Federated Soft Gradient Boosting Machine for Streaming Data

Ji Feng[1,2(✉)], Yi-Xuan Xu[1,3(✉)], Yong-Gang Wang[2], and Yuan Jiang[3]

[1] Baiont Technology, Nanjing, China
{fengji,xuyixuan}@baiontcapital.com
[2] Sinovation Ventures AI Institute, Beijing, China
{fengji,wangyonggang}@chuangxin.com
[3] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{xuyx,jiangy}@lamda.nju.edu.cn

Abstract. Federated learning has received wide attention in both academic and industrial communities recently. Designing federated learning models applicable on the streaming data has received growing interests since the data stored within each participant may often vary from time to time. Based on recent advancements on soft gradient boosting machine, in this work, we propose the federated soft gradient boosting machine framework applicable on the streaming data. Compared with traditional gradient boosting methods, where base learners are trained sequentially, each base learner in the proposed framework can be efficiently trained in a parallel and distributed fashion. Experiments validated the effectiveness of the proposed method in terms of accuracy and efficiency, compared with other federated ensemble methods as well as its corresponding centralized versions when facing the streaming data.

Keywords: Federated learning · Gradient boosting · Streaming data

1 Introduction

Gradient boosting machine (GBM) has proven to be one of the best tools for descrete data modeling [11]. Its efficient implementations such as XGBoost [4], LightGBM [17] and CatBoost [25] are still the dominant tools for real-world applications ranging from click through rate (CTR) prediction [15], collaborative filtering [2], particles discovery [1], and many more.

Recently, there have been many attempts trying to marry the power of gradient boosting and deep learning. For instance, multi-layered gradient boosting decision trees [9] is the first non-differentiable system while having the capability of learning distributed representations, which was considered only achievable using neural networks. However, just like other GBMs, each base leaner has to be trained after the previous base learners, making the whole system less efficient in terms of parallel computing.

Q. Yang et al. (Eds.): Federated Learning, LNAI 12500, pp. 93–107, 2020.
https://doi.org/10.1007/978-3-030-63076-8_7

Soft gradient boosting machine (sGBM) [8], on the other hand, is the first differentiable gradient boosting system that all base learners can be simultaneously trained, a huge gain in terms of the training efficiency. To do so, the sGBM first wires multiple differentiable base learners together, and injects both local and global objectives inspired from gradient boosting. Since the whole structure is differentiable, all base learners can then be jointly optimized, achieving a linear speed-up compared to the original GBM. When using differentiable soft decision trees as the base learner, such device can be regarded as an alternative version of the (hard) gradient boosting decision trees (GBDT) with extra benefits, especially on handling the streaming data.

Federated learning is considered to be the next generation of distributed learning [28]. It has several benefits compared to the traditional distributed systems (e.g., MapReduce [6] and Ray), such as less communication costs and more advanced data privacy protection guarantees of local nodes. Such system is best practiced when facing non-iid data from multiple data sources, each with strong privacy concerns. In this work, we propose the federated soft gradient boosting framework, aiming to provide a more efficient distributed implementation of sGBM while keeping all the benefits of federated learning and GBMs.

The rest of the paper is organized as follow: First, some related works are discussed; Second, preliminaries are discussed to make the chapter more self-contained; Third, the problem setting and details on the proposed method are presented; Finally, experiment results on the proposed method are reported, and we conclude in the last section.

2 Related Work

Federated learning is a framework recently proposed by Google [22], which is capable of building a learning model based on the local data distributed across different participants. Meanwhile, the local data on each participant remain private, and is invisible to other participants. It has already been successfully applied to several applications such as Gboard for query suggestions [14]. Federated learning can be roughly categorized into three classes [28]: Horizontal federated learning (HFL) focuses on the scenarios where each participant has different samples in the same feature space; Vertical federated learning (VFL) focuses on the scenarios where participants are with different feature spaces; Federated transfer learning (FTL) locates in the intersection of horizontal and vertical federated learning [21], where data in different participants may have different feature spaces and label spaces. Throughout the paper, we focus on the horizontal federated learning setting.

