Privacy-Preserving Boosting in the Local Setting

Sen Wang, and J. Morris Chang, Senior Member, IEEE

***Abstract*—In machine learning, boosting is one of the most popular methods that is designed to combine multiple base learners into a superior one. The well-known Boosted Decision Tree classifier has been widely adopted in data mining and pattern recognition. With the emerging challenge in privacy, the data, like social images, browsing history, and financial reports, which are held by individuals and entities are more likely to contain sensitive information. The privacy concern is intensified when the data leaves the hand of the owners and is further used for data mining. Such privacy issues demand that the machine learning algorithms should be privacy-aware. Recently, Local Differential Privacy has been proposed as an effective privacy protection approach, which allows data owners to perturb the data before any release. In this paper, we propose a distributed privacy-preserving boosting algorithm that can be applied to various types of classifiers. By adopting LDP as a building block, the proposed boosting algorithm leverages the aggregation of the perturbed data shares to build the base learner, which ensures that privacy is well preserved for the participated data owners. Our experiments demonstrate that the proposed algorithm effectively boosts various classifiers and a high utility is maintained.**

***Index Terms*—Ensemble Learning, Boosting, Local Differential Privacy**

I. Introduction

In machine learning, ensemble learning [1] refers to the strategy of combining multiple hypotheses to form a better one. Lots of work has been conducted under the area and there are three conventional methods, bagging [2], boosting [3], [4] and stacking [5]. Among all three methods, the boosting method trains a series of weak learners, and the final decision is made by a majority voting of these weak learners. As an example, Boosted Decision Tree (BDT) has great popularity and is widely adopted in many applications, like text mining [6], geographical classification [7], and finance [8]. Initially, the boosting algorithm is developed under the assumption of a centralized fashion, where all data has been collected altogether. However, in the big data era, the data is generated and stored far more sparsely, which brings new challenges and difficulties for such centralized algorithm, for instance, privacy protection is one of the most emerging demands in current society.

Data explosion results in tons of data are generated and held by individuals and entities, such as personal images, browsing histories, and financial records. Training a machine learning model using such data can have severe privacy risks [9]. The AOL search engine log [10] and Netflix prize contest [11] attacks highlight such threats and suggest that machine learning model should be privacy-aware. In the last decade, as a promising solution, a mechanism is said to be differentially private [12] if the computation result of a dataset is robust to any change of the individual sample. A common assumption [13] is that there exists a trusted data curator who gathers data from multiple data parties and honestly runs the private algorithms. Comparing to DP, Local Differential Privacy (LDP) [14] eliminates the need of such trusted data curator, a mechanism is said to be local differentially private if the processing makes any two samples indistinguishable. One advantage of LDP is that it allows the data owners to perturb the local data by themselves and only release the perturbed samples if needed. Thus unlike DP, there is no need of a trusted third party anymore.

The indistinguishability of any two data samples brings a strong privacy guarantee for the data owners, and LDP can be used as an effective tool to develop the privacy-preserving machine learning algorithms. To the best of our knowledge, the existing privacy-preserving boosted classifiers are mostly the tree-based classifiers, like differently private decision tree [15] and differentially private GBDT [16], [17]. By design, boosting is not algorithmically constrained and it can be applied various types of the classifiers. In this paper, we are eager to fill such gap by developing a privacy-preserving boosting algorithm that is model-agnostic. In particular, we consider a distributed setting as RAPPOR [18], where a cloud service operator, we call it the data user, has an intention to collect the data samples from multiple data owners (e.g., tens of millions of Chrome web browser users) and fit a boosted classifier (e.g., BDT). The proposed privacy-preserving boosting algorithm leverages LDP as a building block, and it ensures that the privacy of the individual sample held by data owners is preserved.

The LDP mechanism uses a variable, ϵ, to measure the privacy loss, the smaller ϵ is, the more privacy gets preserved, and privacy budget refers to the maximum privacy loss. In non-private boosting, the training data participates in all iterations, while by adopting ϵ-LDP, reusing the same training data will continuously consume the privacy budget and ultimately leave the data deanonymized. To resolve such challenge, only a subset of data owners is picked to participate in one iteration, and each data owner is asked to contribute the local data share at most once, which provides a strong privacy guarantee.

In our design, the base leaner is trained using the perturbed *data shares* held by the data owners. To boost various types of classifiers, the *data share* falls into three types, i.e., *local sample*, *local statistic* and *local classifier*. *Local sample* is the atomic representation of the data possessed by the data owner and *local classifier* refers to a local classifier that trained by individual data owner. And the *local statistic* fills the gap between the two end representations, which is a statistical summary of the local dataset. To protect the data privacy, we leverage the ϵ-LDP mechanism to protect the data share during the joint training. According to each type of local share, we implement an example of the privacy-preserving boosted classifier and demonstrate how the local data shares are perturbed and aggregated.

Overall, the contribution of our work is three fold:

- We propose a distributed privacy-preserving boosting algorithm which utilizes ϵ-LDP mechanism to prevent privacy leakage in the joint training, and the base learner is trained by the aggregation of the perturbed data shares. For a strong privacy guarantee, each data owner only contributes the perturbed data share and participates in the joint training at most once.
- The proposed algorithm is generalized and is able to boost various types of classifiers trained by different data shares. To demonstrate, we implement the privacy-preserving boosted Nearest Centroid Classifier (w.r.t. *local sample*), the BDT (w.r.t. *local statistic*) and the boosted Logistic Regression classifier (w.r.t. *local classifier*). For each boosted classifier, we analyze the convergence condition and the trade-off between the utility and privacy.
- We comprehensively evaluate three boosted classifiers in terms of the classification performance over real and synthetic datasets. In the experiment, we also horizontally compare against two existing perturbation methods. The classification performance confirm that the adopted LDP perturbation method leads to the minimized error and the boosted classifier effectively maintains a high utility.

The rest of the paper is organized as follow. The preliminary is in Section II. The problem definition and proposed solution are introduced in Section III. Section IV introduces 3 classifiers that are boosted by the proposed algorithm. The evaluation is given in Section V. Section VI presents the related work. Section VII provides the conclusion.

TABLE I: Notations and Symbols

(X^l, Y^l)	the dataset held by the lth data owner
$(\boldsymbol{x_i}^l, y_i^l)$	the ith sample and label held by the lth data owner
N^l	number of sample held by lth the data owner
L	the number of data owners
d	dimensiona of the dataset
x_i'	the perturbed output of x_i
w_i	weights of the ith samples
K	number of classes in the dataset
C_k	kth class label
M	number of base learners in the boosting algorithm
T_m	the mth base learner
ϵ-DP	ϵ-differential privacy
ϵ-LDP	ϵ-local differential privacy
LR	logistic regression
DT	decision tree
BDT	boosted decision tree
NCC	nearest centroid classifier
MSE	mean squared error

II. Preliminary

A. Boosting

In supervised learning, Boosting [3] is a widely used ensemble algorithm to reduce the bias and variance. As an generic algorithm, boosting is designed to train a sequence of weak models and the prediction is made by a majority voting of such weak models. Here, the term "weak" indicates that the each individual model might have a low prediction accuracy, i.e., the weak models could perform slightly better than random guess. Since boosting algorithm trains weak models iteratively, we use weak model and base learner interchangeably in this paper. AdaBoost [4] is one of the most popular learning methods in the last two decades. In the classification problem, the base learner in each iteration is trained to correct the predictions of samples that are misclassified in previous round, and it proves [4] that the final model can converge to a strong learner, if the base learner performs better than random guessing. In general, SAMME [19] is a multi-class AdaBoost algorithm that requires the prediction accuracy of the base learner is better than $1/K$, assuming there are K classes in the problem. Given the training data $\{(\boldsymbol{x}_i, y_i), i = 1 : N\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{C_1, C_2, \ldots, C_K\}$, the pseudo code of the SAMME [19] algorithm is presented in Alg. 1:

Algorithm 1 Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME)

```
1:  **Input**: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$.
2:  Initialize the observation weights $w_i = 1/N, i = 1 : N$
3:  **for** m=1:M **do**
4:      Fit a classifier $T_m(\cdot)$ to the training data using weights $w_i$
5:      $err_m \leftarrow \frac{\sum_{i=1}^{N} w_i \mathbb{I}(y_i \neq T_m(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i}$
6:      $\alpha_m \leftarrow log\frac{1-err_m}{err_m} + log(K-1)$
7:      $w_i \leftarrow w_i \cdot exp(\alpha_m \cdot \mathbb{I}(y_i \neq T_m(\boldsymbol{x}_i))), i = 1 : N$
8:      Re-normalize $w_i$
9:  **end for**
10: **return** $\{(\alpha_1, T_1(\cdot)), \ldots, (\alpha_m, T_m(\cdot))\}$
```

With the output of Alg. 1, given a testing sample $\boldsymbol{x}_t$, $\hat{y_t} \in \{C_1, C_2, \ldots, C_K\}$ is assigned by the majority vote of m weighted base learners, i.e., $\text{argmax}_k \sum_m \alpha_m \cdot \mathbb{I}(T_m(\boldsymbol{x}_t) = C_k)$.

B. Boosted Decision Tree

Boosted Decision Tree(BDT) classifier trains a sequence of Decision Tree (DT) classifiers, where the DT is created iteratively. The internal node of the tree represents a condition, and it decides which branch to proceed. Building a DT needs to pick the proper attributes for internal nodes consequently and the tree stops growing when it reaches the leaf node. In each internal node, the best attribute to split is determined by the impurity measurement before and after split over that attribute, where information gain, Gini index and misclassification error [20] are 3 common impurity measurements. In this paper, we use the misclassification error to determine the best attribute, since it is successfully adopted in the distributed environment [21]. Usually, BDT trains several DTs that only have one binary branch, which are called decision stumps and the decision function is given below,

$$f(\boldsymbol{x}|j, \eta) = \begin{cases} +1, \text{ if } x_j \geq \eta, \\ -1, \text{ otherwise}, \end{cases} \tag{1}$$

where x_j refers to the jth feature value of $\boldsymbol{x}$ and η refers to a threshold.

C. Differential Privacy

Differential Privacy (DP) [12] attracts lots of attention in the privacy research community in the last decades, which provides a measurement of the information leakage of individual samples inside an underlying dataset. DP considers the setting that a trusted data curator gathers data from multiple data owners and performs statistical analysis over the data, like learning the mean or variance of the data. However, releasing such statistical analysis without any protection may reveal the existence of the individual sample. Thus the general idea of DP is to perturb the true statistical value and hide the existence of underlying samples. Given a mechanism that satisfies ϵ-DP, the smaller ϵ is, the more difficult to derive the existence of a sample.
Formally, given two data databases A, A^*, it is said that A, A^* are *neighbors* if they differ on at most one row. The definition of a (ϵ, δ)-differential private mechanism over A is defined below:

Definition 1 (ϵ, δ)-Differential Privacy [12]: A randomized mechanism $\mathcal{F}$ is (ϵ, δ)-differentially private if for every two neighboring databases A, A^* and for any $\mathcal{O} \subseteq Range(\mathcal{F})$,

$$Pr[\mathcal{F}(A) \in \mathcal{O}] \leq e^{\epsilon} Pr[\mathcal{F}(A^*) \in \mathcal{O}] + \delta, \tag{2}$$

where $Pr[\cdot]$ denotes the probability of an event, $Range(\mathcal{F})$ denotes the set of all possible outputs of the algorithm $\mathcal{F}$. The smaller ϵ, δ are, the closer $Pr[\mathcal{F}(A) \in \mathcal{O}]$ and $Pr[\mathcal{F}(A^*) \in \mathcal{O}]$ are, and the stronger privacy protection gains. When $\delta = 0$, the mechanism $\mathcal{F}$ satisfies ϵ-DP, which is a stronger privacy guarantee than (ϵ, δ)-DP with $\delta > 0$.

