Applied Soft Computing 62 (2018) 807–816

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Differentially private classification with decision tree ensemble

Xiaoqian Liu [a], Qianmu Li [a,*], Tao Li [b,c,*], Dong Chen [d]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[b] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[c] School of Computing and Information Sciences, Florida International University, Miami, FL 33199, United States
[d] School of Life Science and Technology, Harbin Institute of Technology, Harbin 150001, China

ARTICLE INFO

Article history:
Received 16 March 2017
Received in revised form 22 July 2017
Accepted 2 September 2017
Available online 12 September 2017

Keywords:
Decision tree
Differential privacy
Ensemble
Maximal vote

ABSTRACT

In decision tree classification with differential privacy, it is query intensive to calculate the impurity metrics, such as information gain and gini index. More queries imply more noise addition. Therefore, a straightforward implementation of differential privacy often yields poor accuracy and stableness. This motivates us to adopt better impurity metric for evaluating attributes to build the tree structure recursively. In this paper, we first give a detailed analysis for the statistical queries involved in decision tree induction. Second, we propose a private decision tree algorithm based on the noisy maximal vote. We also present an effective privacy budget allocation strategy. Third, to boost the accuracy and improve the stableness, we construct the ensemble model, where multiple private decision trees are built on bootstrapped samples. Extensive experiments are executed on real datasets to demonstrate that the proposed ensemble model provides accurate and reliable classification results.

1. Introduction

Data mining has effectively and tremendously enhanced the service in diverse areas, e.g., health care, business analysis, and social media. Nevertheless, with knowledge discovery techniques becoming more and more developed in academia and industry, the leakage of individual privacy is turning into an unprecedentedly worrying problem.

Obviously, individual privacy preservation is of great necessity. On one hand, privacy leakage often brings up moral and legal issues. On the other hand, individuals are more willing to contribute their personal data for analysis when given the promise that their privacy is ensured.

Over the last few decades, there has been significant interest in privacy preservation in a broad set of disciplines, such as statistics, cryptography, and computer science. Two interesting privacy preservation strategies are talked about in [1], including *query restriction* and *data perturbation*.

- *Query restriction* In the query restriction approach, queries are required to obey a special structure, supposedly to prevent the adversary from gaining too much information about specific data instances. In addition, only a relatively small number of queries are allowed. This approach prevents the adversary from guessing the sensitive information through composing several query results (differencing attacks). Nevertheless, the algorithmic procedure for deciding whether a pair of queries constitute a differencing attack is computationally intensive. Given the assumption that the result of some query leaks privacy, refusing the query itself is disclosive. For instance, if the query "Does Alice have cancer?" is refused, the adversary may guess Alice has cancer.
- *Data perturbation* Data perturbation techniques are roughly classified into three groups, including *input perturbation*, *output perturbation*, and *objective perturbation*. In the input perturbation approach, a private database is released rather than the original one. The sensitive information is masked with carefully calibrated randomness. Queries are only allowed on the released dataset. In the output perturbation approach, the database first computes the true result on the original database, then returns a noisy version of it. In the objective perturbation approach, the objective function is perturbed before optimizing classifiers in the setting of machine learning. If the loss and regularizer satisfy certain convexity and differentiability criteria, the process of learning can be made privacy preserving.

In the area of data mining, privacy preservation is about the trade-off between data privacy and utility. Conventional data

* Corresponding authors.
E-mail addresses: lxqlara@gmail.com (X. Liu), qianmu@njust.edu.cn (Q. Li), taoli@cs.fiu.edu (T. Li), peakgrin@gmail.com (D. Chen).

https://doi.org/10.1016/j.asoc.2017.09.010
1568-4946/

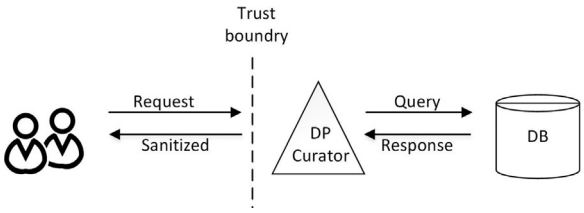

Fig. 1. Data mining with differential privacy.

mining algorithms should be redesigned to preserve sensitive information. Researchers are faced with several difficult problems. First, semantic security in cryptography is inapplicable in privacy preservation, because it cares little about data utility. Second, a concise catch-all definition of privacy is elusive. One of the biggest obstacles is the auxiliary information based on the research of Ganta et al. [2]. It can be gathered cheaply from many channels, e.g., web, public data release, and domain knowledge. The fact is that the adversary can always collect new auxiliary information with new data sources. These information can be composed to comprise the sensitive information. A simple example is that the adversary can deduce if someone has AIDS with the overlapping information between the patient populations of two hospitals. That is the main reason why the anonymization mechanisms [3] fail.

To handle these problems, the cutting-edge privacy definition, i.e., differential privacy [4] is extensively studied. Differential privacy provides robust and provable privacy guarantee. As a randomization method, the extent of privacy is given formally and quantitatively with a numerical metric called privacy budget. The budget measures the distinguishability of the outputs due to the presence of an individual.

The framework of data mining with differential privacy is illustrated in Fig. 1.

The smaller the privacy budget is, the less distinguishable of the algorithm outputs on two adjacent databases (D and $\hat{D}$ are adjacent databases if $\hat{D}$ has one instance absent from D). Hence, it would be more difficult for the adversary to estimate the contribution of some individual. Different from the conventional anonymization approaches in [3], differential privacy is highly robust with the assumption that the adversary has obtained all the auxiliary information.

