# Privacy-Preserving Gradient Boosting Decision Trees

**Qinbin Li,**[1] **Zhaomin Wu,**[1] **Zeyi Wen,**[2] **Bingsheng He**[1]
[1]National University of Singapore
[2]The University of Western Australia
{qinbin, zhaomin, hebs}@comp.nus.edu.sg, zeyi.wen@uwa.edu.au



**Abstract**

The Gradient Boosting Decision Tree (GBDT) is a popular machine learning model for various tasks in recent years. In this paper, we study how to improve model accuracy of GBDT while preserving the strong guarantee of differential privacy. *Sensitivity* and *privacy budget* are two key design aspects for the effectiveness of differential private models. Existing solutions for GBDT with differential privacy suffer from the significant accuracy loss due to too loose sensitivity bounds and ineffective privacy budget allocations (especially across different trees in the GBDT model). Loose sensitivity bounds lead to more noise to obtain a fixed privacy level. Ineffective privacy budget allocations worsen the accuracy loss especially when the number of trees is large. Therefore, we propose a new GBDT training algorithm that achieves tighter sensitivity bounds and more effective noise allocations. Specifically, by investigating the property of gradient and the contribution of each tree in GBDTs, we propose to adaptively control the gradients of training data for each iteration and leaf node clipping in order to tighten the sensitivity bounds. Furthermore, we design a novel boosting framework to allocate the privacy budget between trees so that the accuracy loss can be further reduced. Our experiments show that our approach can achieve much better model accuracy than other baselines.

## 1 Introduction

Gradient Boosting Decision Trees (GBDTs) have achieved state-of-the-art results on many challenging machine learning tasks such as click prediction (Richardson, Dominowska, and Ragno 2007), learning to rank (Burges 2010), and web page classification (Pennacchiotti and Popescu 2011). The algorithm builds a number of decision trees one by one, where each tree tries to fit the residual of the previous trees. With the development of efficient GBDT libraries (Chen and Guestrin 2016; Ke et al. 2017; Prokhorenkova et al. 2018; Wen et al. 2019), the GBDT model has won many awards in recent machine learning competitions and has been widely used both in the academics and in the industry (He et al. 2014; Zhou et al. 2017; Ke et al. 2017; Jiang et al. 2018; Feng, Yu, and Zhou 2018).



Privacy issues have been a hot research topic recently (Shokri et al. 2017; Truex et al. 2018; Fredrikson, Jha, and Ristenpart 2015; Li et al. 2019; Li, Wen, and He 2019). Due to the popularity and wide adoptions of GBDTs, a privacy-preserving GBDT algorithm is particularly timely and necessary. Differential privacy (Dwork 2011) was proposed to protect the individuals of a dataset. In short, a computation is differentially private if the probability of producing a given output does not depend much on whether a particular record is included in the input dataset. Differential privacy has been widely used in many machine learning models such as logistic regression (Chaudhuri and Monteleoni 2009) and neural networks (Abadi et al. 2016; Acs et al. 2018). *Sensitivity* and *privacy budget* are two key design aspects for the effectiveness of differential private models. Many practical differentially private models achieve good model quality by deriving tight sensitivity bounds and allocating privacy budget effectively. In this paper, we study how to improve model accuracy of GBDTs while preserving the strong guarantee of differential privacy.

There have been some potential solutions for improving the effectiveness of differentially private GBDTs (e.g., (Zhao et al. 2018; Liu et al. 2018; Xiang et al. 2018)). However, they can suffer from the significant accuracy loss due to too loose sensitivity bounds and ineffective privacy budget allocations (especially across different trees in the GBDT model).

*Sensitivity bounds:* The previous studies on individual decision trees (Friedman and Schuster 2010; Mohammed et al. 2011; Liu et al. 2018) bound the sensitivities by estimating the range of the function output. However, this method leads to very loose sensitivity bounds in GBDTs, because the range of the gain function output ($G$ in Equation (3) introduced Section 2) is related to the number of instances and the range can be potentially very huge for large data sets. Loose sensitivity bounds lead to more noise to obtain a fixed privacy level, and cause huge accuracy loss.

*Privacy budget allocations:* There have been some previous studies on privacy budget allocations among different trees (Liu et al. 2018; Xiang et al. 2018; Zhao et al. 2018). We can basically divide them into two kinds. 1) The first kind is to allocate the budget equally to each tree using the

*sequential composition* (Liu et al. 2018; Xiang et al. 2018). When the number of trees is large, the given budget allocated to each tree is very small. The scale of the noises can be proportional to the number of trees, which causes huge accuracy loss. 2) The second kind is to give disjoint inputs to different trees (Zhao et al. 2018). Then, each tree only needs to satisfy $\varepsilon$-differential privacy using the *parallel composition*. When the number of trees is large, since the inputs to the trees cannot be overlapped, the number of instances assigned to a tree can be quite small. As a result, the tree can be too weak to achieve meaningful learnt models.

We design a new GBDT training algorithm to address the above-mentioned limitations.

- In order to obtain a tighter sensitivity bound, we propose Gradient-based Data Filtering (GDF) to guarantee the bounds of the sensitivities, and further propose Geometric Leaf Clipping (GLC) to obtain a closer bound on sensitivities taking advantage of the tree learning systems in GBDT.
- Combining both sequential and parallel compositions, we design a novel boosting framework to well exploit the privacy budget of GBDTs and the effect of boosting. Our approach satisfies differential privacy while improving the model accuracy with boosting.
- We have implemented our approach (named DPBoost) based on a popular library called LightGBM (Ke et al. 2017). Our experimental results show that DPBoost is much superior to the other approaches and can achieve competitive performance compared with the ordinary LightGBM.

# 2 Preliminaries

## 2.1 Gradient Boosting Decision Trees

The GBDT is an ensemble model which trains a number of decision trees in a sequential manner. Formally, given a convex loss function $l$ and a dataset with $n$ instances and $d$ features $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n} (\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R})$, GBDT minimizes the following regularized objective (Chen and Guestrin 2016).

$$\tilde{\mathcal{L}} = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{1}$$

where $\Omega(f) = \frac{1}{2}\lambda\|V\|^2$ is a regularization term. Here $\lambda$ is the regularization parameter and $V$ is the leaf weight. Each $f_k$ corresponds to a decision tree. Forming an approximate function of the loss, GBDT minimizes the following objective function at the $t$-th iteration (Si et al. 2017).