Gradient Boosting Machine (GBM) is a sequential ensemble algorithm that can be used to optimize any differentiable loss function [30]. Base learners in GBM are fitted in an iterative fashion. Concretely, at each iteration, a new base learner is fitted to bridge the gap between the output of fitted base learners in GBM and the ground-truth. The entire learning procedure of GBM can be interpreted as conducting gradient descent in the functional space [11]. The decision tree extension of GBM, Gradient Boosting Decision Tree (GBDT), is one

of the most widely-used ensemble algorithms in the literature. Efficient implementations on GBDT, such as XGBoost [4], LightGBM [17], and CatBoost [25], achieve excellent predictive performance on a broad range of real-world tasks such as particle discovery [1], click through rate prediction [15], and many more. Recently, Feng et al. shows that GBDT can also be used for learning distributed representations [9], which is originally believed to be the special properties of neural networks. Due to the fact that base learners in GBM are fitted sequentially, the original GBM suffers from large training costs. Furthermore, it cannot be directly applied to the streaming data as base learners cannot be modified once they were fitted. More recently, the soft Gradient Boosting Machine (sGBM) is proposed [8], which is able to jointly fit all base learners by assuming that they are differentiable. Apart from achieving competitive performance to the traditional GBMs, sGBM also greatly reduces the training costs by fitting all base learners jointly.

Recently, there have been growing interests on extending GBM to the framework of federated learning. SecureBoost is a seminal tree-based gradient boosting algorithm that solves the vertical federated learning problem [5]. It is able to achieve the same performance as the non-federated version of decision tree based GBM that requires all local data to be aggregated before the training stage. SecureGBM is another GBM framework that focuses on the scenarios in federated learning where participants may have different features, and only one participant owns the ground-truth [10]. Li et al. proposes a novel gradient boosting decision tree model for horizontal federated learning, which achieves much higher training efficiency through a relaxation on the privacy constraints [19]. However, to the best of our knowledges, there is no work on studying the federated version of GBM applicable on the streaming data.

3 Preliminaries

To make our work more self-contained, we give a detailed introduction on related topics in this section: (1) Horizontal federated learning; (2) Gradient boosting machine (GBM) and soft gradient boosting machine (sGBM).

3.1 Horizontal Federated Learning

Federated learning is able to build a learning model using distributed datasets across all participants without any leakage on the private datasets [28]. Both the theoretical studies and experiments show that the privacy and security problems can be greatly mitigated using federated learning. For horizontal federated learning, a standard procedure on building a federated model can be roughly summarized as four steps: (1) Each participant builds a model using local data, and sends the model parameters or training gradients to the coordinator (e.g., a central server) after encryptions; (2) The coordinator aggregates all local parameters or training gradients to updated the federated model; (3) The coordinator broadcasts the updated model to all participants after model encryptions; (4)

Each participant updates the local model after receiving the shared model from the coordinator. The procedure above can be iteratively conducted for many rounds to achieve better performance. Figure 1 is a graphical illustration on building a federated model in horizontal federated learning.

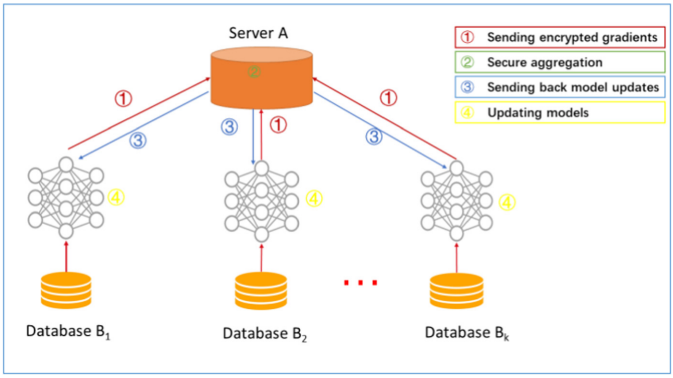

Fig. 1. A paradigm of horizontal federated learning (HFL) [28].

For many applications in federated learning, a central problem is how to effectively utilize the training data arriving in a streaming fashion [16]. A federated learning model applicable on the streaming data is able to improve the performance on both learning tasks and security, as the transmission costs between participants and the coordinator, and inconsistencies in dynamically evolving datasets can both be greatly reduced.

3.2 GBM and sGBM

Gradient boosting machine (GBM) is a popularly-used ensemble algorithm in the literature [11]. Given M base learners $\{h_m\}_{m=1}^M$, with each of them parametrized by $\{\boldsymbol{\theta}_m\}_{m=1}^M$, the goal of GBM is to determine their parameters such that the additive output of all base learners $\sum_{m=1}^M h_m(\mathbf{x};\boldsymbol{\theta}_m)$ is able to minimize a predefined empirical loss function l over the training dataset $\{\mathbf{x}^i, y^i\}_{i=1}^N$. Formally,

$$\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*, \cdots, \boldsymbol{\theta}_M^* = \arg\min_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_M} \sum_{i=1}^N l(y^i, \sum_{m=1}^M h_m(\mathbf{x}^i;\boldsymbol{\theta}_m)). \tag{1}$$