In data mining with differential privacy, the privacy and algorithmic requirements are considered simultaneously. Only sanitized query results will be released to the data miner. Through shifting the value of privacy budget, the extent of how private the sanitization mechanism is would be under control.

Our contributions: In this paper, we focus on designing differentially private decision tree and its ensemble counterpart to perform the classification task. The *goal* is to guarantee highly accurate and stable performance with privacy preservation. Our work can be illustrated from four aspects.

- Initially, we give detailed statistical query analysis in differentially private decision tree. Based on that too many queries lead to low accuracy, we indicate that a straightforward way of masking each count in the calculation of information gain is impractical. We also demonstrate that trivial count substitution from bottom to top is inappropriate when the tree structure is unknown.
- Second, we present a differentially private decision tree method. Handling count query is easy. Hence, for preserving the internal nodes in the tree, we avoid perturbing the information gain as a whole metric (which is more intractable than count query). Instead, we select the attribute with maximal class vote (the maximal class count among all possible partitions of an attribute) to estimate the impurity. The intuition is that the noise addition process would occur once for a single attribute. Hence, the number of queries is effectively reduced. In addition, for preserving the leaf nodes, the class distribution of all leaves are perturbed. Because there is no overlapping between each leaf, the perturbation is done with small privacy budget. Through adding noise to the returned counts, we make sure that the tree building process satisfies differential privacy.
- Third, we design a budget allocation strategy for assigning different budgets regarding different depths and splitting attribute candidates. When the tree depth is larger and more splitting attribute candidates are available, the true count is easier to be overcome by noise because of further budget division. Therefore, we tend to allocate bigger budgets to these nodes to preserve true information.
- Fourth, we construct the ensemble of private decision trees to boost and smooth the accuracy in a differentially private way. It is known that the decision tree model is non-stable. Therefore, we apply the ensemble trick to improve the classification performance through integrating the decisions of multiple trees. We do extensive experiments to show that the proposed algorithm provides good classification performance under differential privacy.

The rest of the paper is organized as follows. We overview some related work in Section 2 and review the concepts of differential privacy in Section 3. We then give a detailed analysis for building differentially private decision trees in Section 4. We propose our approach to build private decision trees and the ensemble model in Section 5. We illustrate experimental evaluation in Section 6. Finally, we conclude the whole work in Section 7.

2. Related work

In the area of knowledge discovery, a lot of research results under differential privacy have been proposed over the last few decades. As can be seen in [5–11], differentially private data mining applications cover classification, clustering, recommender system, and deep learning. There are also several platforms presented to implement differential privacy. The PINQ platform designed by McSherry [12] presents differentially private data analysis. Analysts make queries on the database through a SQL-like language. The queries are executed without the requirement for privacy preservation expertise. In [13], Blum et al., propose a framework for implementing differential privacy in the setting of statistical query. In the framework, the conventional data mining applications, e.g., principal component analysis, k-means clustering, the perceptron learning, and the ID3 algorithm are redesigned into their differentially private version. In [14], Roy et al., design a framework for privacy-preserving MapReduce computations with untrusted code. In [15], Mohan et al., design a privacy preserving system based on the sample and aggregate framework (SAF) proposed by Nissim et al. [16]. In this way, the query sensitivity is effectively reduced through localizing the computation.

On the topic of decision tree induction, several pieces of work have been done to preserve data owners' privacy. In [17,18], privacy-preserving decision trees are built on unreal datasets. In [19–23], private decision trees are built in multi-party scenario. In [24], Dowd et al., talk about decision tree learning using random substitution, which differs from the way in the setting of statistical query. In [25], decision trees are built on two vertically partitioned datasets for sharing data with privacy preservation. In [26], Friedman et al., propose a k-anonymous decision tree induction approach. Instead of mining patterns on k-anonymous dataset, anonymization is embedded within the tree induction process.

To combine differential privacy and decision tree induction, a series of approaches have been proposed in [13,27–29]. The SuLQ-based ID3 algorithm in [13] is for building private ID3 decision trees through perturbing the count query results in the internal nodes and leaf nodes with the differentially private primitive. However, the SuLQ-based ID3 algorithm suffers from low accuracy due to too much noise addition. To calculate the information gain for a relatively large dataset, the number of queries turns to be huge. In the implementation of differential privacy, the privacy budget should be divided among all those queries. Accordingly, each budget share ϵ_s would be relatively small. However, the magnitude of noise is equal to $Lap(1/\epsilon_s)$ as is given in [27]. It changes inversely with the magnitude of the privacy budget. Smaller budget implies larger noise. As a result, the model suffers from low accuracy.

A refinement method, Algorithm DiffID3 is proposed by Friedman and Schuster [28] under the exponential mechanism [29]. They sample the best attribute to split on according to the noisy selecting probability assigned to each attribute. In this method, further partition of the privacy budget is avoided in each round of partitioning the data instances. However, building single decision tree with randomness often leads to great variance in the classification accuracy. In addition, using count for attribute evaluation in the exponential mechanism causes large integer calculation problems.

To handle the high variance problem, researchers resort to ensemble models. The intuition is that the variance can be effectively decreased through model averaging. In addition, the accuracy is improved because of inner error canceling out.