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} f_t^2(\mathbf{x}_i)] + \Omega(f_t) \tag{2}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ is first order gradient statistics on the loss function. The decision tree is built from the root until reaching the maximum depth. Assume $I_L$ and $I_R$ are the instance sets of left and right nodes after a split. Letting $I = I_L \cup I_R$, the gain of the split is given by

$$G(I_L, I_R) = \frac{(\sum_{i\in I_L} g_i)^2}{|I_L| + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{|I_R| + \lambda} \tag{3}$$

GBDT traverses all the feature values to find the split that maximizes the gain. If the current node does not meet the requirements of splitting (e.g., achieve the max depth or the gain is smaller than zero), it becomes a leaf node and the optimal leaf value is given by

$$V(I) = -\frac{\sum_{i\in I} g_i}{|I| + \lambda} \tag{4}$$

Like the learning rate in stochastic optimization, a shrinkage rate $\eta$ (Friedman 2002) is usually applied to the leaf values, which can reduce the influence of each individual tree and leave space for future trees to improve the model.

## 2.2 Differential Privacy

Differential privacy (Dwork 2011) is a popular standard of privacy protection with provable privacy guarantee. It guarantees that the probability of producing a given output does not depend much on whether a particular record is included in the input dataset or not.

**Definition 1.** ($\varepsilon$-Differential Privacy) Let $\varepsilon$ be a positive real number and $f$ be a randomized function. The function $f$ is said to provide $\varepsilon$-differential privacy if, for any two datasets $D$ and $D'$ that differ in a single record and any output $O$ of function $f$,

$$Pr[f(D) \in O] \leq e^{\varepsilon} \cdot Pr[f(D') \in O] \tag{5}$$

Here $\varepsilon$ is a privacy budget. To achieve $\varepsilon$-differential privacy, the Laplace mechanism and exponential mechanism (Dwork, Roth, and others 2014) are usually adopted by adding noise calibrated to the *sensitivity* of a function.

**Definition 2.** (Sensitivity) Let $f : \mathcal{D} \to \mathcal{R}^d$ be a function. The sensitivity of $f$ is

$$\Delta f = \max_{D,D'\in\mathcal{D}} \|f(D) - f(D')\|_1 \tag{6}$$

where $D$ and $D'$ have at most one different record.

**Theorem 1.** (Laplace Mechanism) Let $f : \mathcal{D} \to \mathcal{R}^d$ be a function. The Laplace Mechanism $F$ is defined as

$$F(D) = f(D) + Lap(0, \Delta f/\varepsilon) \tag{7}$$

where the noise $Lap(0, \Delta f/\varepsilon)$ is drawn from a Laplace distribution with mean zero and scale $\Delta f/\epsilon$. Then $F$ provides $\epsilon$-differential privacy.

**Theorem 2.** (Exponential Mechanism) Let $u : (\mathcal{D} \times \mathcal{R}) \to \mathbb{R}$ be a utility function. The exponential mechanism $F$ is defined as

$$\begin{aligned} F(D, u) = &\text{ choose } r \in \mathcal{R} \text{ with} \\ &\text{probability } \propto exp(\frac{\varepsilon u(D, r)}{2\Delta u}) \end{aligned} \tag{8}$$

Then $F$ provides $\varepsilon$-differential privacy.

The above-mentioned mechanisms provide privacy guarantees for a single function. For an algorithm with multiple functions, there are two privacy budget composition theorems (Dwork, Roth, and others 2014).

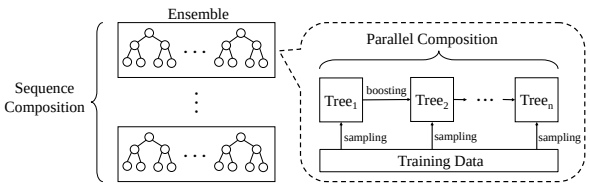



Figure 1: The two-level boosting design of DPBoost

**Theorem 3.** (Sequential Composition) Let $f = \{f_1, ..., f_m\}$ be a series of functions performed sequentially on a dataset. If $f_i$ provides $\varepsilon_i$-differential privacy, then $f$ provides $\sum_{i=1}^{m} \varepsilon_i$-differential privacy.

**Theorem 4.** (Parallel Composition) Let $f = \{f_1, ..., f_m\}$ be a series of functions performed separately on disjoint subsets of the entire dataset. If $f_i$ provides $\varepsilon_i$-differential privacy, then $f$ provides $\max(\varepsilon_1, ..., \varepsilon_m)$-differential privacy.

## 3 Our Design: DPBoost

Given a privacy budget $\varepsilon$ and a dataset with $n$ instances and $d$ features $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n} (\mathbf{x}_i \in \mathbb{R}^d, y_i \in [-1, 1])$, we develop a new GBDT training algorithm named DPBoost to achieve $\varepsilon$-differential privacy while trying to reduce the accuracy loss. Moreover, like the setting in the previous work (Abadi et al. 2016), we consider a strong adversary with full access to the model's parameters. Thus, we also provide differential privacy guarantees for each tree node.

Figure 1 shows the overall framework of DPBoost. We design a novel two-level boosting framework to exploit both sequential composition and parallel composition. Inside an ensemble, a number of trees are trained using the disjoint subsets of data sampled from the dataset. Then, multiple such rounds are trained in a sequential manner. For achieving differential privacy, sequential composition and parallel composition are applied between ensembles and inside an ensemble, respectively. Next, we describe our algorithm in detail, including the techniques to bound sensitivities and effective privacy budget allocations.

### 3.1 Tighter Sensitivity Bounds

The previous studies on individual decision trees (Friedman and Schuster 2010; Mohammed et al. 2011; Liu et al. 2018) bound the sensitivities by estimating the range of the function output. For example, if the range of a function output is $[-\gamma, \gamma]$, then the sensitivity of the function is no more than $2\gamma$. However, the range of function $G$ in Equation (3) is related to the number of instances, which can cause very large sensitivity if the dataset is large. Thus, instead of estimating the range, we strictly derive the sensitivity ($\Delta G$ and $\Delta V$) according to Definition 2. Their bounds are given in the below two lemmas.

**Lemma 1.** Letting $g^* = max_{i \in \mathcal{D}} |g_i|$, we have $\Delta G \leq \frac{3\lambda+2}{(\lambda+1)(\lambda+2)} g^{*2}$.