The original GBM determines the learner parameter $\boldsymbol{\theta}_m$ in a sequential way, resulting in a total number of M training iterations [11]. Concretely, during the m-th iteration, the learner parameter $\boldsymbol{\theta}_m^*$ is determined by:

$$\boldsymbol{\theta}_m^* = \arg\min_{\boldsymbol{\theta}_m} \sum_{i=1}^N (r_m^i - h_m(\mathbf{x}^i;\boldsymbol{\theta}_m))^2, \tag{2}$$

where r_m^i denotes the residual defined in GBM, defined as the negative gradient of the loss function with respect to the additive output of fitted base learners before h_m. Therefore, the learning target of the m-th base learner in gradient boosting can be considered as a regression problem on a new training dataset $\{\mathbf{x}^i, r_m^i\}_{m=1}^M$ using the squared error. Once the learner parameter $\boldsymbol{\theta}_m^*$ is determined, the training procedure of GBM moves on to the next iteration.

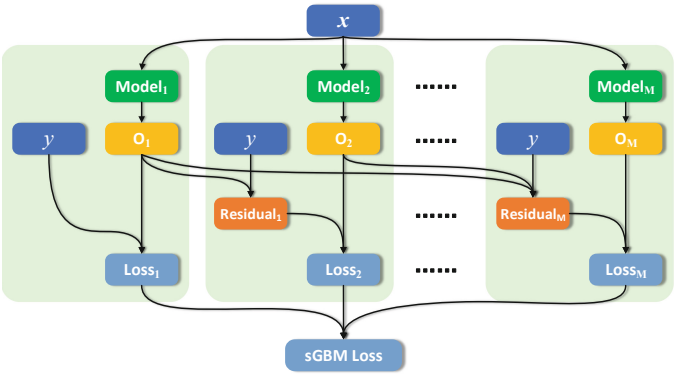

Fig. 2. The graphical illustration on soft gradient boosting machine [8].

On the other hand, sGBM jointly fits all base learners based on the assumption that base learners are differentiable. To be brief, given the output of all base learners on a sample $\mathbf{x}^i$: $\{o_1^i, o_2^i, \cdots, o_M^i\}$, sGBM then simultaneously computes the residuals for all base learners. For the m-th base learner, its residual r_m^i on a sample $\mathbf{x}^i$ is defined as follows:

$$r_m^i = -\frac{l(y^i, \sum_{j=0}^{m-1} o_j^i)}{\sum_{j=0}^{m-1} o_j^i}, \tag{3}$$

where o_0^i is a pre-defined null output, and returns zero for any given input $\mathbf{x}^i$. Therefore, the residual for a base learner in sGBM is defined as the negative gradient of the additive output of base learners before it with respect to the loss function. Note that a key difference on the definition of residual between GBM and sGBM is that sGBM does not require a base learner to first be fitted before computing the residual for subsequent base learners. Given the set $\{o_1^i, o_2^i, \cdots, o_M^i\}$, the residuals for all base learners $\{r_1^i, r_2^i, \cdots, r_M^i\}$ can be computed in parallel. Given the computed residuals $\{r_m^i\}_{m=1}^M$, a final sGBM loss on each sample $\mathbf{x}^i$ is then defined as:

$$\mathcal{L}^i = \sum_{m=1}^{M} (o_m^i - r_m^i)^2. \tag{4}$$

Assuming that all base learners are differentiable, sGBM then is able to adopt the error back-propagation and online optimization techniques (e.g., stochastic gradient descent) to simultaneously update parameters for all base learners. Figure 2 presents the computation graph of sGBM to obtain a final loss $\mathcal{L}$.

3.3 Streaming Data

With a rapid growth on the volume of datasets (e.g., ImageNet [7], COCO [20]), it becomes increasingly difficult to deploy learning models under the traditional offline setting. For example, computation devices in many scenarios have very limited computation power and memory storage (e.g., smartphones, wireless sensors, and many more), making deploying learning models in the offline setting onto them prohibitively expensive. Meanwhile, these methods cannot properly handle the concept drift, such as distribution changes on the training data [13]. As a result, many algorithms focusing on streaming data have been developed. For example, models lying in the offline setting can be combined with model reuse to handle the streaming data [26,29]. On the other hand, models that can be directly fitted using some variants of online optimization techniques can naturally adapt to the streaming data, such as the deep neural network.

4 The Proposed Method

In this section, we formally present the proposed `Fed-sGBM`. First, the problem setting is presented; Second, details on the training stage of `Fed-sGBM` are introduced; Third, we introduce the extension of `Fed-sGBM` when using decision trees as the base learner. We conclude this section with a discussion on the communication costs of `Fed-sGBM`.