D. Local Differential Privacy

Similar to DP, Local Differential Privacy (LDP) is introduced by Duchi et al. [14], which assumes that the data owners don't even trust the data curator, and the privacy protection is shifted from the data curator to data owners themselves. And the perturbation can be applied to the samples of a dataset rather than the statistics. The formal definition is given below:

Definition 2 ϵ-Local Differential Privacy [14]: A randomized mechanism $\mathcal{G}$ satisfies ϵ-LDP if for any *input* v_1 and v_2 and for any $\mathcal{O} \subseteq Range(\mathcal{G})$:

$$Pr[\mathcal{G}(v_1) \in \mathcal{O}] \leq e^{\epsilon} Pr[\mathcal{G}(v_2) \in \mathcal{O}]. \tag{3}$$

Compared to DP, LDP provides a stronger privacy protection in the distributed setting. Instead of releasing the true dataset, the data owners who own a local dataset perturb all samples using the mechanism that satisfies ϵ-LDP and then only share the perturbed samples, which ensures that the true dataset never leave the data owners' hands.

Several perturbation methods that satisfy ϵ-LDP are studied to perturb the categorical [18], [22] and numerical [23]–[25] values respectively. In this paper, we use algorithms developed by Wang et al. [25] as a building block to perturb the local shares held by the data owner, the formal algorithms are introduced in the following subsections.

Algorithm 2 Piecewise Mechanism for One-Dimensional Numeric Data [25]

```
1: **Input**: $(x_i, \epsilon)$
2: $\Delta \leftarrow \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}$
3: $\psi_{left}(x_i) \leftarrow \frac{\Delta+1}{2} \cdot x_i - \frac{\Delta-1}{2}$
4: $\psi_{right}(x_i) \leftarrow \psi_{left}(x_i) + \Delta - 1$
5: Sample $v$ uniformly at random from $[0, 1]$;
6: if $v < \frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$ then
7:     Sample $x_i'$ uniformly at random from $[\psi_{left}(x_i), \psi_{right}(x_i)]$
8: else
9:     Sample $x_i'$ uniformly at random from $[-\Delta, \psi_{left}(x_i)] \cup [\psi_{right}(x_i), \Delta]$
10: end if
11: return $x_i'$
```

1) One-Dimensional Perturbation: Alg. 2 takes a single numerical value $x_i \in [-1, 1]$ as input and returns a perturbed value $x_i' \in [-\Delta, \Delta]$, where it confines x_i' to a relatively small domain and allows x_i' to be close to x_i (i.e., $x_i' \in [\psi_{left}(x_i), \psi_{right}(x_i)]$) with reasonably large probability (i.e., $\frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$). For the mean estimation, with at least $1-\beta$ probability, it is shown that $|\frac{1}{n}\sum_{i=1}^{n} x_i' - \frac{1}{n}\sum_{i=1}^{n} x_i| \in O(\frac{\sqrt{log(1/\beta)}}{\epsilon\sqrt{n}})$, which is an error bound with the incurred noise and is asymptotically optimal [24]. In the case that the input domain is not within the range $[-1, 1]$, i.e., $x_i \in [-a, a], a > 0$, $x_i^* = x_i/a$ is calculated and used as input of Alg. 2. Then $x_i^{*\prime}$ is the perturbed output of x_i^*, and $x_i^{*\prime} \cdot a$ is released. It's also assumed that a is the public information in the literature [24].

Algorithm 3 Piecewise Mechanism for Multi-Dimensional Numeric Data [25]

```
1: **Input**: $(\boldsymbol{x}, \epsilon)$
2: $\boldsymbol{x}' \leftarrow < 0, 0, \ldots, 0 >$
3: $k \leftarrow max\{1, min\{d, \lfloor \frac{\epsilon}{2.5} \rfloor\}\}$
4: Sample $k$ values uniformly without replacement from $\{1, 2, \ldots, d\}$
5: for each sampled attribute $j$ do
6:     $x_j' = \frac{d}{k} Alg.\ 2(x_j, \frac{\epsilon}{k})$
7: end for
8: return $\boldsymbol{x}'$
```

2) Multi-Dimensional Perturbation: Alg. 3 takes a d-dimension vector $\boldsymbol{x} \in [-1, 1]^d$ as input and returns a perturbed vector $\boldsymbol{x}' \in [-d \cdot \Delta, d \cdot \Delta]^d$. To minimize the worst-case noise variance, $k < d$ attributes are randomly chosen to be perturbed by Alg. 2. Compared to naive solution of perturbing each attribute with a privacy budget of ϵ/d, Alg. 3 incurs less noise in the mean estimation, where the total amount of noise of the naive solution is in $O(\frac{d\sqrt{logd}}{\epsilon\sqrt{n}})$, which is super-linear to d; and it can be shown that Alg. 3 is still asymptotically optimal, i.e., $E[max_{j \in [1,d]} |\frac{1}{n}\sum_{i=1}^{n} x_{i,j}' - \frac{1}{n}\sum_{i=1}^{n} x_{i,j}|] \in O(\frac{\sqrt{dlog(d/\beta)}}{\epsilon\sqrt{n}})$ with at least $1-\beta$ probability, where the full proof is referred to the original work [25].

III. Problem Definition and Proposed Solution

A. Problem Definition

In this paper, we are interested in developing a privacy-preserving boosting algorithm in a distributed setting. As Fig.1 shows, we consider the following problem: *Given L data owners, each one holds a local dataset $(X^l, Y^l), \boldsymbol{x}_i^l \in \mathbb{R}^d$ and $y_i^l \in \{C_1, C_2, \ldots, C_K\}$, where y_i^l is the class label associated with x_i^l; the untrustworthy data user would like to fit a boosted classifier using $\{(X^l, Y^l), l = 1 : L\}$. In this paper, we assume that the data possessed by the data owner is in numerical representation; for the categorical feature that has k distinct values, it can be transformed to $k-1$ binary features using one-hot encoding and then be proceeded accordingly.* The privacy constraint comes in two-folds at the data owner side, firstly, the data owner is not willing to share the true training samples to the data user in order to protect the data privacy; secondly, any inference of the individual data sample should be prevented from the intermediate exchanged messages. The symbols and notations used in this paper are summarized in Table. I.

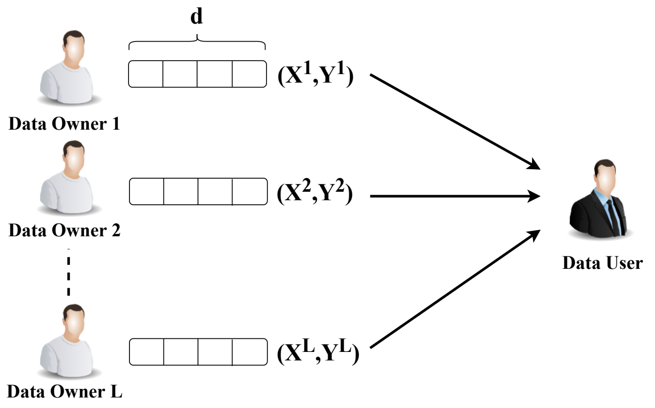


Fig. 1: Problem Overview. Data owner l holds a set of samples (X^l, Y^l), $x_i^l \in \mathbb{R}^d$, and $y_i^l \in \{C_1, C_2, \ldots, C_K\}$ is the label associates with x_i^l; the untrustworthy data user would like to fit a boosted classifier with training samples from all L data owners. The privacy of x_i^l should be protected against the untrustworthy data user.

B. Threat Model

In the problem, we assume that the data owners are honest-but-curious, which implies that every data owner is obliged to follow the protocol, while each one intentionally likes to extend their knowledge during the execution of the protocol; the data user is untrustworthy and is assumed to be the adversary, who intends to infer the private information possessed by the data owner. It is further assumed that the data user holds a subset of data that follows the same distribution, and he has arbitrary background knowledge as well as unlimited computation power. The goal is to enforce the privacy of individual data instances while maintaining the utility of the learned classifier. Furthermore, the untrustworthy data user could behave dishonestly, which would not compromise data owner's privacy with our solution, but it will hurt the utility of the learned classifier. Therefore, it is of the data user's interest to correctly execute the algorithm. As such, our solution protects the data privacy for each data owner. Since the data owner is assumed to be honest-but-curious, data pollution attacks, e.g., data owners maliciously modify their inputs to bias the classifier learned by the data user, are beyond the scope of this paper.

C. Privacy-Preserving Boosting

The proposed privacy-preserving boosting algorithm is run by the data owner and data user collaboratively. Recall that boosting algorithm train a sequence of base learners iteratively, and each base learner is learned by a subset of data owners in our problem. Instead of sharing the true local datasets, our proposed algorithm utilizes the perturbed local datasets to build the base learners. Alg. 4 gives a high-level overview of the privacy-preserving boosting algorithm.

Algorithm 4 Privacy-Preserving Boosting

```
1:  Input: $\{(X^1, Y^1, \epsilon^1), \ldots, (X^L, Y^L, \epsilon^L), (X^u, Y^u)\}$
2:  for l=1:L do
3:      Initialize the observation weights $w_i^l = 1/N^l, i = 1 : N^l$
4:  end for
5:  for m=1:M do
6:      Data user randomly selects a distinct group of $H$ data owners.
7:      for h=1:H do
8:          The $h$-th data owner prepares the local share $V^h \in \mathbb{R}^d$ and perturbs it with $\epsilon^h$ to get $V^{h'}$
9:      end for
10:     Data user collects $V^{h'}$ and computes the base learner $T_m \leftarrow$ Aggregate $(V^{1'}, \ldots, V^{H'})$
11:     $we_m \leftarrow \sum_{i=1}^{N^u} w_i^u \mathbb{I}(y_i^u \neq T_m(\boldsymbol{x_i}^u))$
12:     $ws_m \leftarrow \sum_{i=1}^{N^u} w_i^u$
13:     $err_m \leftarrow \frac{we_m}{ws_m}$
14:     $\alpha_m \leftarrow log\frac{1-err_m}{err_m} + log(K-1)$
15:     if $\alpha_m \leq 0$ then
16:         Go to line 6 and recompute $T_m$
17:     end if
18:     for l=1:L do
19:         $w_i^l \leftarrow w_i^l \cdot exp(\alpha_m \cdot \mathbb{I}(y_i^l \neq T_m(\boldsymbol{x}_i^l))), i = 1 : N^l$
20:     end for
21: end for
22: return $\{(\alpha_1, T_1(\cdot)), \ldots, (\alpha_m, T_m(\cdot))\}$
```

Alg. 4 consists of three stages: 1), line 6-9, a subset of H data owners get randomly selected to participate in current round and each data owner prepares a perturbation of the local share. It should be noted that the representation of the local share is different according to the type of the classifier to boost, and we further demonstrate the perturbation using 3 different types of classifiers in Section IV; 2), line 10-14, data user aggregates the perturbed local shares to train the base learner, and then the data user applies his own local dataset (X^u, Y^u) to derive the error rates err_m and the weights α_m; 3), line 18-20, the data user shares α_m to L data owners and lets each update the weights of local samples. Overall, line 6-20 completes one iteration of the privacy-preserving boosting, and line 22 outputs the final meta-classifier which consists of m base learners.

D. Privacy and Utility Analysis

There are three main changes in our proposed algorithm compared to Alg. 1.

1) To protect the privacy, at line 8 of Alg. 4, the data owner computes a local share and perturbs it accordingly; it's the perturbed value that submitted to the data user rather than the true values, i.e., in a single iteration, the data user only accesses $V^{h'}$, which is the perturbation of V^h.