A differentially private ensemble method is proposed by Rana et al. [30] by relaxing the privacy preservation requirement. They assume that each tree is independent so that the privacy budget will not accumulate. However, the assumption needs more theoretical analysis. Also, in the scenario of binomial classification talked about in [30], the information gain lies in the range of [0, 1]. As is given in [28], the sensitivity of the information gain function is bounded to $log_2(N+1)+1/ln2$, where N denotes the number of data instances. When perturbing the attribute selection score, the noise magnitude should be far larger than the magnitude of the information gain itself. This makes the impurity metric meaningless.

Jagannathan et al., bring up the differentially private random forest model in [31]. In their method, the internal nodes of each tree are selected totally randomly, which means there is no privacy cost. Relying on the "free randomness", no information is gathered from the original database, therefore not privacy leaking. However, in [31], the number of randomly built trees is relatively difficult to balance between the dilemma of accuracy and privacy. Because the tree structure is generated without any leading heuristic, more trees should be built for stable classification results. However, improving the number of trees implies reducing budget share, which hurts the accuracy.

Another related method is given in [32] by Fletcher and Islam. They extend the method in [31] with a pruning strategy based on the signal-to-noise ratio (SNR). The statistics of the internal nodes are estimated through summing up backwardly from the noisy leaves to internal nodes. Nodes with SNR < 1 are pruned since there is a big chance that the true counts are overcome by noise.

Our approach differs from the preceding lines of work in three aspects. First, we avoid using information gain as the impurity metric as it is query intensive. Unlike in building random trees, we use the maximal label vote (masked with noise) as the heuristic to lead the induction process. Second, we design a budget allocation strategy through offering more budget to nodes on deeper levels. This is different from other algorithms where all the counts are given the exactly same privacy budget. Third, most of the previous work executes experiments to demonstrate the accuracy. Whereas we experimentally show that the proposed algorithm provides both accurate and stable performance.

3. Preliminaries

3.1. Differential privacy

Differential privacy uses statistical bounds to limit the privacy loss of an individual incurred by running statistical queries on datasets. It provides a "query and noisy response" framework. Through perturbing the result of a computation, no distinguishable difference is returned. That indicates the adversary can hardly tell whether an individual is in the database or not. In this way, the sensitive information of the individual is safely preserved. We give an overlook of the related definitions as below.

Definition 1 *(ϵ-Differential privacy [4]).* A randomization mechanism P provides ϵ-differential privacy if $Pr[P(D) \in \mathcal{R}] \leq e^{\epsilon} Pr[P(\hat{D}) \in \mathcal{R}]$ for all adjacent databases D and $\hat{D}$. Here $\mathcal{R}$ is the output range.

In the definition of differential privacy, the degree of perturbation is related to the global sensitivity of the query function.

Definition 2 *(Global sensitivity [27]).* Given a function $f : D \to \mathbb{R}^d$ over an arbitrary domain D, the global sensitivity of f is $S(f) = \max_{D \Delta \hat{D}=1} ||(f(D) - f(\hat{D})||_1$.

In the process of perturbation, the magnitude of noise depends on the query function f and the privacy budget ϵ, instead of the database itself. Intuitively, noise grows (and must grow) with the total number of queries. But when the data size grows large, huge quantities of queries become practical. This is because more information is provided when the data size is large, whereas the noise does not increase along with the data size. Differential privacy is the statistical property of the query function. Therefore, it is independent of the computational power and the auxiliary information available to the adversary. This feature ensures the robustness of differentially private algorithms. The sensitive information of a specific individual will not be leaked even if the adversary has acquired the information of all the other individuals in the database.

However, differential privacy is not an absolute guarantee of privacy. In a society that has decided that the benefits of certain databases outweigh the costs, differential privacy ensures that only a limited amount of additional risk is incurred by participating in the socially beneficial databases.

3.2. Satisfying differential privacy

When the query result is real-valued, the Laplace mechanism [27] gives the implementation of differential privacy. A randomization mechanism satisfies ϵ-differential privacy when its output is masked with the randomness drawn from a calibrated Laplace distribution. The randomness is well controlled with the privacy budget ϵ. Formal definition is givens as follows.

Theorem 1 (Laplace mechanism [27]). *Given a function $f : D \to \mathbb{R}^d$ over an arbitrary domain D, a randomization mechanism P provides ϵ-differential privacy when $P(D) = f(D) + Lap(S(f)/\epsilon)$.*

The intuition is that the probability distribution difference due to the absence of an individual is controllable through shifting the privacy budget as illustrated in Fig. 2. Smaller privacy budget leads to smaller gaps between the two distributions. Thus, it is more difficult for the adversary to gain information through composing the returned results.

When the query result is categorical, McSherry et al., propose the exponential mechanism in [29] based on random selection. A quality score is given to each candidate based on the likelihood

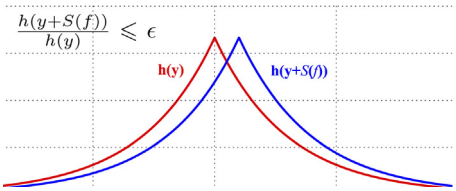

Fig. 2. The noisy output under Laplace mechanism.

of returning this specific output. The higher the quality score of some output is, the more likely it would be returned. The quality function helps to generate a probability distribution over the output domain. To guarantee differential privacy, the probability for each specific output is perturbed according to the sensitivity of the quality function and the privacy budget. Therefore, the chance to choose a certain output is masked. Exponential mechanism favors higher scoring outputs. After feeding the quality of an output to the mechanism, the probability is exponentially exaggerated. Formal definition is givens as follows.