4.1 Problem Setting

In the section, we formally introduce the problem setting. Given K participants $\{P_1, P_2, \cdots, P_K\}$, with each of them equipped with a local dataset $\mathcal{D}_k$, the problem is to build a GBM model based on all local datasets $\{\mathcal{D}_k\}_{k=1}^K$ with the existence of a coordinator C. Meanwhile, it is required that the local dataset of each participant remains private to the coordinator and other participants. Following [5], we make the assumption that the coordinator C is trustable, and the communication between C and each participant P_k is securely encrypted using schemes such as homomorphic encryption. Furthermore, each local dataset $\mathcal{D}_k$ evolves with the time. Upon a new local dataset $\mathcal{D}_k^{(t)}$ arriving at the time t, the old dataset $\mathcal{D}_k^{(t-1)}$ will be instantly discarded and no longer available for the training stage. Such streaming setting is more realistic in the real-world because participants in federated learning can be cheap devices such as smartphones, which are unable to store a large volume of local data. Figure 3 is a graphical illustration on our problem setting.

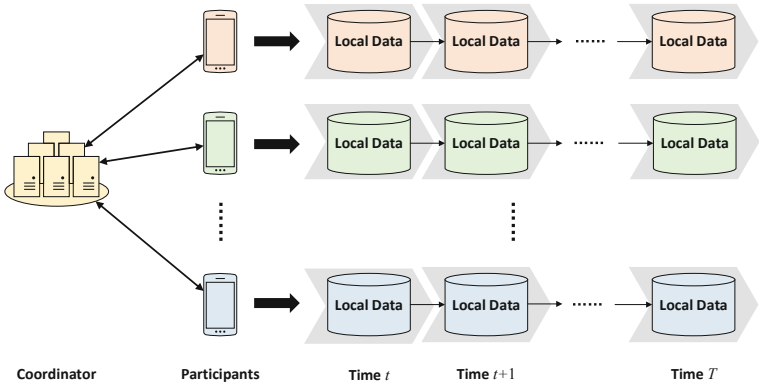

Fig. 3. Graphical illustration of local participants over time

4.2 Federated Soft Gradient Boosting Machine

Gradient Boosting Machine is a powerful ensemble algorithm that achieves excellent predictive performance on many real-world problems [23]. However, it cannot be directly applied to our problem setting illustrated above because of its inherently sequential training procedure: a base learner is required to be fixed before fitting the next base learner. We propose to extend the sGBM onto the framework of federated learning to solve the targeted problem, where base learners can be trained in a parallel and distributed fashion. Our results are a novel GBM model under the framework of federated learning. The rest of this section focuses on the training details on the proposed `Fed-sGBM`.

First, the training stage of `Fed-sGBM` is initiated by the coordinator C. After determining the kind of base learner and the number of base learners M, the coordinator then builds M base learners of the specified type $\{h_m\}_{m=1}^{M}$, and the learner parameters $\{\boldsymbol{\theta}_m\}_{m=1}^{M}$ are randomly initialized. Second, the model parameters are updated iteratively for a total number of T rounds via the communications between the coordinator and all participants. At last, the coordinator aggregates the local model parameters from all participants, and computes a final model via learner-wise model averaging:

$$h_m(\mathbf{x};\boldsymbol{\theta}_m) = \frac{1}{K}\sum_{k=1}^{K} h_{m,k}(\mathbf{x};\boldsymbol{\theta}_{m,k}) \ \forall m \in \{1,2,\cdots,M\}, \tag{5}$$

where $h_{m,k}(\mathbf{x};\boldsymbol{\theta}_{m,k})$ denotes the parameter of the m-th base learner in the participant P_k. To prevent the local datasets from leaking, the communications between the coordinator and participants are restricted to transmitting the model parameters. Furthermore, the homomorphic encryption is used to improve the security on transmissions.

Concretely, at each round t, each participant P_k first receives a copy of model parameters from the coordinator C via broadcasting. After then, each participant splits the local dataset into non-overlapping batches $\{B_1, B_2, \cdots, B_{|\mathcal{B}|}\}$ for

Algorithm 1: Training stage of `Fed-sGBM`

Input: Number of rounds T, learner parameters from the coordinator $\{\boldsymbol{\theta}_m\}_{m=1}^{M}$, local training data $\{\mathcal{D}_k\}_{k=1}^{K}$
Output: Updated model parameters $\{\boldsymbol{\theta}_m^*\}_{m=1}^{M}$

