, R., Guan, H.: Fedala: Adaptive local aggregation for personalized federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 11237–11244 (2023)