Theorem 2 (Exponential mechanism [29]). *Let* $q : (D^n \times R) \to \mathbb{R}$ *be a quality function. Given a data instance* $d \in D^n$*, assign a quality score to each output* $r \in R$*. A randomization mechanism P provides* ϵ*-differential privacy when* $P(d, q) = \left\{\text{returnrwithprobability} \propto \exp\left(\frac{\epsilon q(d,r)}{2S(q)}\right)\right\}$*,* $q(d, r)$ *denotes the score for choosing an output r, and* $S(q)$ *denotes the sensitivity of the quality function.*

Differential privacy has two composition properties, including sequential composition and parallel composition [12]. When knowledge is discovered on the database, a large number of queries will be issued on joint and disjoint data partitions. The composition properties indicate whether the privacy budget should be accumulated or not in the calculation. The formal explanation of the two composition properties is given as below.

Theorem 3 (Sequential composition [12]). *When a randomization mechanism* P_i *is applied to joint subsets of database D providing* ϵ_i*-differential privacy, the sequential effect of perturbation leads to accumulation of privacy budget. That indicates the whole mechanism provides* $\sum_i \epsilon_i$*-differential privacy.*

The sequential composition property gives the data analysts the chance to issue more than one query to the joint data partitions. This is practical because the data analysts usually need to post a large number of queries on databases with sharing data instances.

Theorem 4 (Parallel composition [12]). *When a randomization mechanism* P_i *is applied to disjoint subsets of the database D, providing* ϵ*-differential privacy, the privacy budget will not accumulate. That indicates the whole mechanism provides* ϵ*-differential privacy.*

Because these data partitions are disjoint, the absence of an individual affects just one partition. The budget needs no accumulation since in the non-overlapping scenario, a specific data instance will avoid being queried repeatedly.

4. Analysis on building private trees

In this section, we give the analysis of difficulties in building a differentially private tree. At first, we give an overview of the formulation of decision trees. The structure of the decision tree can be illustrated as a directed acyclic graph shown in Fig. 3.

Building a decision tree is a heuristic process. A series of if–then rules are generated based on a handful of information metrics, e.g., information gain, gain ratio, and gini index. Without loss of generality, we perform the analysis on information gain. Initially, all data points are assumed to reside at the root node. Branches are grown

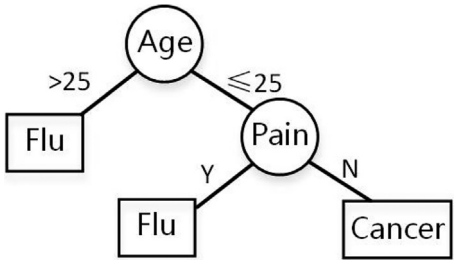

Fig. 3. The structure of decision tree.

iteratively through partitioning all the data instances in a way of selecting the best current internal node. The objective is to get the class distribution purer and purer.

Notations are explained as follows. The attribute set is denoted as $\mathcal{A} = \{A_1, \ldots, A_d\}$ and the dataset is denoted as $\mathcal{D} = \{D_1, \ldots, D_n\}$ over these attributes, where n is the number of instances and d is the number of attributes. Each data instance is associated with a class label c in the label set C. For some splitting attribute A with i potential values $\{V_1, V_2, \ldots, V_i\}$, let n_i denote the count of instances when A takes value V_i. In this branch, let $n_{i,c}$ denote the count of instances taking class label c. All data instances are partitioned according to the attribute which gives the maximum information gain in the ID3 algorithm proposed by Quinlan [33]. Class distribution is evaluated before and after a split to see if it gets purer. The process ends until the attributes run out or the tree has reached the maximal depth.

Under the framework of differential privacy, the conventional decision tree method should be reviewed from the perspective of statistical query. Each time the class distribution is evaluated, a series of count queries will be issued on the dataset.

We analyze how the count query is issued to figure out why the conventional impurity metric calculation leads to poor performance. First, we give the formulation of the problem as below. Given a dataset $\mathcal{D}$ and the class label c from the label set C, the calculation of entropy is given in Eq. (1).

$$H_C(\mathcal{D}) = -\sum_{c \in C} \frac{n_c}{n} \log \frac{n_c}{n} \tag{1}$$

When given the splitting attribute A with i possible values, the dataset $\mathcal{D}$ is partitioned into $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_i\}$. The conditional entropy can be computed as presented in Eq. (2).

$$H_{C|A}(\mathcal{D}) = \sum \frac{n_i}{n} H_C(\mathcal{D}_i) \tag{2}$$

Entropy and conditional entropy are combined to calculate the information gain, as illustrated in Eq. (3). The attribute which brings about the largest information gain is selected as the "best" for partitioning the instances. Through performing the "best attribute" selection recursively, the structure of a decision tree is generated.

$$InfoGain(A, \mathcal{D}) = H_C(\mathcal{D}) - H_{C|A}(\mathcal{D}) \tag{3}$$

4.1. Differential privacy promise for decision tree

In the structure of a decision tree, there are two kinds of nodes, i.e., the internal node for partitioning and the leaf node for assigning class label. The count query is recursively issued over the whole tree. Assume two trees have been built on adjacent databases. Through comparing the counts, the adversary can easily get the sensitive information. For instance, the adversary will know whether the differencing individual has AIDS through comparing the count results under branch "*hasAIDS = true*". In this way, we say the privacy of this individual is compromised.