for $t = 1, 2, \cdots, T$ **do**
  `// The side of participants`
  for $k = 1, 2, \cdots, K$ **do**
    Receive the learner parameters from the coordinator: $\boldsymbol{\theta}_{m,k} \leftarrow \boldsymbol{\theta}_m \ \forall m \in \{1, 2, \cdots, M\}$;
    Split the local training data $\mathcal{D}_k$ into batches: $\mathcal{B} \leftarrow \{B_1, B_2, \cdots, B_{|\mathcal{B}|}\}$;
    for $b = 1, 2, \cdots, |\mathcal{B}|$ **do**
      Conduct data forward: $o_m^i \leftarrow \sum_{j=0}^{m-1} h_j(\mathbf{x}^i; \boldsymbol{\theta}_{m,k}) \ \forall \mathbf{x}^i \in B_b$;
      Compute the residual: $r_m^i \leftarrow -\frac{\partial l(y^i, o_m^i)}{\partial o_m^i} \ \forall \mathbf{x}^i \in B_b$;
      Compute the training loss on the current data batch: $\mathcal{L}_{b,k} \leftarrow \sum_{\mathbf{x}^i \in B_b} \sum_{m=1}^{M} (o_m^i - r_m^i)^2$;
      Update $\{\boldsymbol{\theta}_{m,k}\}_{m=1}^{M}$ w.r.t $\mathcal{L}_{b,k}$ using gradient descent ;
    end
    Send $\{\boldsymbol{\theta}_{m,k}\}_{m=1}^{M}$ to the coordinator C ;
  end
  `// The side of the coordinator`
  Receive parameters from all participants: $\{\boldsymbol{\theta}_{m,k}\}_{k=1}^{K} \ \forall m \in \{1, 2, \cdots, M\}$;
  $\boldsymbol{\theta}_m \leftarrow \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\theta}_{m,k} \ \forall m \in \{1, 2, \cdots, M\}$;
end
$\{\boldsymbol{\theta}_m^*\}_{m=1}^{M} \leftarrow \{\boldsymbol{\theta}_m\}_{m=1}^{M}$;
Return $\{\boldsymbol{\theta}_m^*\}_{m=1}^{M}$;

efficient model updating. For each batch of data, a training loss defined in Eq. (4) is first computed. After then, each participant adopts error back-propagation and first-order optimization techniques to update local model parameters, following the routine in sGBM [8]. At last, the updated model parameters is transmitted to the coordinator after encryptions. Upon receiving model parameters from all participants, the coordinator uses model averaging to compute a final model as in Eq. (5), and the training stage moves to the next round. Algorithm 1 presents details on the training stage of `Fed-sGBM`.

Since the local model on the side of participants can be efficiently updated in a mini-batch fashion using back propagation, the proposed `Fed-sGBM` can be naturally applied to the streaming data. Suppose that at time t, a new batch of data $\mathcal{D}_k^{(t)}$ arrives at the participant P_k and replaces the old data $\mathcal{D}_k^{(t-1)}$, there is no need for the participant to train a new local model from scratch. The only modification needs to be made is to update parameters on the batches of the new data $\mathcal{D}_k^{(t)}$. All base learners in the existing local model then can be jointly trained to adapt to the newly-coming batch of data.

According to the Algorithm 1, the communication cost between the coordinator and each participant is $2 \times T \times M \times |\boldsymbol{\theta}|$, where M is the number of base

learners in `Fed-sGBM`, $|\boldsymbol{\theta}|$ denotes the size of model parameters. As an ensemble algorithm in federated learning, a larger value of M is able to effectively improve the overall performance of `Fed-sGBM`, yet the communication cost also increases linearly. Therefore, a trade-off exists between the performance of `Fed-sGBM` and the communication cost.

4.3 Federated Soft Gradient Boosting Decision Tree

Decision tree models, such as Classification And Regression Tree (CART) [3], are popularly used as the base learner in gradient boosting. State-of-the-art gradient boosting decision tree (GBDT) libraries such as XGBoost [4], LightGBM [17], and CatBoost [25] achieve excellent predictive performance on a variety number of real-world tasks. Due to the fact that building a single decision tree is a one-pass procedure that requires all data to first be aggregated before training, extending GBDTs to the horizontal federated learning setting is non-trivial. In this section, we extend the proposed `Fed-sGBM` to the cases where tree-based model is used as the base learner, and the result is a novel GBDT model for horizontal federated learning: `Fed-sGBDT`.

Concretely, a novel soft decision tree model is used in `Fed-sGBDT` to ensure that the entire model can still be updated using back propagation [12]. Compared to classic tree-based models that assign each sample to a single leaf node, soft decision tree assigns each sample to all leaf nodes with different probabilities. To achieve this, each internal node in the soft decision tree is equipped with a logistic regression model that splits each sample to its child nodes with different probabilities. The output of the entire model on a sample is the weighted average of predictions from all leaf nodes, where weights correspond to the probabilities assigned to each leaf node. Due to the fact that the soft decision tree is differentiable, it can be naturally integrated into the proposed `Fed-sGBM`, leading to the proposed `Fed-sGBDT`. Since the entire `Fed-sGBDT` model can still be trained using online optimization techniques such as stochastic gradient descent, it is capable of quickly adapting to the streaming data.