2) The base learner is trained using the aggregated perturbed shares, and the algorithm to perturb each share satisfies ϵ-LDP. In the non-private scenario, each training samples participates in several iterations. However, under the LDP protection, reusing the perturbed training data continuously consumes the privacy budget and ultimately leaves the data deanonymized. More specifically, suppose the local share released by the data owner in the i-th round satisfies ϵ_i-DP. By the composition property of DP [26], if ϵ-DP is required to be enforced for the data owner's data, then it needs that $\sum_i^m \epsilon_i \leq \epsilon$. Considering that the data owner participates in all m rounds, it becomes $\epsilon_i = \epsilon/m$. Then the amount of the noise contributed by each data owner becomes $O(\frac{m\sqrt{dlogd}}{\epsilon})$, which linearly depends on m. Thus to reduce the injected noise, following the existing studies [24], [27], each data owner only participates in at most one round, i.e., at line 6 of Alg. 4, the data user randomly selects a different subset of H data owners for training.

Depend on the specific classifier to boost, there are various representations of the local share to contribute. To illustrate, assuming the local share is a d-dimension vector, and ϵ is the smallest privacy budget across all data owners. By aggregating H perturbed local shares, the error of the estimated mean is in $O(\frac{\sqrt{dlogd}}{\epsilon\sqrt{H}})$. It is clear that, with ϵ and d fixed, the larger H is, the less noise is in the aggregation. Consequently, a small H might result in overwhelming noise in the aggregation, and the trained base learner might perform no better than random guessing. In Section. IV, we will further demonstrate the determination of H given a fixed privacy level ϵ.

3) One key step in boosting is to increase the weights of the misclassified training samples for $(m+1)$-th iteration, which is determined by α_m. A native solution is to let the data owners calculate the classification errors, i.e., $\sum_i w_i \mathbb{I}(y_i \neq T_m(\boldsymbol{x}_i))$, and then submit the perturbed version to the data user. However, the perturbed classification errors result in a noisy α_m and intensify the utility loss of T_m. Thus to correctly adjust the sample weights, we propose to let the data user use his local copy as a validation set to derive α_m, and then shares α_m to all data owners to update the weights. Although it was proposed to address the overfitting issue, calculating the weights using a validation set [28]–[30] has been introduced already. And we also confirm the effectiveness of using the validation set in the experiments. In particular, it should be noted that a valid α_m should be positive. If $\alpha \leq 0$, at line 15-17 of Alg. 4, the data user simply drops the current iteration and selects another H data owners to re-train T_m. This design ensures that T_m performs better than random guessing.

E. Algorithm Convergence

Both AdaBoost [4] and SAMME [19] are proved to be converged given the assumption that the base learner is better than random guessing. The same assumption is also applied to Alg. 4. The base learner is learned from the aggregated perturbed local shares and the utility is affected by the privacy budget ϵ and the number of participating data owners H. As we explained above, the decision rules are different based on the classifier to boost, and the same ϵ and H might affect the utility of the base learner differently. Therefore, we demonstrate the convergence conditions using 3 specific boosted classifiers in Section IV.

IV. Boosted Classifiers

In this section, we introduce 3 boosted classifiers by extending Alg. 4. The local shares used to build the base learners are different, which we categorize them as *local sample*, *local statistic* and *local classifier*. *Local sample* is the atomic representation of the data possessed by the data owner and *local classifier* refers to a classifier that is fitted by the local samples, which is represented by a vector of model parameters. And the *local statistic* fills the gap between the two end representations. The trade-off between the utility and privacy level for each boosted classifier is presented as well.

A. Local Sample

Nearest Centroid Classifier (NCC) is a simple yet powerful classifier, which makes prediction for an observation by assigning it the label of the class whose centroid is nearest. Formally, given the training data $\{(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{C_1, C_2, \ldots, C_K\}\}$, $\boldsymbol{\mu}_k$ is the centroid of class C_k and it is computed as follows,

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:y_i=C_k} \boldsymbol{x}_i, \tag{4}$$

where N_k is the number of samples that belong to class C_k. Given an observation $\boldsymbol{x}_t$, the prediction is determined by the ℓ^p distance to $\boldsymbol{\mu}_k$, i.e., $\hat{y}_t = \operatorname{argmin}_k ||\boldsymbol{x}_t - \boldsymbol{\mu}_k||_p$. In the paper, we use ℓ^2 to measure the distance between the observation and class centroids. In each iteration, the base NCC is trained by the perturbed local samples shared by randomly selected data owners. The execution of privacy-preserving boosted NCC is presented in Alg. 5.

In Alg. 5, line 8 refers to the perturbation of local samples, and the privacy budget ϵ^h is evenly split to all samples held by the participating data owner, i.e., each data sample is assigned ϵ^h/N^h privacy budget. Line 11 refers to the computation of class centroids, where we use $\boldsymbol{\mu}'_k$ to distinguish it from the true centroid $\boldsymbol{\mu}_k$ that is computed using the non-perturbed local samples, i.e., $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:y_i=C_k} \sum_h w_i^h \cdot \boldsymbol{x}_i^h$. The computation of $we_m, ws_m, err_m, \alpha_m$ at line 13 is same as Alg. 4.

Algorithm 5 Privacy-Preserving Boosted NCC

```
1: Input: ${(X^1, Y^1, \epsilon^1), \dots, (X^L, Y^L, \epsilon^L), (X^u, Y^u)}$
2: for l=1:L do
3:     Initialize the observation weights $w_i^l = 1/N^l, i = 1 : N^l$
4: end for
5: for m=1:M do
6:     Data user randomly selects a distinct group of H data owners
7:     for h=1:H do
8:         $\boldsymbol{x}_i^{h\prime} \leftarrow Alg.\ 3(\boldsymbol{x}_i^h, \epsilon^h/N^h), i = 1 : N^h$
9:         $\{(w_i^h \cdot \boldsymbol{x}_i^{h'}, y_i^h) : i = 1 : N^h\}$ is submitted to the Data User
10:    end for
11:    $\boldsymbol{\mu}_k' \leftarrow \frac{1}{N_k}\sum_{i:y_i=C_k}\sum_h w_i^h \cdot \boldsymbol{x}_i^{h\prime}$, where $N_k$ is the number of samples that belong to class $C_k$
12:    $T_m(\boldsymbol{x}) \leftarrow \text{argmin}_k ||\boldsymbol{x} - \boldsymbol{\mu}_k'||_2$
13:    Update $we_m, ws_m, err_m, \alpha_m$
14:    if $\alpha_m \le 0$ then
15:        Go to line 6 and recompute $T_m$
16:    end if
17:    for l=1:L do
18:        $w_i^l \leftarrow w_i^l \cdot exp(\alpha_m \cdot \mathbb{I}(y_i^l \neq T_m(\boldsymbol{x}_i^l))), i = 1 : N^l$
19:    end for
20: end for
21: return ${(\alpha_1, T_1(\cdot)), \dots, (\alpha_m, T_m(\cdot))}$
```

To ensure the convergence of Alg. 5, the base NCC needs to perform better than random guessing, which requires $||\boldsymbol{\mu}_i - \boldsymbol{\mu}_i'||_2 \in O(||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2), \forall i \neq j$. Informally, $\forall j \neq K$,

$$Pr(||\boldsymbol{x}_t - \boldsymbol{\mu}_K'||_2 < ||\boldsymbol{x}_t - \boldsymbol{\mu}_j'||_2 | y_t = C_K) > \frac{1}{K} \tag{5}$$

$$\Rightarrow Pr(||\boldsymbol{x}_t - \boldsymbol{\mu}_K + \boldsymbol{\mu}_K - \boldsymbol{\mu}_K'||_2 < ||\boldsymbol{x}_t - \boldsymbol{\mu}_K + \boldsymbol{\mu}_K - \boldsymbol{\mu}_j + \boldsymbol{\mu}_j - \boldsymbol{\mu}_j'||_2 | y_t = C_K) > \frac{1}{K} \tag{6}$$

$$\Rightarrow Pr(||\boldsymbol{\mu}_K - \boldsymbol{\mu}_K'||_2 < ||\boldsymbol{\mu}_K - \boldsymbol{\mu}_j + \boldsymbol{\mu}_j - \boldsymbol{\mu}_j'||_2 | y_t = C_K) > \frac{1}{K}. \tag{7}$$

Otherwise, the injected noise is overwhelming and the resulted base NCC performs no better than random guessing.

B. Local Statistic

Another type of the classifier is trained by utilizing the statistical information of the local samples, e.g., the decision stump classifier (Equ.1), which is fitted by the aggregation of the impurity measurements from data owners.

First, we introduce the training procedure of the DT by utilizing the misclassification impurity. Assuming it's a binary classification problem and the attribute A^j contains the binary categorical values, the cross table of A^j is a statistical summary and is represented by $S^j = \begin{pmatrix} s_{0,0}^j & s_{0,1}^j \\ s_{1,0}^j & s_{1,1}^j \end{pmatrix}$, where $s_{0,1}^j$ denotes the weighted sum of examples whose $A^j = 0$ and belong to C_1. The misclassification error impurity of the $A^j = 0$ is defined as $1 - max\{s_{0,0}^j, s_{0,1}^j\}/s_0^j$, where s_0^j is the weighted sum of samples that $A^j = 0$ regardless the belonging class. For A^j, maximizing its gain is equivalent to minimizing the weighted sum of impurities [21]:

$$\underset{j}{\text{argmin}}\left\{\frac{s_0^j}{|S^j|}\Big[1 - \frac{max\{s_{0,0}^j, s_{0,1}^j\}}{s_0^j}\Big] + \frac{s_1^j}{|S^j|}\Big[1 - \frac{max\{s_{1,0}^j, s_{1,1}^j\}}{s_1^j}\Big]\right\}$$
$$\iff \underset{j}{\text{argmax}}\{|s_{0,0}^j - s_{0,1}^j| + |s_{1,0}^j - s_{1,1}^j|\}. \tag{8}$$

For ease of explanation, we define $\phi^j = |s_{0,0}^j - s_{0,1}^j| + |s_{1,0}^j - s_{1,1}^j|$, and A^j is the best attribute if it has the minimum classification error impurity and ϕ^j is maximum correspondingly, i.e., $|s_{0,0}^i - s_{0,1}^i| + |s_{1,0}^i - s_{1,1}^i| \le |s_{0,0}^j - s_{0,1}^j| + |s_{1,0}^j - s_{1,1}^j|, \forall i \neq j$. To determine the attribute to split, the cross tables of all attributes are perturbed and aggregated. More specifically, for each data owner, $(s_{0,0}^j - s_{0,1}^j)'$ and $(s_{1,0}^j - s_{1,1}^j)'$ are the perturbed values of $(s_{0,0}^j - s_{0,1}^j)$ and $(s_{1,0}^j - s_{1,1}^j)$; and $\bar{\phi^j}' = |\frac{\sum_h (s_{0,0}^j - s_{0,1}^j)'}{H}| + |\frac{\sum_h (s_{1,0}^j - s_{1,1}^j)'}{H}|$ is the aggregation of H statistical summaries from the participating data owners in each round. Consequently, A^j is the best attribute to split if $\bar{\phi^i}' < \bar{\phi^j}', \forall i \neq j$. Alg. 6 presents the execution details.