Differential privacy makes the promise that even if all other data instances have been acquired by the adversary except for this spe-

cific one, the accurate count of instances in each internal node and leaf node still cannot be obtained accurately. Therefore, the adversary is unable to guess the true value of some individual on a specific attribute or class label. This is done through adding Laplacian noise using a scalar parameter $\Delta f/\epsilon$, to make sure the query results are indistinguishable.

However, differential privacy does not ensure that no sensitive information of the data respondents will be leaked at all. But the risk of leakage is controlled with the privacy budget. The smaller the privacy budget is, the safer the sensitive information of individuals will be.

A primitive implementation of differential privacy for decision trees is proposed by Blum et al. [13], illustrated in Algorithm 1. It is proved that to maximize the information gain after splitting on attribute A can be equivalently transformed to maximize $V_A = \sum_{i=1}^{|A|}\sum_{c=1}^{|C|} n_{i,c} \cdot \log \frac{n_{i,c}}{n_i}$. Under the privacy requirements, the true count is replaced with its noisy version as formally given in Eq. (4). In this method, each noise addition step provides ϵ_p-differential privacy.

Algorithm 1. SuLQID3 algorithm

procedure SuLQID3($\mathcal{D}$, $\mathcal{A}$, C, d, B)
  Input: $\mathcal{D}$ denotes a private dataset, $\mathcal{A} = \{A_1, \ldots, A_d\}$ denotes a set of attributes, C denotes the class attribute, d denotes the maximal tree depth, and B denotes the total differential privacy budget
  $\epsilon_p = \frac{B}{2(d+1)}$
  Build_SuLQID3($\mathcal{D}$, $\mathcal{A}$, C, d, B)
end procedure
procedure Build_SuLQID3($\mathcal{D}$, $\mathcal{A}$, C, d, ϵ_p)
  $t = \max_{A \in \mathcal{A}} |A|$
  $\tilde{n} = n + Lap(1/\epsilon_p)$
  if $\mathcal{A} = \emptyset$ or $d = 0$ or $\frac{n}{|A|*|C|} < \frac{\sqrt{2}}{\epsilon_p}$ **then**
    $Partition(\mathcal{D}, \forall c \in C)$
    $\forall c \in C$: $\tilde{n_c} = n_c + Lap(1/\epsilon_p)$
    return a class label for the leaf node with $argmax_c(\tilde{n_c})$
  end if
  for every attribute $A \in \mathcal{A}$ **do**
    $\mathcal{D}_i = Partition(\mathcal{D}, \forall i \in A)$
    $\forall i \in A$: $Partition(\mathcal{D}_i, \forall c \in C)$
    $\tilde{n_i} = n_i + Lap(2 \cdot |\mathcal{A}|/\epsilon_p)$
    $\tilde{n_{i,c}} = n_{i,c} + Lap(2 \cdot |\mathcal{A}|/\epsilon_p)$
    $\tilde{V_A} = \sum_{i=1}^{|A|}\sum_{c=1}^{|C|} \tilde{n_{i,c}} \cdot \log \frac{\tilde{n_{i,c}}}{\tilde{n_i}}$ ($\tilde{n_i}$ and $\tilde{n_{i,c}}$ less than the noise deviation $\frac{\sqrt{2}}{\epsilon_p}$ are skipped)
  end for
  $Partition(\mathcal{D}, \forall i \in argmax_A(\tilde{V_A}))$
  $\forall i \in \bar{A}$: $Subtree_i = Build\_SuLQID3(\mathcal{D}_i, \mathcal{A}\backslash\bar{A}, C, d-1, \epsilon_p)$
  return a tree with a root node labeled $\bar{A}$ and edges labeled i from 1 to $|A|$ each going to $Subtree_i$
end procedure

$$\tilde{V_A} = \sum_{i=1}^{|A|}\sum_{c=1}^{|C|} \tilde{n_{i,c}} \cdot \log \frac{\tilde{n_{i,c}}}{\tilde{n_i}} \quad (4)$$
$$where$$
$$\tilde{n_i} = n_i + Lap(1/\epsilon_p)$$
$$\tilde{n_{i,c}} = n_{i,c} + Lap(1/\epsilon_p)$$

The above algorithm illustrates how the decision tree induction can be transformed for providing differential privacy. It indicates that the query results for deciding whether to partition on a certain attribute, or whether to classify a node with label c, have to be protected with Laplacian noise. In this way, no distinguishable difference will be observed as is required in the definition of differential privacy. However, calculating the above counts leads to noise repeated addition. Therefore, this approach suffers from low accuracy.

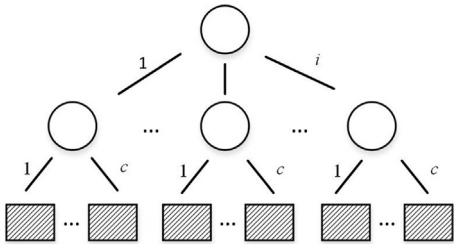

Fig. 4. The subtree structure.