*Proof.* Consider two adjacent instance sets $I_1 = \{\mathbf{x}_i\}_{i=1}^{n}$ and $I_2 = I_1 \cup \{\mathbf{x}_s\}$ that differ in a single instance. Assume $I_1 = I_L \cup I_R$, where $I_L$ and $I_R$ are the instance sets of leaf and right nodes respectively after a split. Without loss of generality, we assume that instance $x_s$ belongs to the left node. We use $n_l$ to denote $|I_L|$. Then, we have

$$\begin{aligned} \Delta G &= |\frac{(\sum_{i \in I_L} g_i + g_s)^2}{n_l + \lambda + 1} - \frac{(\sum_{i \in I_L} g_i)^2}{n_l + \lambda}| \\ &= |\frac{(n_l + \lambda) g_s^2 + 2(n_l + \lambda) g_s \sum_{i \in I_L} g_i - (\sum_{i \in I_L} g_i)^2}{(n_l + \lambda + 1)(n_l + \lambda)}| \end{aligned} \tag{9}$$

Let $h(g_s, \sum_{i \in I_L} g_i) = (n_l+\lambda) g_s^2 + 2(n_l+\lambda) g_s \sum_{i \in I_L} g_i - (\sum_{i \in I_L} g_i)^2$. When $g_s = g^*$ and $\sum_{i \in I_L} g_i = n_l g^*$, $|h(g_s, \sum_{i \in I_L} g_i)|$ can achieve maximum. We have

$$\begin{aligned} \Delta G &= |\frac{(n_l + 1)^2 g^{*2}}{n_l + \lambda + 1} - \frac{n_l^2 g^{*2}}{n_l + \lambda}| \\ &= |1 - \frac{\lambda^2}{(n_l + \lambda + 1)(n_l + \lambda)}| g^{*2} \\ &\leq |1 - \frac{\lambda^2}{(\lambda + 1)(\lambda + 2)}| g^{*2} \\ &= \frac{3\lambda + 2}{(\lambda + 1)(\lambda + 2)} g^{*2} \end{aligned} \tag{10}$$

□

**Lemma 2.** Letting $g^* = max_{i \in \mathcal{D}} |g_i|$, we have $\Delta V \leq \frac{g^*}{1+\lambda}$.

*Proof.* The proof follows a similar way with the proof of Lemma 1. The detailed proof is available in Appendix A. □

For ease of presentation, we call the absolute value of the gradient as 1-norm gradient. As we can see from Lemma 1 and Lemma 2, the sensitivities of nodes are related to the maximum 1-norm gradient. Since there is no a priori bound on the value of the gradients, we have to restrict the range of gradients. A potential solution is to clip the gradient by a threshold, which is often adopted in deep learning (Abadi et al. 2016; Shokri and Shmatikov 2015). However, in GBDTs, since the gradients are computed based on distance between the prediction value and the target value, clipping the gradients means indirectly changing the target value, which may lead to a huge accuracy loss. Here, we propose a new approach named gradient-based data filtering. The basic idea is to restrict the maximum 1-norm gradient by only filtering a very small fraction of the training dataset in each iteration.

**Gradient-based Data Filtering (GDF)** At the beginning of the training, the gradient of instance $\mathbf{x}_i$ is initialized as $g_i = \frac{\partial l(y_i, y)}{\partial y}|_{y=0}$. We let $g_l^* = max_{y_p \in [-1,1]} \| \frac{\partial l(y_p, y)}{\partial y}|_{y=0} \|$, which is the maximum possible 1-norm gradient in the initialization. Note that $g_l^*$ is **independent to training data** and only depends on the loss function $l$ (e.g., $g_l^* = 1$ for square loss function). Since the loss function $l$ is convex (i.e., the gradient is monotonically non-decreasing), the values of the 1-norm gradients tend to

decrease as the number of trees increases in the training. Consequently, as we have shown the experimental results in Appendix B, most instances have a lower 1-norm gradient than $g_l^*$ during the whole training process. Thus, we can filter the training instances by the threshold $g_l^*$. Specifically, at the beginning of each iteration, we filter the instances that have 1-norm gradient larger than $g_l^*$ (i.e., those instances are not considered in this iteration). Only the remaining instances are used as the input to build a new differentially private decision tree in this iteration. Note that the filtered instances may still participate in the training of the later trees. With such gradient-based data filtering technique, we can ensure that the gradients of the used instances are no larger than $g_l^*$. Then, according to Lemma 1 and Lemma 2, we can bound the sensitivities of $G$ and $V$ as shown in Corollary 1.

**Corollary 1.** By applying GDF in the training of GBDTs, we have $\Delta G \leq \frac{3\lambda+2}{(\lambda+1)(\lambda+2)} {g_l^*}^2$ and $\Delta V \leq \frac{g_l^*}{1+\lambda}$.

In the following, we analyze the approximation error of GDF.

**Theorem 5.** Given an instance set $I$, suppose $I = I_f \cup I_f^c$, where $I_f$ is the filtered instance set and $I_f^c$ is the remaining instance set in GDF. Let $p = \frac{|I_f|}{|I|}$ and $\bar{g}_f = \frac{\sum_{i \in I_f} g_i}{|I_f|}$. We denote the approximation error of GDF on leaf values as $\xi_I = |V(I) - V(I_f^c)|$. Then, we have $\xi_I \leq p(|\bar{g}_f| + g_l^*)$.

*Proof.* With Equation (4), we have

$$\begin{aligned}
\xi_I &= |\frac{\sum_{i \in I_f} g_i + \sum_{i \in I_f^c} g_i}{|I| + \lambda} - \frac{\sum_{i \in I_f^c} g_i}{(1-p)|I| + \lambda}| \\
&\leq |\frac{\sum_{i \in I_f} g_i}{|I| + \lambda}| + |(\frac{1}{|I| + \lambda} - \frac{1}{(1-p)|I| + \lambda}) \sum_{i \in I_f^c} g_i| \\
&= p|\frac{\sum_{i \in I_f} g_i}{|I_f| + p\lambda}| + p|\frac{|I|}{(|I| + \lambda)((1-p)|I| + \lambda)} \sum_{i \in I_f^c} g_i| \\
&\leq p|\frac{\sum_{i \in I_f} g_i}{|I_f|}| + p|\frac{\sum_{i \in I_f^c} g_i}{(1-p)|I|}| \leq p(|\bar{g}_f| + g_l^*)
\end{aligned} \tag{11}$$

□

According to Theorem 5, we have the following discussions: (1) The upper bound of the approximation error of GDF does not depend on the number of instances. This good property allows small approximation errors even on large data sets. (2) Normally, most instances have gradient values lower than the threshold $g_l^*$ and the ratio $p$ is low, as also shown in Appendix B. Then, the approximation error is small in practice. (3) The approximation error may be large if $|\bar{g}_f|$ is big. However, the instances with a very large gradient are often outliers in the training data set since they cannot be well learned by GBDTs. Thus, it is reasonable to learn a tree by filtering those outliers.

**Geometric Leaf Clipping (GLC)** GDF provides the same sensitivities for all trees. Since the gradients tend to decrease from iteration to iteration in the training process, there is an opportunity to derive a tighter sensitivity bound as the iterations go. However, it is too complicated to derive the exact decreasing pattern of the gradients in practice. Also, as discussed in the previous section, gradient clipping with an inappropriate decaying threshold can lead to huge accuracy loss. We need a new approach for controlling this decaying effect across different tree learning. Note, while the noises injected in the internal nodes influence the gain of the current split, the noises injected on the leaf value directly influence the prediction value. Here we focus on bounding the sensitivity of leaf nodes.