Algorithm 6 Privacy-Preserving BDT

```
1: Input: ${(X^1, Y^1, \epsilon^1), \dots, (X^L, Y^L, \epsilon^L), (X^u, Y^u)}$
2: for l=1:L do
3:     Initialize the observation weights $w_i^l = 1/N^l, i = 1 : N^l$
4: end for
5: for m=1:M do
6:     Data user randomly selects a distinct group of H data owners
7:     for h=1:H do
8:         $S^j \leftarrow \begin{pmatrix} s_{0,0}^j & s_{0,1}^j \\ s_{1,0}^j & s_{1,1}^j \end{pmatrix}$, where $s_{0,1}^j$ denotes the sum of weights of examples that have $A^j = 0$ and belong to $C_1$
9:         $\Delta s^h \leftarrow \{(s_{0,0}^j - s_{0,1}^j, s_{1,0}^j - s_{1,1}^j), j = 1 : d\}$
10:        $\Delta s^{h'} \leftarrow Alg.\ 3(Flatten(\Delta s^h), \epsilon^h), Flatten(\Delta s^h) \in \mathbb{R}^{2d}$
11:        $\Delta s^{h'}$ is submitted to the Data User
12:    end for
13:    $\bar{\phi^j}' \leftarrow |\frac{\sum_h [(s_{0,0}^j - s_{0,1}^j)']}{H}| + |\frac{\sum_h [(s_{1,0}^j - s_{1,1}^j)']}{H}|, j = 1 : d$
14:    $T_m \leftarrow f(\boldsymbol{x}|j, \eta)$, given $\bar{\phi^i}' < \bar{\phi^j}', \forall i \neq j$
15:    Update $we_m, ws_m, err_m, \alpha_m$
16:    if $\alpha_m \le 0$ then
17:        Go to line 6 and recompute $T_m$
18:    end if
19:    for l=1:L do
20:        $w_i^l \leftarrow w_i^l \cdot exp(\alpha_m \cdot \mathbb{I}(y_i^l \neq T_m(\boldsymbol{x}_i^l))), i = 1 : N^l$
21:    end for
22: end for
23: return ${(\alpha_1, T_1(\cdot)), \dots, (\alpha_m, T_m(\cdot))}$
```

In line 10 of Alg. 6, the cross tables of d attributes are flattened and is composed as one vector, e.g., $Flatten(\Delta s^h) \in \mathbb{R}^{2d}$. Then the vector is perturbed by Alg. 3 with ϵ^h, as it is the only parameter vector transmitted to the data user, and the perturbation makes statistic summaries indistinguishable from each other.

Similar to Alg. 5, the convergence of Alg. 6 requires that the probability of picking best feature is better than random guessing for each decision stump, which implies that $(\phi^j - \phi^{j\prime}) \in O(\phi^i - \phi^j), \forall i \neq j$. Such condition ensures that the best feature can still be effectively selected in each round.

For the feature has continuous values, the splitting strategy can be easily adapted. Assuming A^o represents a feature that has continuous values, the data user pre-computes the threshold η using his local dataset and shares η to data owners. For the participating data owners, the cross table can be computed based on η, where $S^o = \begin{pmatrix} s^o_{<\eta,0} & s^o_{<\eta,1} \\ s^o_{\geq\eta,0} & s^o_{\geq\eta,1} \end{pmatrix}$, $s^o_{<\eta,1}$ denotes the weighted sum of samples with $A^o < \eta$ and belong to C_1. Once the cross tables for all attributes are ready, the calculation of the misclassification error impurity is same as the treatment of the binary attribute described above.

Algorithm 7 Privacy-Preserving Boosted LR

```
1:  Input: $\{(X^1, Y^1, \epsilon^1), \ldots, (X^L, Y^L, \epsilon^L), (X^u, Y^u)\}$
2:  for l=1:L do
3:      Initialize the observation weights $w^l_i = 1/N^l, i = 1 : N^l$
4:  end for
5:  for m=1:M do
6:      Data user randomly selects a distinct group of H data owners
7:      for h=1:H do
8:          $\hat{\boldsymbol{\theta}}^h \leftarrow \text{argmin}_{\boldsymbol{\theta}^h} \frac{1}{N^h}(\sum_{i=1}^{N^h} \mathcal{L}(\boldsymbol{\theta}^h; w^h_i; \boldsymbol{x_i}^h; y^h_i)$,
            where $\mathcal{L}(\boldsymbol{\theta}; w; \boldsymbol{x}; y) = -wylog(sigmoid(\boldsymbol{\theta}^T\boldsymbol{x})) - w(1-y)(1-log(sigmoid(\boldsymbol{\theta}^T\boldsymbol{x}))), \boldsymbol{\theta} \in \mathbb{R}^d$
9:          $\hat{\boldsymbol{\theta}}^{h\prime} \leftarrow Alg.\ 3(\hat{\boldsymbol{\theta}}^h, \epsilon^h)$
10:         $\hat{\boldsymbol{\theta}}^{h\prime}$ is submitted to the Data User
11:     end for
12:     $\boldsymbol{\theta}'_m \leftarrow \frac{1}{H}\sum_h \hat{\boldsymbol{\theta}}^{h\prime}$
13:     $T_m \leftarrow sigmoid(\boldsymbol{\theta}'^T_m \boldsymbol{x})$
14:     Update $we_m, ws_m, err_m, \alpha_m$
15:     if $\alpha_m \leq 0$ then
16:         Go to line 6 and recompute $T_m$
17:     end if
18:     for l=1:L do
19:         $w^l_i \leftarrow w^l_i \cdot exp(\alpha_m \cdot \mathbb{I}(y^l_i \neq T_m(\boldsymbol{x}^l_i))), i = 1 : N^l$
20:     end for
21: end for
22: return $\{(\alpha_1, T_1(\cdot)), \ldots, (\alpha_m, T_m(\cdot))\}$
```

C. Local Classifier

The aggregation of local classifiers is a quite popular approach in the literature [31]–[35]. In such scenario, the data owner trains a classifier using his own local dataset and the final classifier is constructed by aggregating the classifier parameters from multiple data owners. Formally, assuming the classifier is trained by optimizing the loss function $\mathcal{L}(\cdot)$, where the parameter vector $\boldsymbol{\theta}$ of $\mathcal{L}(\cdot)$ is estimated as:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}[\frac{1}{N}(\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{\theta}; w_i; \boldsymbol{x_i}; y_i)]. \quad (9)$$

In our problem, $\hat{\boldsymbol{\theta}}^h$ is the parameter of the local classifier learned by data owner h, and the base learner in each iteration is obtained by aggregating multiple perturbed ones:

$$\boldsymbol{\theta}' = \frac{1}{H}\sum_h \hat{\boldsymbol{\theta}}^{h\prime}. \quad (10)$$

Alg. 7 presents the execution details of boosting the logistic regression classifier privately. $\boldsymbol{\theta}^h$ is learned at line 8 and it is perturbed at line 9, i.e., $\hat{\boldsymbol{\theta}}^{h\prime}$ = Alg. 3($\hat{\boldsymbol{\theta}}^h, \epsilon^h$). Similar to the setting of *local statistic*, $\hat{\boldsymbol{\theta}}^h$ is perturbed using budget ϵ^h and the resulted model parameters are indistinguishable from the data owners.

For multi-class classification problem, the one-versus-all strategy is adopted for linear regression; and the performance of the base learner should be better than $1/2$ to allow Alg. 7 to converge, which implies $sgn(\boldsymbol{\theta}'^T\boldsymbol{x})$ needs to same as $sgn(\boldsymbol{\theta}^T\boldsymbol{x})$.

The privacy protection of the *local statistic* and *local classifier* is slightly different from the *local sample* scenario. To train the boosted NCC, individual sample $\boldsymbol{x}^h_i$ in (X^h, Y^h) is perturbed by Alg. 3, which satisfies LDP. By receiving the perturbed samples, the data user cannot distinguish whether the true tuple is $\boldsymbol{x}^h_i$ or $\boldsymbol{x}^{h\prime}_i$ with high confidence. While for BDT and boosted LR, the perturbation happens on either the cross tables or the parameters of learned LR, which are the functional outputs of (X^h, Y^h). It can be proved that Alg. 3 satisfies DP, for instance, assuming (X^h, Y^h) and $(X^h, Y^h)^*$ are two neighboring datasets, $f(\cdot)$ refers to the learned LR and $g()$ refers to Alg. 3, it shows that $Pr[g(f((X^h, Y^h)) \in \mathcal{O}] \leq e^\epsilon Pr[g(f((X^h, Y^h)^*) \in \mathcal{O}]$ by the definition of LDP; and the sensitivity of $f(\cdot)$ is given by $||f(X^h, Y^h) - f(X^h, Y^h)^*||_1 \leq 2d$, since $f()$ is scaled to [-1,1] [25].

V. Experiment

In this section, we evaluate the 3 proposed boosted classifiers in Section. IV. The utility of the boosted classifiers are assessed in terms of prediction accuracy over the real and synthetic datasets, i.e., the MNIST, Fashion-MNIST[1], and Brazil dataset. The descriptions of the datasets are given below,

The MNIST dataset contains 28×28 grayscale images of handwritten digits from 0 to 9, which has 60,000 samples for training and 10,000 samples for testing. To have large enough volume of samples for multi-round training, we apply the random shifts and rotations from the Image Augmentation API[2] to enlarge the size of the training samples. Among all digits, the confusable digit "4" and "9" are selected for binary

[1]https://github.com/zalandoresearch/fashion-mnist

[2]https://keras.io/api/preprocessing/image/

classification in the experiment. For each class, we use 20,000 samples for training and 1,000 samples for evaluation, and there are extra 1,000 samples reserved for the data user.

The Fashion-MNIST dataset contains the grayscale article images of 10 classes. The dataset is intended to replace the overused MNIST dataset, and it shares the same image size and structure as the MNIST dataset. Similar to MNIST augmentation, we enlarge the size of training images and pick the confusable class "T-shirt/top" and "shirt" for binary classification. For each class, we use 20,000 samples for training and 1,000 samples for evaluation, and there are extra 1,000 samples reserved for the data user.

The Brazil dataset contains the census records in Brazil between 1970 and 2010, which are from the Integrated Public Use Microdata Series [36] (IPUMS). The dataset has 10 attributes, 2 of them are numerical (e.g., age and totalIncome) and the rest of them are categorical. In the experiment, the totalIncome is used as the dependent variable and is converted to binary attribute by mapping the value larger than mean value to 1, and 0 otherwise [25]. The categorical attributes with k distinct values are transformed to k-1 binary attributes. After the transformation, the dimension of the dataset becomes 67. We use 12×10^6 samples for training and 2×10^6 samples for evaluation, and there are 100,000 samples reserved for the data user.

The synthetic dataset is generated using the scikit-learn[3] API, which contains 2 classes and 20 real-valued attributes; 10 of them are the meaningful attributes, the rest of them are generated as non-informative features that are the linear combinations of the 10 meaningful ones. There are 10^6 samples generated in total and the number of positive samples is equal to the negative samples. In the experiment, 75% of samples are used for training and 20% are used for testing, the other 5% samples are reserved for the data user.