4.2. Query analysis

In this section, we make an analysis of the statistical queries involved in the process of growing decision trees. For an internal node A, denote the instance count as q, and the count corresponding to the ith value of A is denoted as q_i. The query results should satisfy the relation illustrated in Eq. (5).

$$q = q_1 + \cdots + q_i \quad (5)$$

Trivially calculate information gain using noisy counts is budget costing. When choosing the splitting attribute, the sensitivity of the queries accumulates because the subsets partitioned by different attributes are likely to overlap. Hence, the sensitivity for choosing a splitting attribute equals to the number of splitting attribute candidates. For instance, if there are 3 attributes left, then the sensitivity of choosing the next splitting attribute equals to 3. For a subtree with attribute A as root node, based on all its branches and class distributions, the total number of queries roughly equals to $|A|*(|C|+1)+1$. The structure of the subtree is illustrated in Fig. 4. Noisy information gain is calculated based on these query results. However, that is a relatively large number of queries for evaluating just a single attribute.

Another consideration is that evenly dividing the noise does not meet the characteristics of count query. When the splitting attribute is certain, there is a chance to reduce noise. Because for a splitting node, all the partitions are disjoint. Without loss of generality, for attribute A with i different values, the presence of an individual affects at most one partition. That indicates adding or removing an individual influences the query results for q and one of $q_1, \ldots, q_i$ with maximum change of 1. In this case, the sum square error introduced should be $2*(\sqrt{2}/\epsilon)^2 = 4/\epsilon^2$ under the Laplace mechanism, for obtaining the above $i+1$ query results. In the pessimistic method of [13], the sum square error equals to $(i+1)*(\sqrt{2}/\epsilon)^2 = 2(i+1)/\epsilon^2$. In most cases, the relation $2(i+1)/\epsilon^2 \geqslant 4/\epsilon^2$ stands because the number of partition is at least 1. We would think about dividing the noise of magnitude of $4/\epsilon^2$ among these queries. However, the sensitivity share would be less than one for this noise dividing operation, which makes no sense according to the characteristic of count query.

4.3. Accumulative count substitution

Based on the query relaxation analysis presented by Xiao and Tao [34], to save privacy budget under the consistency relation, the noisy counts of each partition and the root node can be obtained through accumulating recursively from bottom to top in the subtree as illustrated in Fig. 5. This process is based on the consistency depicted in Eq. (6). In this way, zero new knowledge is revealed because the noisy results have already been released publicly under

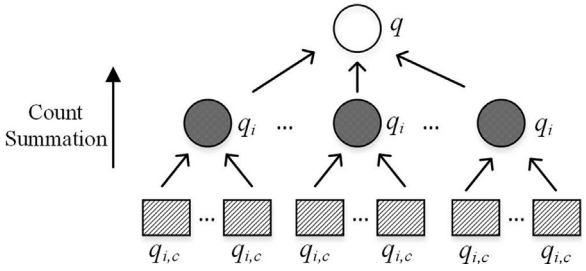

Fig. 5. Bottom-up private count accumulation.

differential privacy. However, the idea is impractical because one has to know the tree structure in advance.

$$\bar{q} = \bar{q_1} + \cdots + \bar{q_i}$$
$$\bar{q_i} = \bar{q_{i,1}} + \cdots + \bar{q_{i,c}} \tag{6}$$

One way is to build trees totally randomly as in [32]. However, we want to select the splitting attribute based on its representative capability (covered with noise), instead of totally casually.

In a nutshell, growing decision trees under differential privacy is faced with several difficulties. First, trivially perturbing all the count queries provides low accuracy. Second, evenly dividing all the noise is hard to explain. Third, noisy count substitution from bottom to top is on the impractical assumption that the tree structure is given.

5. Growing private trees and constructing the ensemble model

In the design of private decision trees, there are two key problems, including how the tree structure is generated, and how the total privacy budget should be allocated. Based on the analysis in Section 4, it is often necessary to make a large number of low granularity queries for the calculation of information gain. Extensive queries result in low accuracy. Therefore in our method, each attribute is estimated with a single statistic, which is the maximal label vote, to avoid noise repeated addition. In this way, the number of queries is reduced effectively.

When the tree structure is being generated, the main task is to select the best attribute to split on. It is more intractable than perturbing the leaf nodes due to data overlapping. When the tree is growing larger, the count will decrease because of data partition. Therefore, we tend to treat the nodes on deeper level with larger budgets. The objective is to prevent the noise from overcoming the true count due to little given budget.

To perturb the results according to the local returned counts, we provide the budget allocation strategy in Section 5.1. The processes of building private decision tree and ensemble are given in Sections 5.2 and 5.3, respectively. The privacy analysis is given in Section 5.4.

5.1. Budget allocation strategy

Assume the privacy budget is ϵ, which is used to perturb both the internal nodes and the leaf nodes. In the selection of internal nodes, ϵ is partitioned into s_i shares based on the kth harmonic number as illustrated in Eq. (7). $s_l = 1$ denotes the budget share for leaf nodes. Thus, we have the total shares as $s_t = s_i + s_l$. We obtain the unit share of privacy budget based on Eq. (8).

$$s_i = \frac{1}{k} + \frac{1}{k-1} + \cdots + 1 \tag{7}$$

$$\epsilon_u = \frac{\epsilon}{s_t} \tag{8}$$

However, there is still further budget division among all the available splitting attribute candidates. For instance, assume the root node is given budget share as $1/k$. Because there are k candidates for the root node, each attribute will be perturbed with noise $Lap(k^2/\epsilon_u)$, instead of $Lap(k/\epsilon_u)$, to make sure the whole process of choosing the best attribute is ϵ_u/k-differentially private. We rank all available attributes with their noisy maximal votes. And we are only interested in k attributes (k is set smaller than d). Therefore, we need to multiply the budget share by k. That implies after k rounds of evaluating the available attributes, the total budget cost is equal to ϵ_u/k. Likewise, for the second splitting attribute, the budget share is equal to $1/(k-1) = (k-1)^*(1/(k-1)^2)$.