Fortunately, according to Equation (4), the leaf values also decrease as the gradients decrease. Since the GBDT model trains a tree at a time to fit the residual of the trees that precede it, clipping the leaf nodes would mostly influence the convergence rate but not the objective of GBDTs. Thus, we propose adaptive leaf clipping to achieve a decaying sensitivity on the leaf nodes. Since it is unpractical to derive the exact decreasing pattern of the leaf values in GBDTs, we start with a simple case and further analyze its findings in practice.

**Theorem 6.** Consider a simple case that each leaf has only one single instance during the GBDT training. Suppose the shrinkage rate is $\eta$. We use $V_t$ to denote the leaf value of the $t$-th tree in GBDT. Then, we have $|V_t| \leq g_l^*(1-\eta)^{t-1}$.

*Proof.* For simplicity, we assume the label of the instance is -1 and the gradient of the instance is initialized as $g_l^*$. For the first tree, we have $V_1 = -\frac{g_l^*}{1+\lambda} \geq -g_l^*$. Since the shrinkage rate is $\eta$, the improvement of the prediction value on the first tree is $-\eta g_l^*$. Thus, we have

$$\begin{aligned}
|V_2| &\leq \|\frac{\partial l(-1, y)}{\partial y}|_{y=\eta V_1}\| \\
&\approx \|\frac{\partial l(-1, y)}{\partial y}|_{y=0} + \eta V_1\| \\
&\leq g_l^*(1-\eta)
\end{aligned} \tag{12}$$

In the same way, we can get $|V_t| \leq g_l^*(1-\eta)^{t-1}$. □

Although the simple case in Theorem 6 may not fully reflect decaying patterns of the leaf value in practice, it can give an insight on the reduction of the leaf values as the number of trees increases. The leaf values in each tree form a geometric sequence with base $g_l^*$ and common ratio $(1-\eta)$. Based on this observation, we propose geometric leaf clipping. Specifically, in the training of the tree in iteration $t$ in GBDTs, we clip the leaf values with the threshold $g_l^*(1-\eta)^{t-1}$ before applying Laplace mechanism (i.e., $V_t' = V_t \cdot \min(1, g_l^*(1-\eta)^{t-1}/|V_t|)$). That means, if the leaf value is larger than the threshold, its value is set to be the threshold. Then, combining with Corollary 1, we get the following result on bounding the sensitivity on each tree in the training process.

**Corollary 2.** With GDF and GLC, the sensitivity of leaf nodes in the tree of the $t$-th iteration satisfies $\Delta V_t \leq \min(\frac{g_l^*}{1+\lambda}, 2g_l^*(1-\eta)^{t-1})$.

We have conducted experiments on the effect of geometric clipping, which are shown in Appendix B. Our experiments show that GLC can effectively improve the performance of DPBoost.

### 3.2 Privacy Budget Allocations

As in Introduction, previous approaches (Liu et al. 2018; Xiang et al. 2018; Zhao et al. 2018) suffer from accuracy loss, due to the ineffective privacy budget allocations across trees. The accuracy loss can be even bigger, when the number of trees in GBDT is large. For completeness, we first briefly present the mechanism for building a single tree $t$ with a given privacy budget $\varepsilon_t$, by using an approach in the previous study (Mohammed et al. 2011; Zhao et al. 2018). Next, we present our proposed approach for budget allocation across trees in details.

**Budget Allocation for A Single Tree** Algorithm 1 shows the procedure of learning a differentially private decision tree. In the beginning, we use GDF (introduced in Section 3.1) to preprocess the training dataset. Then, the decision tree is built from root until reaching the maximum depth. For the internal nodes, we adopt the exponential mechanism when selecting the split value. Considering the gain $G$ as the utility function, the feature value with higher gain has a higher probability to be chosen as the split value. For the leaf nodes, we first clip the leaf values using GLC (introduced in Section 3.1). Then, the Laplace mechanism is applied to inject random noises to the leaf values. For the privacy budget allocation inside a tree, we adopt the mechanism in the existing studies (Mohammed et al. 2011; Zhao et al. 2018). Specifically, we allocate a half of the privacy budget for the leaf nodes (i.e., $\varepsilon_{leaf}$), and then equally divide the remaining budget to each depth of the internal nodes (each level gets $\varepsilon_{nleaf}$).

**Theorem 7.** The output of Algorithm 1 satisfies $\varepsilon_t$-differential privacy.

*Proof.* Since the nodes in one depth have disjoint inputs, according to the parallel composition, the privacy budget consumption in one depth only need to be counted once. Thus, the total privacy budget consumption is no more than $\varepsilon_{leaf} * Depth_{max} + \varepsilon_{nleaf} = \varepsilon_t$. □

**Budget Allocation Across trees** We propose a two-level boosting structure called Ensemble of Ensembles (EoE), which can exploit both sequential composition and parallel composition to allocate the privacy budget between trees. Within each ensemble, we first train a number of trees with disjoint subsets sampled from the dataset $\mathcal{D}$. Thus, the parallel composition is applied inside an ensemble. Then, multiple such ensembles are trained in a sequential manner using the same training set $\mathcal{D}$. As a result, the sequential composition is applied between ensembles. Such a design can utilize the privacy budget while maintaining the effectiveness of boosting. EoE can effectively address the side effect of geometric leaf clipping in some cases, which cause the leaf values to have a too tight restriction as the iteration grows.

**Algorithm 1:** TrainSingleTree: Train a differentially private decision tree

**Input:** $I$: training data, $Depth_{max}$: maximum depth
**Input:** $\varepsilon_t$: privacy budget
$\varepsilon_{leaf} \leftarrow \frac{\varepsilon_t}{2}$ // privacy budget for leaf nodes
$\varepsilon_{nleaf} \leftarrow \frac{\varepsilon_t}{2Depth_{max}}$ // privacy budget for internal nodes
Perform gradient-based data filtering on dataset $I$.
**for** $depth = 1$ **to** $Depth_{max}$ **do**
  **for** *each node in current depth* **do**
    **for** *each split value* $i$ **do**
      Compute gain $G_i$ according to Equation (3).
      $P_i \leftarrow exp(\frac{\varepsilon_{nleaf} G_i}{2\Delta G})$
    /* Apply exponential mechanism */
    Choose a value $s$ with probability $(P_s / \sum_i P_i)$.
    Split current node by feature value $s$.
**for** *each leaf node* $i$ **do**
  Compute leaf value $V_i$ according to Equation (4).
  Perform geometric leaf clipping on $V_i$.
  /* Apply Laplace mechanism */
  $V_i \leftarrow V_i + Lap(0, \Delta V / \varepsilon_{nleaf})$
**Output:** A $\varepsilon_t$-differentially private decision tree