5 Experiment

Experiments are divided into five sub-sections, and the goals are to validate that: (1) The proposed method is an effective ensemble algorithm for horizontal federated learning; (2) The proposed method applies well to the streaming data setting in federated learning, which are frequently encountered in the real-world.

First, we introduce the setup of the experiments; Second, we compare the performance of `Fed-sGBM` with different contenders; After then, we investigate the performance under different number of participants and base learners, separately; At last, the performance of `Fed-sGBM` on the streaming data is presented.

5.1 Experiment Setup

Dataset. Three benchmark datasets on classification are selected to evaluate the performance of `Fed-sGBM`: *Letter*, *MNIST*, and *CIFAR-10*. *Letter* is a dataset that contains samples belonging to 26 capital letters in the English alphabet. Each sample is associated with 16 features that contain statistical information such as edge counts. *MNIST* is a dataset on handwritten digit recognition, and each sample corresponds to a gray image of size 28×28. *CIFAR-10* is another image classification dataset with 10 classes, and each sample is a colorful image of size $3 \times 32 \times 32$. All datasets are normalized to its mean and unit variance before the training stage. Table 1 presents basic statistics on the datasets used.

Table 1. Basic statistics on the datasets used

Dataset name	# Training	# Evaluating	# Features	# Classes
Letter	16000	4000	16	26
MNIST	60000	10000	28×28	10
CIFAR-10	50000	10000	$3 \times 32 \times 32$	10

Base Learner. Three different kinds of base learners are included to evaluate `Fed-sGBM`: *SoftTree*, *MLP*, and *CNN*. For *SoftTree*, we set the tree depth as 5. We also use a regularization term introduced in [12] during the training stage to encourage the *SoftTree* to exploit its leaf nodes. The coefficient of this regularization term is set as 0.001. For *MLP*, since it is difficult to find the best network architecture for each dataset in practice, we directly use the architecture reported in the literature [31]. Concretely, the architecture of *MLP* on *Letter* dataset is Input$-70-50-$Output, and the architecture on *MNIST* and *CIFAR-10* dataset is Input$-512-512-$Output. For *CNN*, we adopt a modified version of LeNet-5 [18] with Relu activation and dropout [27].

Computation. All experiments are conducted on a single machine with 32GB RAM, a Xeon E5-2650v4 CPU, and a RTX-2080Ti GPU. We implement the `Fed-sGBM` using PyTorch [24]. The distributed communication package Torch-Distributed is used to simulate the distributed setting in federated learning.

Simulation. Given K participants exist in federated learning, we split the training data into K non-overlapping parts with relatively equal size, and each part is treated as the local dataset on a participant P_k.

5.2 Performance Comparison

In this section, we evaluate the performance of `Fed-sGBM` as an ensemble algorithm in federated learning. For performance comparison, we also report the results of `Fed-Voting`, sGBM [8], and the original GBM [11]. In `Fed-Voting`, the coordinator and all participants jointly learn a voting-based ensemble model

without sharing the local data. Notice that sGBM and GBM requires local data from all participants, and cannot be directly applied to the federated learning setting. We keep the type of base learner same in three methods, and set the number of base learner as 10. The number of participants in `Fed-sGBM` and `Fed-Voting` is set as 2. Experiment results on different configurations of datasets and base learners are presented in Table 2. The performance of *CNN* on the *Letter* dataset is ignored since it cannot be applied to tabular datasets.

According to Table 2, it can be observed that the performance of `Fed-sGBM` is slightly worse than sGBM, which is reasonable considering that the `Fed-sGBM` is trained in a distributed fashion. On the other hand, the performance of `Fed-sGBM` outperforms `Fed-Voting` by a large margin on different configurations of base learner and dataset except the Tree@Letter, validating its effectiveness as a general ensemble method in federated learning.

Table 2. Performance comparison between `Fed-sGBM` and different baselines

Configuration	`Fed-sGBM`	`Fed-Voting`	`sGBM`	`GBM`
Tree@Letter	88.43	94.88	89.15	87.43
MLP@Letter	94.10	89.78	95.60	95.83
Tree@MNIST	96.37	94.34	97.02	95.88
MLP@MNIST	98.82	97.44	98.70	98.49
CNN@MNIST	99.52	99.04	99.53	99.34
Tree@CIFAR-10	50.70	41.57	51.86	50.92
MLP@CIFAR-10	58.10	51.62	57.46	55.89
CNN@CIFAR-10	74.92	71.97	75.02	74.68

5.3 Performance Under Different Number of Base Learners

In this section, we investigate the performance of `Fed-sGBM` when increasing the number of base learners. The goal is to validate that increasing the number of base learners is able to effectively improve the performance of `Fed-sGBM`.