For a better study of the trade-off between the utility and privacy, we implement the boosted classifiers with two other perturbation methods, i.e., Laplace mechanism [37] and Duchi et al.'s mechanism. The Laplace mechanism perturbs the single numerical value by adding a random value from $Lap(0, 2/\epsilon)$, where $Lap(\mu, \lambda)$ follows a Laplace distribution that has 0 mean and scale λ. The estimated mean of n perturbed values is unbiased and the error is in $\frac{1}{\epsilon\sqrt{n}}$. However, to perturb multiple attributes, all attributes have to evenly share the total budget ϵ, and the error of the estimated mean of the multi-dimensional vector is super-linear to d. Duchi et al. proposed a solution to perturb multiple numerical attributes [24], which is also asymptotically optimal but has a larger constant than Alg. 3. For consistency, the term *Laplace* is referred to the Laplace mechanism, the term *Duchi* is referred to the Duchi et al.'s mechanism and *PM* is referred to Alg. 3. For each boosted classifier, we prefix the perturbation method with classifier for easy interpretation, e.g., Lap-NCC refers to the NCC that is trained by the samples perturbed by the Laplace mechanism, and PM-BDT refers to the BDT trained by the samples perturbed by PM. In the experiment, **we further assume that the same privacy budget is assigned to all data owners.**

[3]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

Given the fixed privacy budget, the larger dimension of the dataset is, the more noise is resulted in. To study the utility impact of the number of dimensions, the MNIST and Fashion-MNIST dataset are transformed to the histogram frequency. More specifically, the Histogram of Oriented Gradients (HOG) descriptors are first computed from the image dataset, which provides the histograms of directions of gradients that are used to capture the edges and corners of an image. A Bag of Visual Words (BOVW) model is then associated with the HOG descriptors, where the K-Means clustering is performed over the generated descriptors and the center of clusters are treated as the visual dictionary's vocabularies. Finally, the histogram of frequency is calculated for each visual word, and the image is transformed to the histogram representation. By using BOVW, we have the flexibility to decide the number of visual vocabularies, **in our experiment, the MNIST is transformed to 50 dimensions and Fashion-MNIST is transformed to 100 dimensions**.

To measure the classification accuracy, we calculate the accuracy and misclassification rate for base learners and boosted classifiers, i.e.,

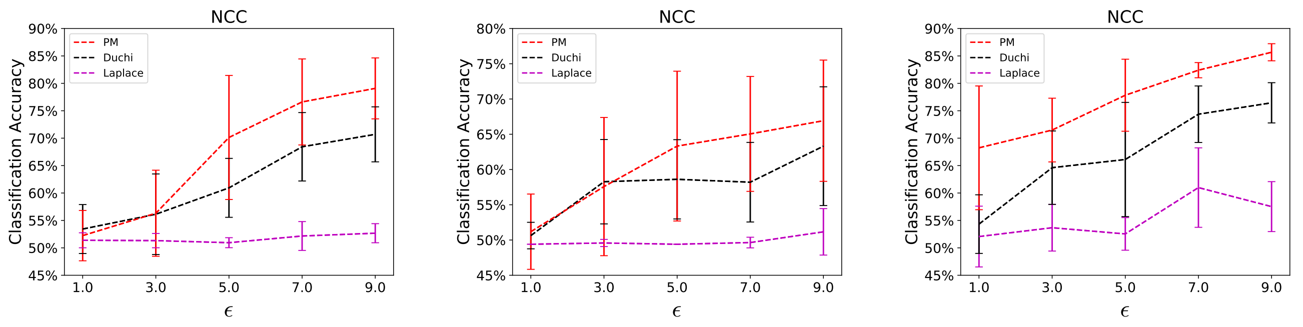
$$\textit{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{11}$$

$$\textit{misclassification rate} = 1 - \textit{accuracy}, \tag{12}$$

where TP refers to True Positive, TN refers to True Negative, FP refers to False Positive and FN refers to False Negative.

A. Boosted Nearest Centroid Classifier

The privacy-preserving boosted NCC is evaluated over the MNIST, Fashion-MNIST and the synthetic dataset. For MNIST and Fashion-MNIST, we simulate 1,000 data owners to participate in each iteration and assign 4 samples to each data owner. For the synthetic dataset, we simulate 2,000 data owners in each round and assign 4 samples to each data owner. In Alg. 5, the class centroids are computed via the collected perturbed local samples from the participating data owners, and the estimated error is in $O(\frac{\sqrt{dlogd}}{\epsilon\sqrt{\sum_h N^h}})$, where d is the dimension of the data, ϵ is the privacy budget for a single sample and $\sum_h N^h$ is the number of collected training samples in one round, for MNIST, $d = 50$ and $\sum_h N^h = 4,000$. Fig. 2 shows the classification accuracy of the first base NCC under various privacy budgets, note that the ϵ in the x-axis denotes the privacy budget assigned for each data owner, and the local samples split the privacy budget equally, e.g., each sample receives $\epsilon/4$ budget since there are 4 samples possessed by each data owner. In Fig. 2, for all 3 datasets, PM-NCC has the highest accuracies for each privacy level, which indicates PM-NCC preserves the most utility comparing to 2 other perturbation methods. It also shows that when ϵ is small, e.g., $\epsilon = 1.0$, PM-NCC and Duchi-NCC perform no better than random guessing, i.e., the accuracies in average are about 0.5; and the accuracies of both base learners increase as the privacy budgets become larger. The pattern is as expected, since the error of the centroids estimation, i.e.,



(a) MNIST (b) Fashion-MNIST (c) Synthetic

Fig. 2: Classification accuracy of the first NCC base learner w.r.t. various privacy budgets. For MNIST and Fashion-MNIST, there are 1,000 data owners participated in a single round and each data owner holds 4 samples. For synthetic dataset, there are 2,000 data owners in a single round and each data owner holds 4 samples. For MNIST and Fashion-MNIST, the non-private NCC classification accuracies are 91%; for the synthetic dataset, the non-private NCC classification accuracy is 87%.

TABLE II: ℓ_p^2 of the first iteration of Lap-NCC, Duchi-NCC and PM-NCC respectively. The ℓ_b^2 of MNIST is 1.01; The ℓ_b^2 of Fashion-MNIST is 1.12; The ℓ_b^2 of the synthetic dataset is 0.5. The smallest ℓ_p^2 is highlighted in bold for each dataset.

	Perturbation \ ϵ	1.0	3.0	5.0	7.0	9.0
MNIST	Laplace	88.950 + 6.466	28.310 + 1.653	17.623 + 1.235	12.504 + 0.897	9.866 + 0.856
	Duchi	33.625 + 1.259	9.501 + 0.335	4.899 + 0.084	3.136 + 0.164	2.247 + 0.084
	PM	**14.088 + 0.878**	**4.518 + 0.210**	**2.965 + 0.129**	**2.449 + 0.140**	**2.111 + 0.106**
Fashion-MNIST	Laplace	256.974 + 15.871	82.804 + 4.038	50.814 + 2.293	35.740 + 1.439	27.713 + 1.256
	Duchi	51.749 + 1.507	15.223 + 0.330	8.096 + 0.350	5.166 + 0.126	3.995 + 0.213
	PM	**24.850 + 1.349**	**8.575 + 0.412**	**5.451 + 0.344**	**4.383 + 0.155**	**3.910 + 0.178**
Synthetic	Laplace	15.681 + 1.375	5.314 + 0.471	3.230 + 0.354	2.464 + 0.214	1.904 + 0.133
	Duchi	3.905 + 0.353	1.307 + 0.158	0.927 + 0.077	0.650 + 0.066	0.594 + 0.045
	PM	**3.266 + 0.305**	**0.970 + 0.069**	**0.553 + 0.060**	**0.372 + 0.042**	**0.268 + 0.034**

$||\boldsymbol{\mu}_k - \boldsymbol{\mu}'_k||_2$, becomes smaller as ϵ becomes larger. To better illustrate the point, Table. II presents the estimated error for all privacy levels.

To recap, we use ℓ_b^2 to represent the distance between the true centroids of classes, i.e., $\ell_b^2 = ||\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1||_2$, and ℓ_b^2 is fixed for a given dataset. Similarly, we use ℓ_p^2 to represent the distance between the true centroids and the perturbed ones, i.e., $\ell_p^2 = \frac{||\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_0||_2 + ||\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_1||_2}{2}$, and Table. II presents the averaged ℓ_p^2 for all privacy levels in 10 runs. For MNIST, the ℓ_b^2 is about 1.01, and the PM-NCC consistently has the smallest ℓ_p^2 compared to 2 other perturbation methods across all privacy levels, i.e., ℓ_p^2 of Lap-NCC is about 6 times larger than PM-NCC, and ℓ_p^2 of Duchi-NCC is nearly 2 times larger than PM-NCC when $\epsilon = 1.0$. However, given the smallest ϵ(i.e.,ϵ =1.0), ℓ_p^2 of PM-NCC is about 14.09, which is significantly greater than 1.01, which implies that the noise dominates the base learner and the output of the base learner is like random guessing. As ϵ increases, ℓ_p^2 drops quickly, and the classification accuracy gets improved. Even though ℓ_p^2(i.e.,$\ell_p^2 \approx 2.97$) is not strictly less than ℓ_b^2 when $\epsilon = 5.0$, it can be seen that the accuracy is about 70% while the variance is still large. As we explained in Section. IV-A, the base NCC learned at this privacy level should be able to converge and we confirm it in the next experiments. For the synthetic dataset, $\ell_b^2 = 0.5$ and $\ell_p^2 \leq 0.372$ when $\epsilon \geq 7.0$, and the accuracy of PM-NCC is almost identical to the non-private learner and it implies that the noise is small enough and it doesn't affect the utility the learner.

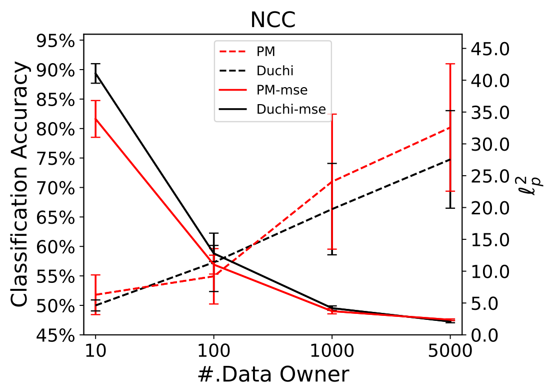
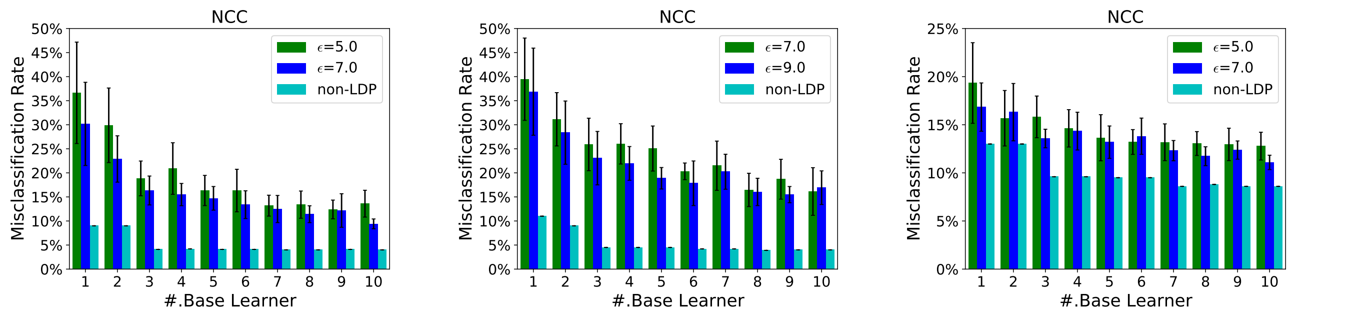


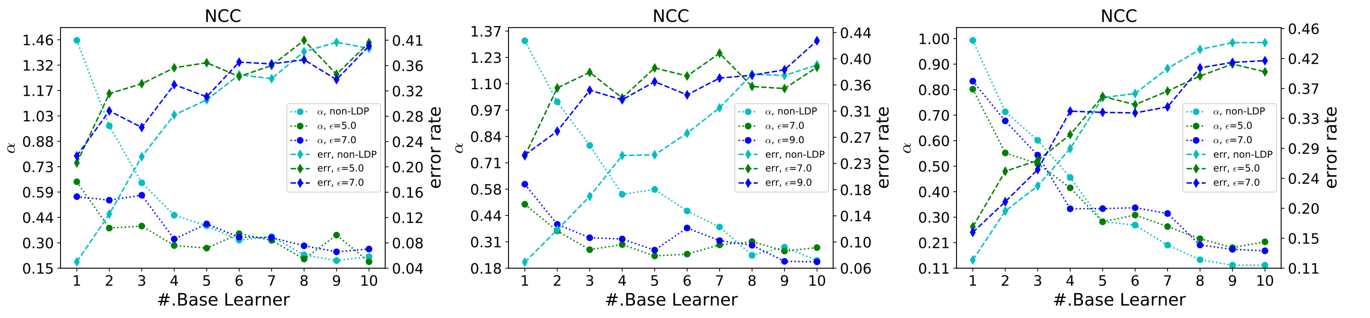
Fig. 3: NCC classification accuracy & ℓ_p^2 w.r.t. various numbers of data owners in MNIST, where each data owner holds 4 samples and ϵ =5.0.