Algorithm 2 shows our boosting framework. Given the total number of trees $T$ and the number of trees inside an ensemble $T_e$, we can get the total number of ensembles $N_e = \lceil T/T_e \rceil$. Then, the privacy budget for each tree is $(\varepsilon/N_e)$. When building the $t$-th differentially private decision tree, we first calculate the position of the tree in the ensemble as $t_e = t \mod T_e$. Since the maximum leaf value of each tree is different in GLC, to utilize the contribution of the front trees, the number of instances allocated to a tree is proportional to its leaf sensitivity. Specifically, we randomly choose $(\frac{|\mathcal{D}|\eta(1-\eta)^{t_e}}{1-(1-\eta)^{T_e}})$ unused instances from the dataset $I$ as the input, where $I$ is initialized to the entire training dataset $\mathcal{D}$ at the beginning of an ensemble. Each tree is built using TrainSingleTree in Algorithm 1. All the $T$ trees are trained one by one, and these trees constitute our final learned model.

**Theorem 8.** The output of Algorithm 2 satisfies $\varepsilon$-differential privacy.

*Proof.* Since the trees in an ensemble have disjoint inputs, the privacy budget consumption of an ensemble is still $(\varepsilon/N_e)$ due to the parallel composition. Since there are $N_e$ ensembles in total, according to sequential composition, the total privacy budget consumption is $\varepsilon/N_e * N_e = \varepsilon$. □

## 4 Experiments

In this section, we evaluate the effectiveness and efficiency of DPBoost. We compare DPBoost with three other approaches: 1) **NP** (the vanilla GBDT): Train GBDTs without privacy concerns. 2) **PARA**: A recent approach (Zhao et al. 2018) that adopts parallel composition to train multiple trees, and uses only a half of unused instances when

**Algorithm 2:** Train differentially private GBDTs

**Input:** $\mathcal{D}$: Dataset, $Depth_{max}$: maximum depth
**Input:** $\varepsilon$: privacy budget, $\lambda$: regularization parameter
**Input:** $T$: total number of trees, $l$: loss function
**Input:** $T_e$: number of trees in an ensemble

$N_e \leftarrow \lceil T/T_e \rceil$ // the number of ensembles
$\varepsilon_e \leftarrow \varepsilon/N_e$ // privacy budget for each tree
**for** $t = 1$ **to** $T$ **do**
  Update gradients of all training instances on loss $l$.
  $t_e \leftarrow t \mod T_e$
  **if** $t_e == 1$ **then**
    $I \leftarrow \mathcal{D}$ // initialize the dataset for an ensemble
  Randomly pick $(\frac{|\mathcal{D}|\eta(1-\eta)^{t_e}}{1-(1-\eta)^{T_e}})$ instances from $I$ to constitute the subset $I_t$.
  $I \leftarrow I - I_t$
  TrainSingleTree (dataset = $I_t$,
    maximum depth = $Depth_{max}$,
    privacy budget = $\varepsilon_e$,
    $\Delta G = \frac{3\lambda+2}{(\lambda+1)(\lambda+2)} g_l^{*2}$,
    $\Delta V = \min(\frac{g_l^*}{1+\lambda}, 2g_l^*(1-\eta)^{t-1}))$.

**Output:** $\varepsilon$-differentially private GBDTs

training a differentially private tree. 3) **SEQ**: we extend the previous approach on decision trees (Liu et al. 2018) that aggregates differentially private decision trees using sequential composition. Since the original study does not provide sensitivity bounds in GBDTs (Liu et al. 2018), we set $\Delta G = \frac{3\lambda+2}{(\lambda+1)(\lambda+2)} g^{*2}$ and $\Delta V = \frac{g_m}{1+\lambda}$ in SEQ using our GDF technique.

We implemented DPBoost based on LightGBM[1]. Our experiments are conducted on a machine with one Xeon W-2155 10 core CPU. We use 10 public datasets in our evaluation. The details of the datasets are summarized in Table 1. There are eight real-world datasets and two synthetic datasets (i.e., synthetic_cls and synthetic_reg). The real-world datasets are available from the LIBSVM website[2]. The synthetic datasets are generated using scikit-learn[3] (Pedregosa et al. 2011). We show test errors and RMSE (root mean square error) for the classification and regression task, respectively. The maximum depth is set to 6. The regularization parameter $\lambda$ is set to 0.1. We use square loss function and the threshold $g_l^*$ is set to 1. For the regression task, the labels are scaled into $[-1, 1]$ before training (the reported RMSE is re-scaled). We use 5-fold cross-validation for model evaluation. The number of trees inside an ensemble is set to 50 in DPBoost. We have also tried the other settings for the number of trees inside an ensemble (e.g., 20 and 40). The experiments are available in Appendix C.

[1]https://github.com/microsoft/LightGBM

[2]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[3]https://scikit-learn.org/stable/datasets/index.html#sample-generators

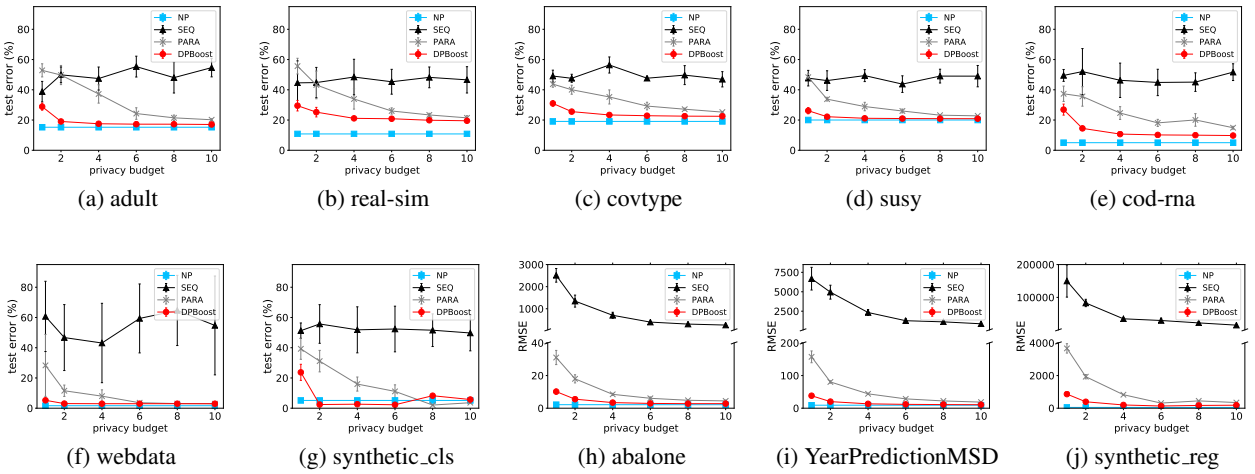
Table 1: Datasets used in the experiments.