Based on the datasets and base learners presented above, we set the number of base learners as $\{1, 5, 10, 15, 20\}$, and evaluate the testing accuracy of `Fed-sGBM`. The experiment results are presented in Fig. 4. It can be shown that the testing accuracy of `Fed-sGBM` consistently increases on all combinations of datasets and base learners with more base learners added. The experiment results validate our claims: increasing the number of base learners effectively improves the performance of `Fed-sGBM` on different datasets.

5.4 Performance Under Different Number of Participants

In this section, we evaluate the performance of `Fed-sGBM` with different number of participants existing in federated learning. The goal is to validate that the performance of `Fed-sGBM` is robust to the number of participants.

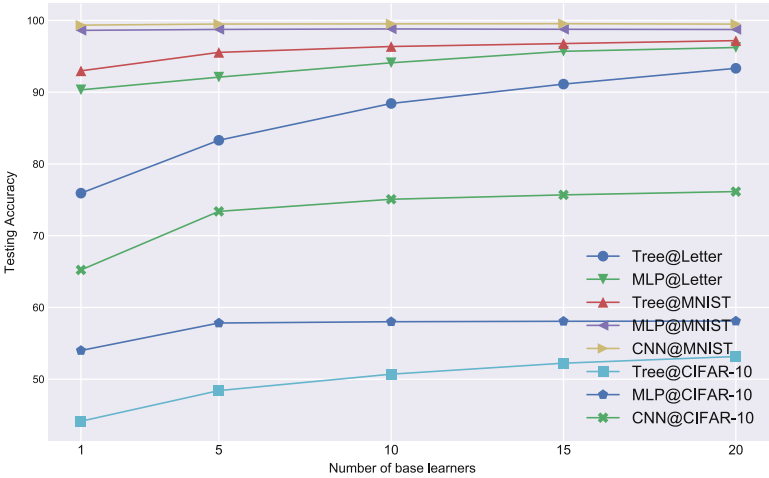

Fig. 4. Performance of `Fed-sGBM` with different number of base learners

Concretely, we train a shared `Fed-sGBM` model when there are $\{2, 3, 4\}$ participants in federated learning, respectively. The experiment results are presented in Fig. 5. According to the experiment results, it can be observed that the performance of `Fed-sGBM` only deteriorates slightly on three datasets with more participants added. Since the volume of the local dataset on each participant decreases drastically based on our experiment setup, the experiment results validate that the performance of `Fed-sGBM` is robust to the number of participants. Notice that the performance of `Fed-sGBM` even improves with the number of participants increasing when using the *MLP* base learner on *CIFAR-10* dataset. We conjecture that the reason is that the procedure of obtaining a jointly learned `Fed-sGBM` model can be considered as one kind of ensemble on all local models (i.e., model averaging). Therefore, the final performance of `Fed-sGBM` improves with more participants added, despite the fact that each participant has less volume of training data.

5.5 Performance on Streaming Data

In this section, we evaluate the performance of `Fed-sGBM` under the streaming data setting in federated learning. Given a dataset, we first randomly split the training part into 5 non-overlapping batches to simulate the streaming data setting. At each time t ($t = 1, 2, \cdots, 5$), each participant P_k will receive a newly-coming batch of data $\mathcal{D}_k^{(t)}$, and the local model will then be fitted on $\mathcal{D}_k^{(t)}$. We report the performance of `Fed-sGBM` on the testing data after the shared model is fitted on each batch of data from all participants, leading to a total number of five testing accuracy records. The experiment results on three combinations of base learner and dataset (MLP@Letter, Tree@MNIST, CNN@CIFAR-10) are presented in Table 3. The number of base learners and participants in `Fed-sGBM` is set as 10 and 2, separately.