The proposed budget allocation strategy implies that the closer an internal node is to the root node, the smaller budget share it will be given. The objective is to balance the noise and information. Internal nodes that are closer to the root are assumed to be more resilient to larger noise. On the other side, leaf nodes are given a treat for privacy budget with a share as $s_l = 1$, because they affect the crucial decision for the classification results. All the leaf nodes are handled at one time when no further partitioning is needed.

5.2. Growing single trees

In our method, we avoid using information gain as the impurity metric due to noise repeated addition. Instead, the inequality relation of the counts in the bottom layer of Fig. 4 is acquired and only the count of the most frequent class label in each branch is released with the differential privacy primitive. This maximal count is termed as maximal vote and used as the estimation statistic for class distribution. We assume that the larger the maximal vote is, the purer the class distribution is. In this way, only one query result needs to be masked for evaluating the contribution of one attribute for decreasing the impurity. Compared to the $|A|^*(|C|+1)+1$ queries in the previous method, the number of queries has been effectively reduced.

The tree is grown through recursively selecting the best attribute based on noisy maximal vote. We denote this method as *MaxTree* as illustrated in Algorithm 2. As is mentioned in Section 5.1, the privacy budget cost of *MaxTree* algorithm is ϵ.

Algorithm 2. MaxTree algorithm

procedure MaxTree$\mathcal{D}$, $\mathcal{A}$, C, k, ϵ
  Input: $\mathcal{D}$ denotes a dataset, $\mathcal{A} = \{A_1, \ldots, A_d\}$ denotes a set of attributes, C denotes the class label set, k denotes the pre-defined number of attributes, s_t denotes the total budget share, and ϵ denotes the privacy budget
  $\epsilon_l = \frac{\epsilon}{s_t}$
  PTree($\mathcal{D}$, $\mathcal{A}$, C, k, ϵ)
end procedure
procedure PTree$\mathcal{D}$, $\mathcal{A}$, C, k, ϵ
  $\epsilon_i = \frac{\epsilon}{s_t} * \frac{1}{k^2}$
  if $\mathcal{A} = \emptyset$ or $k = 0$ **thne**
Partition($\mathcal{D}$, $\forall c \in C$)
$\forall c \in C$: classify the node according to $arg\max_c(n_c + Lap(1/\epsilon_l))$
  end if
  for every attribute $A \in \mathcal{A}$ **do**
$\forall c \in C$ and $\forall i \in A$, $score_A = \max(n_{i,c}) + Lap(1/\epsilon_i)$
  end for
  $\forall A \in \mathcal{A}, \bar{A} = argmax_A(score_A)$
  Partition($\mathcal{D}$, $\forall i \in \bar{A}$)
  $\forall i \in \bar{A}$: $Subtree_i =$ *PTree*($\mathcal{D}_i$, $\mathcal{A} \backslash \bar{A}$, C, $k-1$, ϵ)
  return a tree with a root node labeled $\bar{A}$ and edges labeled i from 1 to $|\bar{A}|$ each going to $Subtree_i$
end procedure

It is obvious that all the leaf nodes have no joint data instances. For perturbing the class distribution of each leaf, the privacy budget

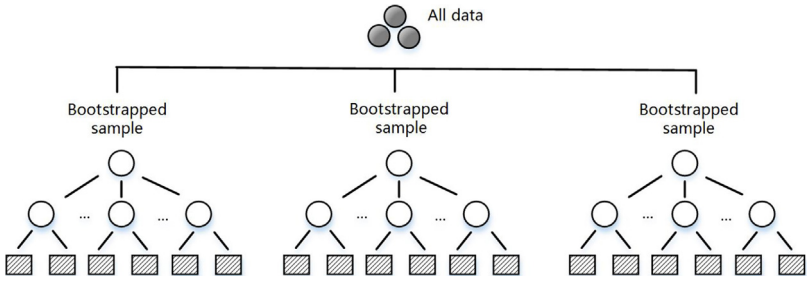

Fig. 6. The ensemble of trees.

share is set to be ϵ_l as given in Algorithm 2. This procedure is a trivial operation of adding random noise of magnitude $Lap(1/\epsilon_l)$ to each count.

5.3. The ensemble of private trees

As a heuristic induction method, decision tree is not stable. The high variance problem is even worse due to the extra randomness we introduce. In order to reduce model variance, the bagging (bootstrapped aggregating) technique [35] is used to construct ensemble model consisting of multiple trees on randomly sampled subsets of the original dataset. The process is shown in Fig. 6. For a test instance, we calculate the label vote among all the trees and return the label with maximal vote. The idea is to take the advantage that differential privacy is immune to post processing. Therefore, the ensemble procedure does not hurt differential privacy. Through constructing the ensemble, the model is more resilient to the introduced randomness and more reliable. Empirically, the classification performance shows great improvement in accuracy and reduction in variance.