| datasets | #data | #features | task |
|---|---|---|---|
| adult | 32,561 | 123 | classification |
| real-sim | 72,309 | 20,958 | |
| covtype | 581,012 | 54 | |
| susy | 5,000,000 | 18 | |
| cod-rna | 59,535 | 8 | |
| webdata | 49,749 | 300 | |
| synthetic_cls | 1,000,000 | 400 | |
| abalone | 4,177 | 8 | regression |
| YearPredictionMSD | 463,715 | 90 | |
| synthetic_reg | 1,000,000 | 400 | |

## 4.1 Test Errors

We first set the number of ensembles to one in DPBoost and the number of trees to 50 for all approaches. Figure 2 shows the test errors of four approaches with different $\varepsilon$ (i.e., 1, 2, 4, 6, 8, and 10). We have the following observations. First, SEQ performs very badly on all the datasets. The test errors are around 50% in the classification task. With only sequential composition, each tree in SEQ gets a very small privacy budget. The noises in SEQ are huge and lead to high test errors in the prediction. Second, DPBoost can always outperform PARA and SEQ especially when the given budget is small. DPBoost outperforms PARA, mainly because our tightening bounds on sensitivity allows us to use a smaller noise to achieve differential privacy. When the privacy budget is one, DPBoost can achieve 10% lower test errors on average in the classification task and significant reduction on RMSE in the regression tasks. Moreover, the variance of DPBoost is usually close to zero. DPBoost is very stable compared with SEQ and PARA. Last, DPBoost can achieve competitive performance compared with NP. The results of DPBoost and NP are quite close in many cases, which show high model quality of our design.

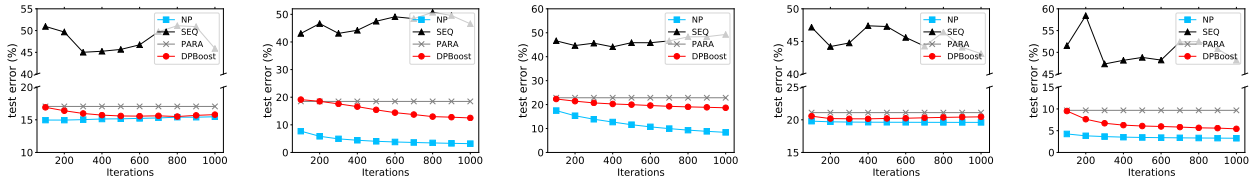
To show the effect of boosting, we increase the number of ensembles to 20 and the maximum number of trees to 1000. The privacy budget for each ensemble is set to 5. For fairness, the total privacy budget for SEQ and PARA is set to 100 to achieve the same privacy level as DPBoost. We choose the first five datasets as representatives. Figure 3 shows the convergence curves of four approaches. First, since the privacy budget for each tree is still small, the errors of SEQ are very high. Second, since PARA takes a half of the unused instances at each iteration, it can only train a limited number of trees until the unused instances are too few to train an effective tree (e.g., about 20 trees for dataset *SUSY*). Then, the curve of PARA quickly becomes almost flat and the performance cannot increase as the iteration grows even given a larger total privacy budget. Last, DPBoost has quite good behavior of reducing test errors as the number of trees increases. DPBoost can continue to decrease the accuracy loss and outperform PARA and SEQ even more, which demonstrate the effectiveness of our privacy budget allocation. Also, DPBoost can preserve the effect of boosting well.




(a) adult (b) real-sim (c) covtype (d) susy (e) cod-rna

(f) webdata (g) synthetic_cls (h) abalone (i) YearPredictionMSD (j) synthetic_reg

Figure 2: Comparison of the test errors/RMSE given different total privacy budgets. The number of trees is set to 50.




(a) adult (b) real-sim (c) covtype (d) susy (e) cod-rna

Figure 3: Comparison of test error convergence. The number of trees is set to 1000.

Table 2: Training time per tree (second) of DPBoost and NP.

| datasets | DPBoost | NP |
|---|---|---|
| adult | 0.019 | 0.007 |
| real-sim | 2.97 | 0.82 |
| covtype | 0.085 | 0.044 |
| SUSY | 0.38 | 0.32 |
| cod-rna | 0.016 | 0.009 |
| webdata | 0.032 | 0.013 |
| synthetic_cls | 1.00 | 0.36 |
| abalone | 2.95 | 2.85 |
| YearPrediction | 0.38 | 0.12 |
| synthetic_reg | 0.96 | 0.36 |

### 4.2 Training Time Efficiency

We show the training time comparison between DPBoost and NP. The computation overhead of our approach mainly comes from the exponential mechanism, which computes a probability for each gain. Thus, this overhead depends on the number of split values and increases as the number of dimensions of training data increases. Table 2 shows the average training time per tree of DPBoost and NP. The setting is the same as the second experiment of Section 4.1. The training time per tree of DPBoost is comparable to NP in many cases (meaning that the overhead can be very small), or about 2 to 3 times slower than NP in other cases. Nevertheless, the training of DPBoost is very fast. The time per tree of DPBoost is no more than 3 seconds in those 10 datasets.

## 5 Conclusions

Differential privacy has been an effective mechanism for protecting data privacy. Since GBDT has become a popular and widely used training system for many machine learning and data mining applications, we propose a differentially private GBDT algorithm called DPBoost. It addresses the limitations of previous works on serious accuracy loss due to loose sensitivity bounds and ineffective privacy budget allocations. Specifically, we propose gradient-based data filtering and geometric leaf clipping to control the training process in order to tighten the sensitivity bound. Moreover, we design a two-level boosting framework to well exploit both the privacy budget and the effect of boosting. Our experiments show the effectiveness and efficiency of DPBoost.

## Acknowledgements

This work is supported by a MoE AcRF Tier 1 grant (T1 251RES1824) and a MOE Tier 2 grant (MOE2017-T2-1-122) in Singapore. The authors thank Xuejun Zhao for her discussion.

## References


Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 23nd ACM SIGSAC conference on computer and communications security*, 308–318. ACM.

Acs, G.; Melis, L.; Castelluccia, C.; and De Cristofaro, E. 2018. Differentially private mixture of generative neural networks. *IEEE TKDE* 31(6):1109–1121.

Burges, C. J. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11(23-581):81.

Chaudhuri, K., and Monteleoni, C. 2009. Privacy-preserving logistic regression. In *Advances in neural information processing systems*, 289–296.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *SIGKDD*, 785–794. ACM.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9(3–4):211–407.

Dwork, C. 2011. Differential privacy. *Encyclopedia of Cryptography and Security* 338–340.