In the aggregation, the more data owners participate, the more perturbed samples are collected and the less noise would be. To illustrate this point, Fig. 3 presents the classification accuracy and ℓ_p^2 in terms of various amounts of data owners in MNIST. The left y-axis indicates the classification accuracy and the right y-axis shows ℓ_p^2. In Fig. 3, the privacy level is fixed at 5.0, and it is readily to see the improvement of accuracy by increasing the number of data owners (resp. number of perturbed samples) from 100 (resp. 400) to 1000 (resp. 4000), which shows the rough lower bound of the number



(a) MNIST (b) Fashion-MNIST (c) Synthetic

Fig. 4: Boosted PM-NCC misclassification rate w.r.t. various numbers of base learners. For both MNIST and Fashion-MNIST, there are 1,000 distinct data owners participating in each round and each data owner holds 4 samples, and the PM-NCC is boosted in 10 rounds in total. For synthetic dataset, there are 2,000 distinct data owners participating in each round and each data owner holds 4 samples.

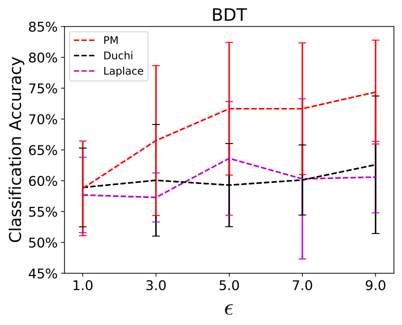


(a) MNIST (b) Fashion-MNIST (c) Synthetic

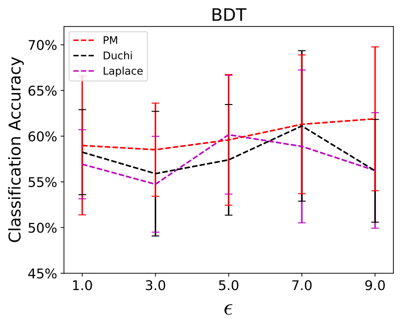
Fig. 5: PM-NCC α_m and err_m w.r.t. the number of base learners. For all 3 datasets, the PM-NCC is boosted in 10 iterations, and the variance is omitted from the figure.

of needed samples to make PM-NCC beat random guessing. For the computation complexity, there are $N^h \times H$ samples collected in one round and the perturbation of each sample has the complexity of $O(d)$, thus the overall computation cost in one round is in $O(N^h Hd)$. Comparing Duchi-NCC to PM-NCC, it shows that PM-NCC consistently has a lower ℓ_p^2 than Duchi-NCC, even though their errors are close to each other. Combining Fig. 3 with Table. II, it implies that both perturbation methods are asymptotically similar while the error bound of Duchi has a larger constant than PM.

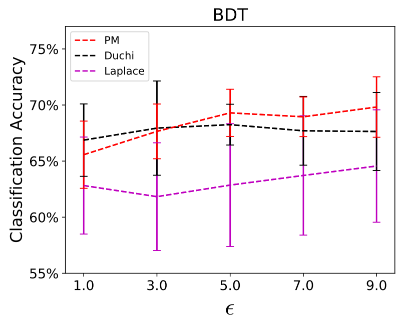
After studying the performance of a single base learner, the PM-NCC is boosted in multiple iterations. In the experiment, there are 1,000 data owners in each iteration, and PM-NCC is boosted in 10 iterations. The misclassification rate is plotted in Fig. 4. The non-private NCC is also boosted in 10 iterations and the result confirms the improved classification performance of boosted NCC. For MNIST, the misclassification rate of the non-private boosted NCC is reduced from 9% to 4%; for Fashion-MNIST, the misclassification rate of the non-private boosted NCC is reduced from 11% to 5%; for the synthetic dataset, the misclassification rate of the non-private version is reduced from 13% to 9%. As analyzed in Section. IV-A, Alg. 5 is able to converge only if the base learner performs better than random guessing. From Fig. 2, we can roughly observe the minimum privacy budget to satisfy this assumption for each dataset, for instance, for MNIST, given the fixed number of perturbed samples, the accuracy of the PM-BNCC is about 70% when $\epsilon = 5.0$. Thus for each dataset, we select the privacy budgets which ensures that the classification accuracy of base NCC is above 50%. In Fig. 4, for MNIST, the misclassification rate of PM-BNCC is reduced from 37% (resp. 30%) to 17% (resp. 9%) when $\epsilon = 5.0$ (resp. 7.0); similarly, for Fashion-MNIST, the misclassification rate is reduced from 39% (resp. 37%) to 19% (resp. 20%) at $\epsilon = 7.0$ (resp. 9.0), and the performances are close under 2 privacy levels. For the synthetic dataset, the misclassification rate is reduced from 19% (resp. 17%) to 14% (resp. 12%) given $\epsilon = 5.0$ (resp. 7.0). In general, it shows that the performance of the PM-BNCC get improved when the privacy levels are relaxed, which is due to the reduced noise in the estimated mean. To have a view of the convergence speed, Fig. 5 plots the weight(α_m) and error rate(err_m) of each base NCC, which are referred at line 13 in Alg. 5. For MNIST and Fashion-MNIST, the err_m of PM-NCC gradually increases from 0.2 to 0.5 in 8 iterations; in the meanwhile, the corresponding α_m gradually decreases to 0, which implies that more iterations would add



(a) MNIST



(b) Fashion-MNIST



(c) Synthetic

Fig. 6: Decision stump (DT) classification accuracy w.r.t. various privacy budgets. For MNIST and Fashion-MNIST, there are 200 data owners participated in a single round and each data owner holds 20 samples. For Synthetic dataset, there are 1,000 data owners participated in a single round and each data owner holds 80 samples.

TABLE III: Δ_{pim} for all 3 perturbation methods. The Δ_{im} of MNIST is 0.2, Δ_{im} of Fashion-MNIST is 0.17 and the Δ_{im} of the synthetic dataset is 0.3. The smallest Δ_{pim} is highlighted in bold for each dataset.

	Perturbation \ ϵ	1.0	3.0	5.0	7.0	9.0
MNIST	Laplace	1290.09 + 242.05	141.96 + 18.39	47.95 + 6.54	26.18 + 2.82	14.58 + 1.95
	Duchi	12.24 + 1.86	3.03 + 0.60	2.56 + 0.43	2.34 + 0.29	2.40 + 0.50
	PM	**5.32 + 1.06**	**0.27 + 0.05**	**0.22 + 0.02**	**0.20 + 0.02**	**0.18 + 0.02**
Fashion-MNIST	Laplace	5613.86 + 539.39	559.66 + 69.57	200.22 + 19.55	103.88 + 8.77	60.32 + 5.97
	Duchi	25.13 + 2.18	6.03 + 0.37	5.24 + 0.55	4.99 + 0.51	5.25 + 0.51
	PM	**10.71 + 0.97**	**0.63 + 0.08**	**0.48 + 0.06**	**0.22 + 0.03**	**0.19 + 0.01**
Synthetic	Laplace	36.76 + 8.86	3.53 + 0.91	1.06 + 0.30	0.53 + 0.09	0.21 + 0.05
	Duchi	0.81 + 0.28	0.23 + 0.07	0.18 + 0.03	0.15 + 0.04	0.13 + 0.04
	PM	**0.807 + 0.213**	**0.029 + 0.008**	**0.037 + 0.007**	**0.013 + 0.002**	**0.013 + 0.004**

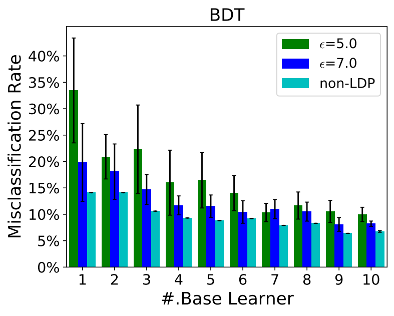
neutral gains. For the synthetic dataset, it shows that the PM-BNCC has an almost identical converging speed to non-private BNCC, which confirms that the PM-NCC effectively maintains the prediction capacity at the given privacy level ($\epsilon \geq 5.0$).

B. Boosted Decision Tree

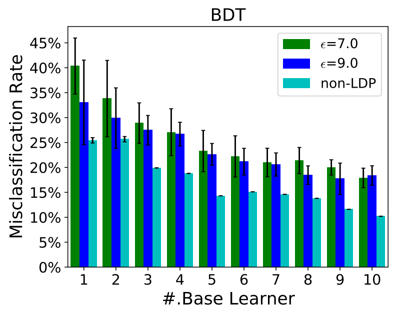
The evaluation of the BDT classifier is performed over the MNIST, Fashion-MNIST and the synthetic dataset. For MNIST and Fashion-MNIST, there are 200 data owners participated in a single round and each data owner holds 20 samples; for the synthetic dataset, there are 1,000 data owners participated in a single round and each data owner holds 80 samples. In each round, the best feature to split is determined by the aggregated misclassification error impurities. The overwhelming noise in the perturbed cross tables could disturb the aggregated impurities and return a random feature instead. As we analyzed in Section. IV-B, to ensure the true best feature to be decided, the injected noise should small enough to not disturb the order of the impurities, i.e., $(|s^j_{0,0}-s^j_{0,1}|+|s^j_{1,0}-s^j_{1,1}|)-(|(s^j_{0,0}-s^j_{0,1})'|+|(s^j_{1,0}-s^j_{1,1})'|) \in O(|s^i_{0,0}-s^i_{0,1}|+|s^i_{1,0}-s^i_{1,1}|-|s^j_{0,0}-s^j_{0,1}|+|s^j_{1,0}-s^j_{1,1}|), \forall i \neq j$. For better illustration, we define the Δ_{im} as the averaged difference of impurities between any 2 features, $\Delta_{im} = \frac{\sum_{i,j}|s^i_{0,0}-s^i_{0,1}|+|s^i_{1,0}-s^i_{1,1}|-(|s^j_{0,0}-s^j_{0,1}|+|s^j_{1,0}-s^j_{1,1}|), \forall i \neq j}{\binom{d}{2}}$, where i, j indicates ith and jth feature, and d is the number of features, and $\Delta_{pim} = \frac{\sum_{j=1}^{d}|s^j_{0,0}-s^j_{0,1}|+|s^j_{1,0}-s^j_{1,1}|-(|(s^j_{0,0}-s^j_{0,1})'|+|(s^j_{1,0}-s^j_{1,1})'|)}{d}$. Table. III presents Δ_{pim} across various privacy levels, and the classification accuracy of the first DT is given in Fig. 6.