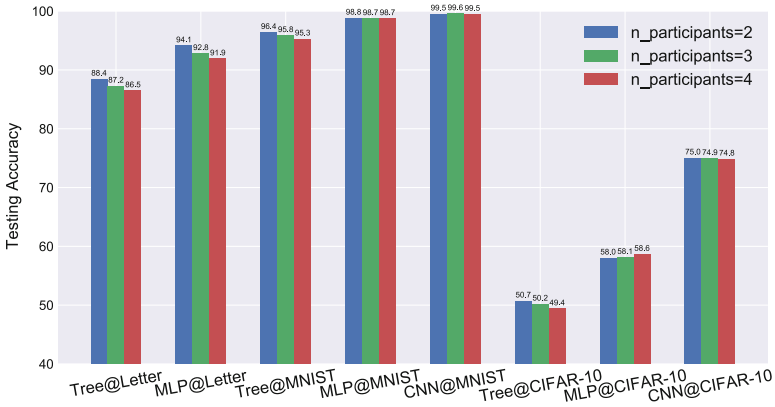

Fig. 5. Performance of `Fed-sGBM` with different number of participants

Table 3. Performance of `Fed-sGBM` on the streaming data setting

Learner@Dataset	Timestamp ID				
	1	2	3	4	5
MLP@Letter	79.00	83.75	85.00	86.78	86.93
Tree@MNIST	94.57	95.62	96.34	96.73	96.73
CNN@CIFAR-10	60.77	62.12	63.54	64.92	65.45

According to the experiment results in Table 3, it can be observed that the performance of `Fed-sGBM` consistently improves with more batches of data coming. Meanwhile, a better performance can also be achieved by tuning the parameters and training the local model on each batch of data for more epochs. Such results cannot be easily achieved by the original GBM because at each time t, a new model has to be trained from scratch, with the learned local model before totally wasted.

6 Conclusion

In this chapter, we propose a novel gradient boosting model for horizontal federated learning. The proposed method is able to utilize local datasets from all participants to efficiently build a shared model without any leakage on the local data. Compared to existing works on combining gradient boosting with federated learning, the proposed `Fed-sGBM` can be applied onto the streaming data, and is robust to the number of participants in federated learning. Extensive experiments are conducted to evaluate the performance of `Fed-sGBM` as an ensemble algorithm in the framework of federated learning, along with the performance on streaming data.

References

1. Baldi, P., Sadowski, P., Whiteson, D.: Searching for exotic particles in high-energy physics with deep learning. Nat. Commun. **5**(1), 1–9 (2014)
2. Bennett, J., Lanning, S., et al.: The Netflix prize. In: KDD Cup 2007, vol. 35 (2007)
3. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. CRC Press, Boca Raton (1984)
4. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: SIGKDD, pp. 785–794 (2016)
5. Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Yang, Q.: Secureboost: a lossless federated learning framework. arXiv preprint arXiv:1901.08755 (2019)
6. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: ImageNet: a large-scale hierarchical image database. In: CVPR, pp. 248–255 (2009)
8. Feng, J., Xu, Y.X., Jiang, Y., Zhou, Z.H.: Soft gradient boosting machine. arXiv preprint arXiv:2006.04059 (2020)
9. Feng, J., Yu, Y., Zhou, Z.H.: Multi-layered gradient boosting decision trees. In: NIPS, pp. 3551–3561 (2018)
10. Feng, Z., et al.: SecureGBM: secure multi-party gradient boosting. In: IEEE International Conference on Big Data, pp. 1312–1321. IEEE (2019)
11. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**, 1189–1232 (2001)
12. Frosst, N., Hinton, G.: Distilling a neural network into a soft decision tree. arXiv preprint arXiv:1711.09784 (2017)
13. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 1–37 (2014)
14. Hard, A., et al.: Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604 (2018)
15. He, X., et al.: Practical lessons from predicting clicks on ads at Facebook. In: International Workshop on Data Mining for Online Advertising, pp. 1–9 (2014)
16. Kairouz, P., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
17. Ke, G., et al.: LightGbm: a highly efficient gradient boosting decision tree. In: NIPS, pp. 3146–3154 (2017)
18. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. IEEE **86**(11), 2278–2324 (1998)
19. Li, Q., Wen, Z., He, B.: Practical federated gradient boosting decision trees. In: AAAI, pp. 4642–4649 (2020)
20. Lin, T.-Y.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
21. Liu, Y., Chen, T., Yang, Q.: Secure federated transfer learning. arXiv preprint arXiv:1812.03337 (2018)
22. 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 (2017)
23. Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial. Front. Neurorobotics **7**, 21 (2013)

24. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: NIPS, pp. 8026–8037 (2019)
25. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. In: NIPS, pp. 6638–6648 (2018)
26. Shalev-Shwartz, S.: Online learning and online convex optimization. Foundations Trends Mach. Learn. **4**(2), 107–194 (2011)
27. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
28. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 1–19 (2019)
29. Zhao, P., Cai, L.-W., Zhou, Z.-H.: Handling concept drift via model reuse. Mach. Learn. **109**(3), 533–568 (2019). https://doi.org/10.1007/s10994-019-05835-w
30. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. CRC Press, Boca Raton (2012)
31. Zhou, Z.H., Feng, J.: Deep forest. In: IJCAI, pp. 3553–3559 (2017)