Feng, J.; Yu, Y.; and Zhou, Z.-H. 2018. Multi-layered gradient boosting decision trees. In *Advances in neural information processing systems*, 3551–3561.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 1322–1333. ACM.

Friedman, A., and Schuster, A. 2010. Data mining with differential privacy. In *SIGKDD*, 493–502. ACM.

Friedman, J. H. 2002. Stochastic gradient boosting. *Computational statistics & data analysis* 38(4):367–378.

He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 1–9. ACM.

Jiang, J.; Cui, B.; Zhang, C.; and Fu, F. 2018. Dimboost: Boosting gradient boosting decision tree to higher dimensions. In *Proceedings of the 2018 International Conference on Management of Data*, 1363–1376. ACM.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 3146–3154.

Li, Q.; Wen, Z.; Wu, Z.; Hu, S.; Wang, N.; and He, B. 2019. Federated learning systems: Vision, hype and reality for data privacy and protection. *arXiv preprint arXiv:1907.09693*.

Li, Q.; Wen, Z.; and He, B. 2019. Practical federated gradient boosting decision trees. *AAAI-20, arXiv preprint arXiv:1911.04206*.

Liu, X.; Li, Q.; Li, T.; and Chen, D. 2018. Differentially private classification with decision tree ensemble. *Applied Soft Computing* 62:807–816.

Mohammed, N.; Chen, R.; Fung, B.; and Yu, P. S. 2011. Differentially private data release for data mining. In *SIGKDD*, 493–501. ACM.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *JMLR* 12(Oct):2825–2830.

Pennacchiotti, M., and Popescu, A.-M. 2011. A machine learning approach to twitter user classification. In *Fifth International AAAI Conference on Weblogs and Social Media*.

Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V.; and Gulin, A. 2018. Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, 6638–6648.

Richardson, M.; Dominowska, E.; and Ragno, R. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, 521–530. ACM.

Shokri, R., and Shmatikov, V. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 1310–1321. ACM.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

Si, S.; Zhang, H.; Keerthi, S. S.; Mahajan, D.; Dhillon, I. S.; and Hsieh, C.-J. 2017. Gradient boosted decision trees for high dimensional sparse output. In *ICML*, 3182–3190. JMLR. org.

Truex, S.; Liu, L.; Gursoy, M. E.; Yu, L.; and Wei, W. 2018. Towards demystifying membership inference attacks. *arXiv preprint arXiv:1807.09173*.

Wen, Z.; Shi, J.; He, B.; Li, Q.; and Chen, J. 2019. ThunderGBM: Fast GBDTs and random forests on GPUs. *https://github.com/Xtra-Computing/thundergbm*.

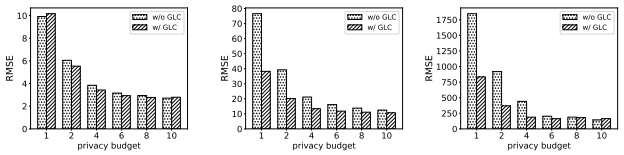
Xiang, T.; Li, Y.; Li, X.; Zhong, S.; and Yu, S. 2018. Collaborative ensemble learning under differential privacy. In *Web Intelligence*, volume 16, 73–87. IOS Press.

Zhao, L.; Ni, L.; Hu, S.; Chen, Y.; Zhou, P.; Xiao, F.; and Wu, L. 2018. Inprivate digging: Enabling tree-based distributed data mining with differential privacy. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 2087–2095. IEEE.

Zhou, J.; Cui, Q.; Li, X.; Zhao, P.; Qu, S.; and Huang, J. 2017. Psmart: Parameter server based multiple additive regression trees system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 879–880. International World Wide Web Conferences Steering Committee.

Table 3: The number of training instances with/without gradient-based data filtering

| | w/ GDF | w/o GDF | filtered ratio |
|---|---|---|---|
| abalone | 3340 | 3292 | 1.44% |
| YearPredictionMSD | 370902 | 340989 | 8.06% |
| sklearn_reg | 800000 | 799999 | 0 |




(a) abalone (b) YearPrediction (c) synthetic_reg

Figure 4: The effect of geometric leaf clipping

## A Proof of Lemma 2

*Proof.* Consider two adjacent instance sets $I_1 = \{\mathbf{x}_i\}_{i=1}^n$ and $I_2 = I_1 \cup \{\mathbf{x}_s\}$ that differ in a single instance. We have

$$\begin{aligned} \Delta V &= |-\frac{\sum_{i=1}^n g_i}{n+\lambda} - (-\frac{\sum_{i=1}^n g_i + g_s}{n+1+\lambda})| \\ &= |\frac{(n+\lambda)g_s - \sum_{i=1}^n g_i}{(n+\lambda)(n+1+\lambda)}| \end{aligned} \tag{13}$$

When $g_s = g^*$ and $\sum_{i=1}^n g_i = -ng^*$, the above function can achieve maximum. Thus, we have

$$\Delta V \leq \frac{(2n+\lambda)g^*}{(n+\lambda)(n+1+\lambda)} \leq \frac{g^*}{1+\lambda} \tag{14}$$