For MNIST and Fashion-MNIST, Δ_{im} is around 0.2 and 0.17 respectively. In Table. III, Δ_{pim} of Lap-DT and Duchi-DT are both overwhelming even when $\epsilon = 9.0$ such that the true best feature is hardly estimated, that's why the classification accuracy is similar to random guessing across all privacy levels. For MNIST, Δ_{pim} of PM-DT approximates to Δ_{im} when $\epsilon \geq 5.0$, and the probability to return the true best feature increases accordingly; for Fashion-MNIST, the Δ_{pim} of PM-DT approximates to Δ_{im} when $\epsilon \geq 9.0$, thus its classification accuracy doesn't beat random guessing significantly. For MNIST and Fashion-MNIST, there is a relatively large variance observed for the estimated accuracy of PM-DT in Fig. 6 and it is as expected. The reason is that the error in the aggregation of cross tables leads to a random feature to be split, and the accuracy of the PM-DT learned by such random feature is most likely around 0.5. For the synthetic dataset, the Δ_{pim} of Duchi-DT and PM-DT are smaller than Δ_{im} when $\epsilon \geq 3.0$, and the classification accuracies are quite stable and close approximate to the measurement of non-private DT. It is observed that the value of Δ_{pim} has a fluctuation in the experiment, e.g., Δ_{pim} doesn't change significantly when ϵ is increased from 3.0 (resp. 7.0) to 5.0 (resp. 9.0).

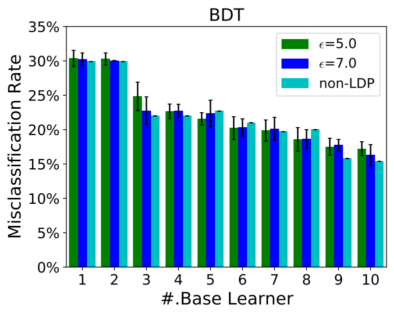
The PM-BDT is trained with 10 decision stumps and the misclassification rate is presented in Fig. 7. For MNIST, the misclassification rate of the non-private BDT is reduced from 14% to 7%, and the misclassification rate of PM-BDT is reduced from 34% (resp. 19%) to 13% (resp. 9%) when $\epsilon =$



(a) MNIST

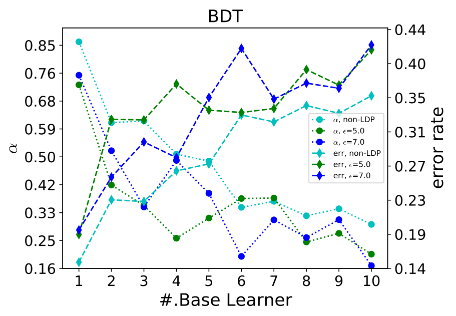


(b) Fashion-MNIST

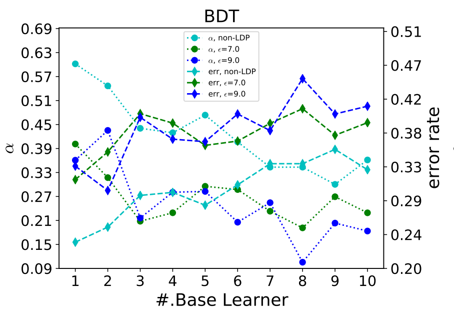


(c) Synthetic

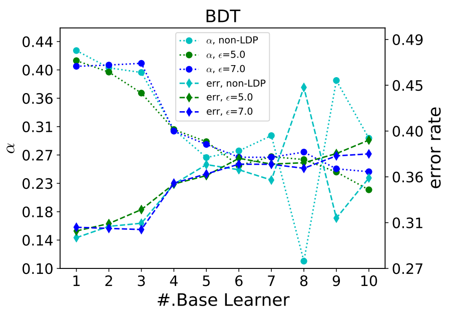
Fig. 7: PM-BDT misclassification rate w.r.t. the number of decision stumps. For all 3 datasets, the PM-BDT is composed with 10 decision stumps.



(a) MNIST



(b) Fashion-MNIST



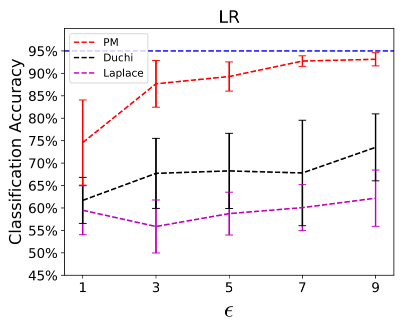
(c) Synthetic

Fig. 8: PM-BDT α_m and err_m w.r.t. the number of decision stumps. For all 3 datasets, the PM-BDT is trained with 10 decision stumps, and the variance is omitted from the figure.

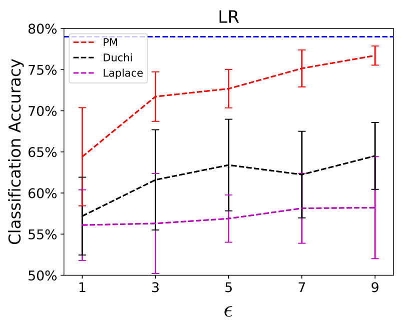
5.0 (resp. 7.0), and the performance is close to the non-private BDT when $\epsilon = 7.0$. For Fashion-MNIST, the misclassification rate of PM-BDT is reduced from 40% (resp. 34%) to 18% (resp. 19%) when $\epsilon = 7.0$ (resp. 9.0). For the synthetic dataset, the misclassification rate of PM-BDT is close to the non-private BDT when $\epsilon \geq 5.0$, which is reduced from 30% to 19% with 10 DTs. For the MNIST and Fashion-MNIST dataset, it can be seen that the performance of PM-BDT get improved as the privacy levels are relaxed, the reason is that the noise is reduced in the aggregated cross tables and it results in a higher chance of returning the true best feature. The convergence speed is also studied in terms of the base learner weight(α_m) and error rates, and the values are plotted in Fig. 8. For the MNIST, the err_m of PM-BDT grows from 0.2 to 0.35 in 7 rounds. For Fashion-MNIST, the err_m grows from 0.33 to 0.4 in 10 rounds. For the synthetic dataset, the err_m grows from 0.31 to 0.39 in 10 rounds. For all 3 dataset, the α_m and err_m of PM-BDT are actually quite close to non-private BDT, especially for the synthetic dataset, which confirms that PM-BDT maintains a good utility under different privacy level. Furthermore, the α_m of both PM-BDT and non-private BDT are not close to 0 after 10 iterations, which indicates that the accuracy might be improved by adding more decision stumps.

C. Boosted Logistic Regression Classifier

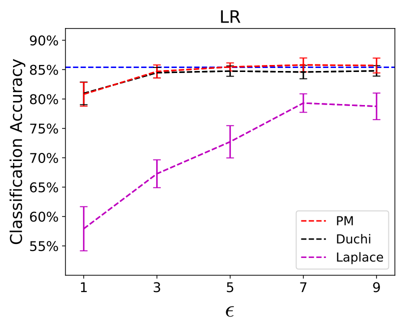
The evaluation of the boosted LR is performed over the MNIST, Fashion-MNIST and Brazil datasets. The local LR is firstly fitted by the participated data owners, and the parameters of the base LR are the estimated mean of the perturbed local LR parameters. Fig. 9 presents the classification performance of the first base LR under various privacy budgets. And to explain the classification performance, we compare the square root of the average of squared errors, i.e., $RMSE_{\boldsymbol{\theta}} = \sqrt{\frac{\sum_i (\theta_i - \theta_i')^2}{d}}$, to the dot product of $\boldsymbol{\theta}$ and the unit vector, i.e., $||\boldsymbol{\theta}||_1$. The prediction of LR is determined by $sgn(\boldsymbol{\theta}^T \boldsymbol{x})$, and the utility of the base LR is maintained if $sgn(\boldsymbol{\theta'}^T \boldsymbol{x})$ is same as $sgn(\boldsymbol{\theta}^T \boldsymbol{x})$. We hypothesize that $\boldsymbol{\theta'}^T \boldsymbol{x}$ and $\boldsymbol{\theta}^T \boldsymbol{x}$ will have the same sign if $RMSE_{\boldsymbol{\theta}} < ||\boldsymbol{\theta}||_1$. Table. IV gives the $RMSE_{\boldsymbol{\theta}}$ for various privacy levels. For MNIST, the $RMSE_{\boldsymbol{\theta}}$ of PM-LR is close to $||\boldsymbol{\theta}||_1$ when $\epsilon \geq 3.0$, and it is observed that the accuracy of PM-LR is closed to non-private LR at the same privacy level in Fig. 9. For the Brazil dataset, due to the large number of participating data owners in each round, $RMSE_{\boldsymbol{\theta}}$ of Duchi-LR and PM-LR are far less than $||\boldsymbol{\theta}||_1$ when $\epsilon \geq 3.0$, and it can be seen that their prediction capacities are almost same as the non-private LR. It's also observed that $RMSE_{\boldsymbol{\theta}}$ of Duchi-LR starts to converge when $\epsilon = 3.0$, and $RMSE_{\boldsymbol{\theta}}$ of PM-LR has a similar fluctuation when ϵ grows from 3.0 (resp. 7.0) to 5.0 (resp.



(a) MNIST



(b) Fashion-MNIST



(c) Brazil

Fig. 9: The base LR classification accuracy w.r.t. various privacy budgets, the blue horizontal dot line gives the classification accuracy of the non-private LR. For MNIST and Fashion-MNIST, there are 100 data owners participating in each round and each data owner holds 40 samples. For Brazil, there are 10,000 data owners participating in each round and each data owner holds 100 samples.

TABLE IV: $RMSE_{\boldsymbol{\theta}}$ of the aggregation of the perturbed LR parameters in terms of Laplace, Duchi and PM respectively. The $||\boldsymbol{\theta}||_1$ of MNIST is 0.43, the $||\boldsymbol{\theta}||_1$ of Fashion-MNIST is 0.71, and the $||\boldsymbol{\theta}||_1$ of Brazil is 0.22.

	Perturbation \ ϵ	1.0	3.0	5.0	7.0	9.0
MNIST	Laplace	518.3575 + 90.0363	58.0616 + 4.3229	21.6944 + 2.4638	11.7771 + 2.2048	7.0411 + 1.3191
	Duchi	11.8556 + 2.5871	2.9711 + 0.4087	2.8311 + 0.4233	2.8072 + 0.3401	2.7251 + 0.4631
	PM	**5.2721 + 0.9986**	**0.4872 + 0.1812**	**0.3388 + 0.0652**	**0.1703 + 0.0390**	**0.1483 + 0.0349**
Fashion-MNIST	Laplace	270.5490 + 52.7679	30.7099 + 5.1766	10.9630 + 1.7186	5.9292 + 1.0899	3.6319 + 0.6998
	Duchi	7.2207 + 1.2067	1.9912 + 0.2762	1.5668 + 0.2830	1.5195 + 0.2493	1.4536 + 0.4074
	PM	**3.1646 + 0.7951**	**0.3595 + 0.1080**	**0.2461 + 0.0419**	**0.1183 + 0.0215**	**0.1091 + 0.0244**
Brazil	Laplace	11.9691 + 1.3442	1.2872 + 0.1661	0.5361 + 0.0904	0.2347 + 0.0295	0.1658 + 0.0286
	Duchi	**0.1616 + 0.0227**	0.0436 + 0.0070	0.0377 + 0.0075	0.0400 + 0.0059	0.0372 + 0.0027
	PM	0.2050 + 0.0200	**0.0100 + 0.0021**	**0.0105 + 0.0024**	**0.0034 + 0.0012**	**0.0043 + 0.0006**

9.0). The experiments above demonstrate the utility of the base classifier in one round of Alg. 7, however we didn't observe accuracy improvements by boosting LR classifiers. It is also observed that the decision boundaries of LRs are barely shifted as the learned coefficients don't change across iterations.