□

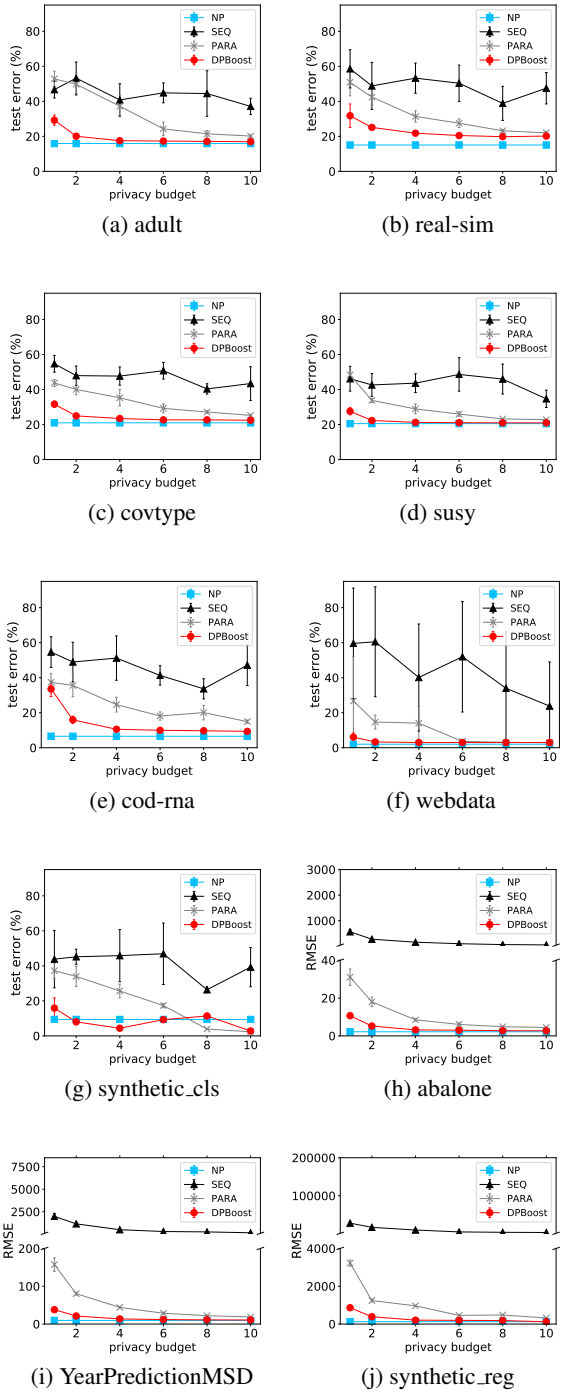
## B Experimental Study on GDF and GLC

We use the regression tasks as the examples to study the effect of gradient-based data filtering and geometric leaf clipping. Table 3 shows the number of instances using gradient-based data filtering. As we can see, the percentage of the filtered instances is small, which is no more than 8%. Thus, the filtering strategy will not produce much approximation error according to Theorem 5.

Figure 4 shows the results with and without geometric leaf clipping in our DPBoost. Except for one case that budget is equal to 1 in *adult*, the geometric leaf clipping can always improve the performance of DPBoost. The improvement is quite significant in *yearpredicionmsd* and *synthetic_reg*.

## C Additional Experiments

Here we show the results of a different number of trees inside an ensemble (i.e., 20 and 40). The results of one ensemble are shown in Figure 5 and Figure 6. As we can see, DPBoost can always outperform SEQ and PARA especially when the given budget is small. Furthermore, the accuracy of DPBoost is close to NP. Then we set the maximum number of trees to 1000. The results are shown in Figure 7 and Figure 8. Still, DPBoost can well exploit the effect of boosting.




(a) adult (b) real-sim

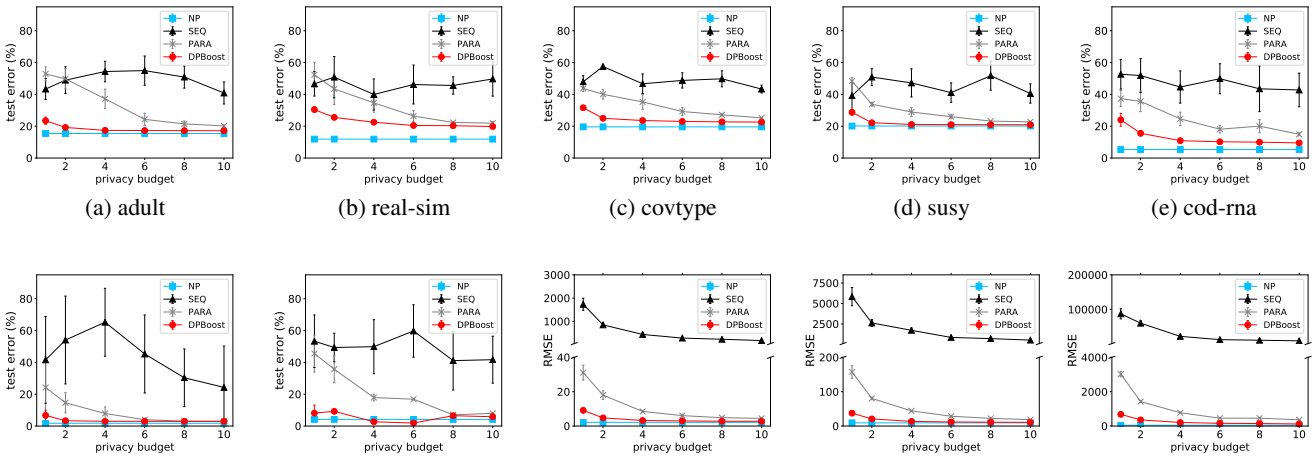
(c) covtype (d) susy

(e) cod-rna (f) webdata

(g) synthetic_cls (h) abalone

(i) YearPredictionMSD (j) synthetic_reg

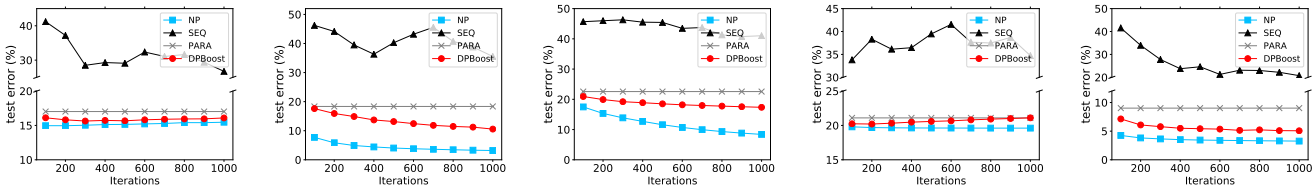
Figure 5: Comparison of the test errors/RMSE given different total privacy budgets. The number of trees and the number of trees inside an ensemble is set to 20.




(a) adult (b) real-sim (c) covtype (d) susy (e) cod-rna

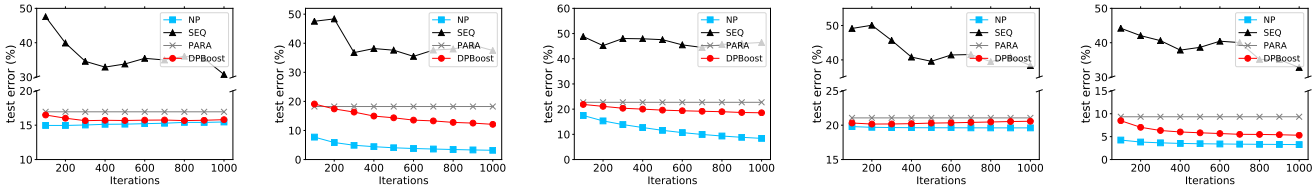
(f) webdata (g) synthetic_cls (h) abalone (i) YearPredictionMSD (j) synthetic_reg

Figure 6: Comparison of the test errors/RMSE given different total privacy budgets. The number of trees and the number of trees inside an ensemble is set to 40.




(a) adult (b) real-sim (c) covtype (d) susy (e) cod-rna

Figure 7: Comparison of test error convergence. The number of trees is set to 1000. The number of trees inside an ensemble is set to 20.




(a) adult (b) real-sim (c) covtype (d) susy (e) cod-rna

Figure 8: Comparison of test error convergence. The number of trees is set to 1000. The number of trees inside an ensemble is set to 40.