Among 3 boosted classifiers, for MNIST dataset, comparing PM-LR to PM-NCC and PM-DT, it shows that PM-LR is able to maintain a good classification accuracy(i.e., 75%) when $\epsilon = 1.0$, while both PM-NCC and PM-DT are dominated by the noise and performs like random guessing at the same privacy level. However, as the privacy level is relaxed, the classification accuracies of the PM-BNCC and PM-BDT can be improved significantly, for instance, when $\epsilon = 7.0$, the accuracy of PM-BNNC is improved to 88% in 10 iterations, and PM-BDT is improved to 91%. For the Fashion-MNIST, it shows a similar pattern for the 3 boosted classifiers, when $\epsilon = 9.0$, the accuracy of PM-BNCC is improved to 79% in 10 iterations, the accuracy of the PM-BDT is improved to 78%, and the accuracy of PM-LR is about 78%.

VI. Related Work

Differentially private boosting algorithm received several studies recently. Liu et al. [15] propose to train the differentially-private decision tree based on the noisy maximal vote. CART-DPsAdaBoost [35] is a AdaBoost algorithm that satisfies ϵ-DP, which iteratively trains the decision stumps and adds Laplace noise in the leaf nodes. Li et al. [17] optimize the privacy budget allocation and improve the classification accuracy of the private Gradient Boosting Decision Tree (GBDT) model. However, the work introduced above assumes the data is already collected, which doesn't face the privacy challenge of sharing information among multiple data parties. To address the exact challenge, Zhao et al. [16] propose a GBDT that privately ensembles trees trained by multiple data owners and satisfies differential privacy. Although the distributed training algorithm satisfies strong privacy guarantee, it suffers from an accuracy loss due to the loose sensitivity bound and large noise incurred by the Laplace mechanism. Xiang et al. [34] propose a collaborative ensemble learning framework and design both differentially private random forest (CRFsDP) and adaptive boosting (CAdaBoostDP) algorithms. It should be noted that CAdaBoostDP and the proposed Alg. 4 in this paper have a similar implementation, however, one concern of CAdaBoostDP is that the integration of multiple local trees is not differentially private. In detail, each data owner sends the prediction accuracy of the local learner and the number of samples to the central agent to compute the weight for each local tree. Although the author argues that SMC is able to hide the plaintext of the message, it is orthogonal to the differential privacy protection.

Local Differential Privacy (LDP) is proposed by Duchi et al. [14] to prevent the information leakage in user level. Several data mining and machine learning algorithms that satisfy LDP are proposed, e.g., probability distribution estimation [38], [39], frequent itemset mining [40], Bayes learning [41] and clustering [42]. Wang et al. [25] propose a piecewise mechanism for multi-dimension numerical attributes perturbation and use it to update the gradient across data parities. Another client-server machine learning framework is designed for Generalized Linear models [43], where the client perturbs the parameters of the updated local model to satisfy ϵ-LDP and shared the perturbed model with the server. Similarly, Zhang et al. [44] propose a solution for multi-party deep learning, where the local gradients are enforced by ϵ-LDP. In this paper, we are interested in proposing a general privacy-preserving framework for boosting and supports the type of classifiers which is hardly trained by the SGD approach.

Federated learning [45] trains a machine learning model by utilizing the dataset held in multiple de-centralized edge devices, and it is assumed that the local datasets are not necessarily identically distributed. Compared to our problem setting, federated learning has a sound property of hiding the local samples, e.g., the participants are able to share the model parameters rather than the true data samples, which alleviates the risk of privacy leakage. However, as some recent study shows, federated learning is still facing critical privacy issues without proper protection, for instance, Melis et al. [46] introduce the inference attacks in federated learning, where an adversarial participant is able to infer the existence of some exact data points in other participants' training sets. Such attack confirms that the privacy protection is orthogonal to the federated learning. Most recently, Wei et al. [47] propose a noising before model aggregation (NbAFL) federated learning framework, which satisfies ϵ-DP by adopting the Gaussian-based mechanism. Since federated learning averages weights from local models for iterative training, a naive solution is to directly perturb weights before averaging. However, the DNN usually has a large volume of trainable parameters, e.g., there are nearly 200K parameters in a 2NN multi-layer perceptron [45]. In our experiment, we tried to perturb the local weights of the 2NN using Alg. 3, while the error is extreme large even by assigning 10K privacy budget for a single participant. On the other hand, the federated learning is also robust on unbalanced and non-IID input, and our proposed boosting algorithm has the same property assuming the base learner is trained by aggregating local samples.

VII. Conclusion

In this paper, we proposed a privacy-preserving boosting algorithm in the distributed setting, where an untrustworthy data user intends to learn a boosted classifier from multiple data parties. The proposed algorithm satisfied Local Differential Privacy (LDP) by only utilizing the perturbed local shares from data owners, while the true values of training samples never leave data owners' hands. The state-of-art private boosting algorithms mostly study certain classifiers, e.g., Boosted Decision Tree (BDT) and Gradient Boosting Decision Tree (GBDT). In contrast, our proposed boosting algorithm is generalized and able to support different types of classifiers. For instance, we implemented a private boosted Nearest Centroid Classifier by utilization the aggregation of the perturbed *local samples*, and a private BDT classifier by using the aggregation of the perturbed *local statistics*. We also demonstrated the perturbation of the Logistic Regression classifier. In the experiment, we implemented all 3 boosted classifiers and analyzed their classification performances over several real and synthetic datasets. The result showed that the privacy-preserving boosted classifiers effectively maintained a superior utility. As the LDP allows the perturbation happens on the data sample level, it provides a great flexibility for different machine learning algorithms. For the future work, one direction is to develop other boosting algorithms that satisfy LDP, like stochastic gradient boosting; another direction is to extend the work to the non-IID data setting.

References

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1990.

[2] L. Breiman, "Bagging predictors," *Machine learning*, 1996.

[3] R. E. Schapire, "The strength of weak learnability," *Machine learning*, 1990.

[4] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, 1997.

[5] D. H. Wolpert, "Stacked generalization," *Neural networks*, 1992.

[6] C. Apte, F. Damerau, S. Weiss *et al.*, *Text mining with decision rules and decision trees.* Citeseer, 1998.

[7] M. Pal and P. M. Mather, "An assessment of the effectiveness of decision tree methods for land cover classification," *Remote sensing of environment*, 2003.

[8] Y. Xia, C. Liu, Y. Li, and N. Liu, "A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring," *Expert Systems with Applications*, 2017.

[9] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security & Privacy*, 2019.

[10] X. Shen, B. Tan, and C. Zhai, "Privacy protection in personalized search," in *ACM Special Interest Group in Information Retrieval Forum*, 2007.

[11] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *IEEE Symposium on Security and Privacy*, 2008.

[12] C. Dwork, "Differential privacy," in *33 International Conference on Automata, Languages and Programming*, 2006.

[13] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *arXiv preprint arXiv:1412.7584*, 2014.

[14] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, 2013.

[15] X. Liu, Q. Li, T. Li, and D. Chen, "Differentially private classification with decision tree ensemble," *Applied Soft Computing*, 2018.

[16] L. Zhao, L. Ni, S. Hu, Y. Chen, P. Zhou, F. Xiao, and L. Wu, "Inprivate digging: Enabling tree-based distributed data mining with differential privacy," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 2018.

[17] Q. Li, Z. Wu, Z. Wen, and B. He, "Privacy-preserving gradient boosting decision trees," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[18] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014.

[19] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, 2009.

[20] P.-N. Tan, *Introduction to data mining*. Pearson Education India, 2018.

[21] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta, "Distributed decision-tree induction in peer-to-peer systems," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2008.

[22] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, 1965.

[23] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proceedings of the 26th USENIX Security Symposium*, 2017.

[24] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Minimax optimal procedures for locally private estimation," *Journal of the American Statistical Association*, 2018.

[25] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, “Collecting and analyzing multidimensional data with local differential privacy,” in *IEEE 35th International Conference on Data Engineering (ICDE)*, 2019.
[26] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *FOCS*, 2007.
[27] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, “Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices,” in *IEEE 35th International Conference on Distributed Computing Systems*, 2015.
[28] T. Bylander and L. Tate, “Using validation sets to avoid overfitting in adaboost,” 2006.
[29] I. Landesa-VáZquez and J. L. Alba-Castro, “Shedding light on the asymmetric learning capability of adaboost,” *Pattern Recognition Letters*, 2012.
[30] D. W. Meijer and D. M. Tax, “Regularizing adaboost with validation sets of increasing size,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016.
[31] D. Caragea, A. Silvescu, and V. Honavar, “A framework for learning from distributed data using sufficient statistics and its application to learning decision trees,” *International Journal of Hybrid Intelligent Systems*, 2004.
[32] M. Pathak, S. Rane, and B. Raj, “Multiparty differential privacy via aggregation of locally trained classifiers,” in *Advances in Neural Information Processing Systems*, 2010.
[33] J. Hamm, Y. Cao, and M. Belkin, “Learning privately from multiparty data,” in *International Conference on Machine Learning*, 2016.
[34] T. Xiang, Y. Li, X. Li, S. Zhong, and S. Yu, “Collaborative ensemble learning under differential privacy,” in *Web Intelligence*. IOS Press, 2018.
[35] J. Jia and W. Qiu, “Research on an ensemble classification algorithm based on differential privacy,” *IEEE Access*, 2020.
[36] “Integrated public use microdata series,” https://www.ipums.org, 2019.
[37] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of cryptography conference*. Springer, 2006.
[38] P. Kairouz, K. Bonawitz, and D. Ramage, “Discrete distribution estimation under local privacy,” *Proceedings of the 33rd International Conference on Machine Learning*, 2016.
[39] M. Ye and A. Barg, “Optimal schemes for discrete distribution estimation under locally differential privacy,” *IEEE Transactions on Information Theory*, 2018.
[40] T. Wang, N. Li, and S. Jha, “Locally differentially private frequent itemset mining,” in *IEEE Symposium on Security and Privacy (SP)*, 2018.
[41] E. Yilmaz, M. Al-Rubaie, and J. M. Chang, “Naive bayes classification under local differential privacy,” in *IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, 2020.
[42] K. Nissim and U. Stemmer, “Clustering algorithms for the centralized and local models,” *arXiv preprint arXiv:1707.04766*, 2017.
[43] V. Pihur, A. Korolova, F. Liu, S. Sankuratripati, M. Yung, D. Huang, and R. Zeng, “Differentially-private ’draw and discard’ machine learning,” *arXiv preprint arXiv:1807.04369*, 2018.
[44] X. Zhang, S. Ji, H. Wang, and T. Wang, “Private, yet practical, multiparty deep learning,” in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.
[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*. PMLR, 2017.
[46] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019.
[47] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, 2020.

Sen Wang received the B.S. degree in software engineering from Xiamen University, China, in 2010, and the M.S. degree in computer science from Iowa State University, in 2013. He is currently pursuing the Ph.D. degree in the Department of Electrical Engineering, University of South Florida, Tampa, Florida. His research interests include the database, security, privacy-preserving data mining and machine learning.

Morris Chang received his BSEE degree from Tatung Institute of Technology, Taiwan and his MS and PhD in Computer Engineering is from North Carolina State University. He is currently a Professor at the Department of Electrical Engineering at University of South Florida. Dr.Changs industrial experience includes positions at Texas Instruments, Taiwan, Microelectronics Center of North Carolina, and AT&T Bell Laboratories, Pennsylvania. He was on the faculty of the Department of Electrical Engineering at Rochester Institute of Technology, Rochester, the Department of Computer Science at Illinois Institute of Technology, Chicago and the Department of Electrical and Computer Engineering at Iowa State University, IA. His research interests include cyber security, wireless networks, energy-aware computing and object-oriented systems. Currently, Dr. Chang is a handling editor of Journal of Microprocessors and Microsystems and the Associate Editor-in-Chief of IEEE IT Professional. He is a senior member of IEEE.