Because each tree is built on a bootstrapped sample of the original dataset, there are joint data instances between multiple samples. We denote the total privacy budget as B, and the number of trees as n_{tr}. Each tree gets privacy budget as B/n_{tr}. To put the preceding algorithms together, we propose the *MaxForest* algorithm for constructing the differentially private ensemble model. This algorithm is illustrated in Algorithm 3.

Algorithm 3. MaxForest algorithm

```
Draw n_tr bootstrapped samples D_1, ..., D_ntr from the original dataset D
for each D_i in D do
    procedure MaxTree(D_i, A, C, d, B/n_tr)
    end procedure
end for
classify instance according to label vote of the ensemble
```

5.4. Privacy analysis

Given privacy budget ϵ for a private decision tree, we demonstrate the tree building process preserves ϵ-differential privacy.

Theorem 5. *In Algorithm 2, the private decision tree building process with noisy maximal votes and noisy leaf nodes provides ϵ-differential privacy.*

As is mentioned in Section 5.1, the budget allocation for selecting the root node is equal to $1/k * \epsilon/s_t$. Likewise, we obtain all the budget costs for other internal nodes and sum them up as in Eq. (9).

$$\begin{aligned}\epsilon_{in} &= \frac{\epsilon}{s_t} * \left(\frac{1}{k} + \frac{1}{k-1} + \cdots + 1\right) \\ &= \frac{\epsilon}{s_t} * s_i\end{aligned} \tag{9}$$

We also have the budget spent on the leaf nodes, which is equal to $\epsilon * 1/s_t$. Based on the sequential composition property of differential privacy, the privacy budget should accumulate. The total privacy budget for selecting splitting attributes in a tree is estimated in Eq. (10).

$$\epsilon * \left(s_i/s_t + 1/s_t\right) = \epsilon \tag{10}$$

As a conclusion, Algorithm 2 provides ϵ-differential privacy. For building the ensemble, the total privacy budget is equal to B. There is data overlapping in the bootstrapped samples. Therefore, the budget accumulates among all the trees. Let $B = n_{tr} * \epsilon$. Theorem 6 is natural.

Theorem 6. *With the total privacy budget B, Algorithm 3 provides B-differential privacy.*

Table 1
Description of real datasets.

Name	Number of instances	Number of attributes	Number of classes
mushroom	8124	23	2
adult	32561	15	2

6. Experimental results

Experiments are executed on real datasets, including *mushroom* and *adult* from the UCI Machine Learning Repository [36] for classification on Weka [37]. The *mushroom* dataset contains the gilled mushroom species information, and the *adult* dataset contains the US census data. Description of the datasets is given in Table 1. In the decision tree induction, the numerical attributes are discretized into five equal bins.

The classification performance is estimated using accuracy on paired t-test with 95% confidence intervals under ten-folds cross validation. Each experiment is run for 100 times.

We compare the performance of the proposed methods, denoted as *MaxTree* and *MaxForest* with those in [31,32], denoted as *JDPForset* and *SNRForest*, respectively. There are clear differences in internal node selection and in privacy budget allocation between the above methods. However, they are close enough to compare.

6.1. Privacy budget effect

We observe the accuracy through varying the privacy budget from 0.01 to 1.5. The results are summarized in Figs. 7 and 8, respectively. 10 trees are built in related algorithms expect for Algorithm *MaxTree*, and k is set from the sequence of {3, 3, 3, 5, 10, 13, 15, 20} for Algorithm *MaxTree* and *MaxForest* to provide good results. In comparison, all attributes are used in Algorithm *JDPForset* and *SNRForest*.

As a single tree model, the results of Algorithm *MaxTree* show obvious unstableness on both datasets, whereas the other three all show improvement with the increasing budget. On the *mushroom* dataset, when the budget is larger than 0.01, Algorithm *MaxForest* gives better results than *JDForest* and *SNRForest*. However, when the budget is smaller than 0.01, *MaxForest* achieves poorer results.

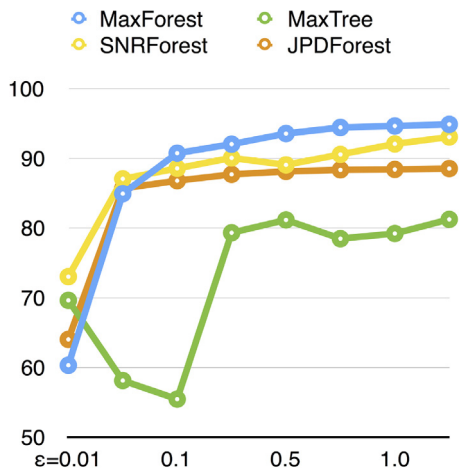

Fig. 7. The effect of varying privacy budget on mushroom dataset.

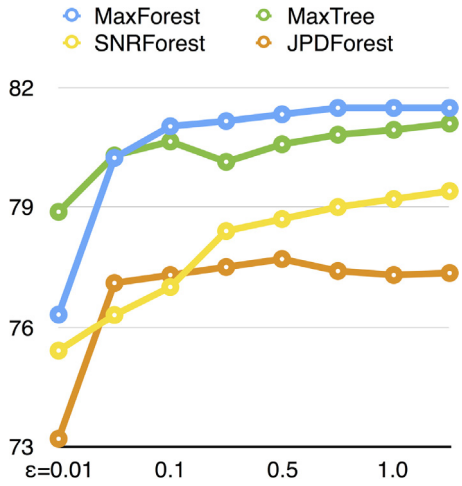

Fig. 8. The effect of varying privacy budget on adult dataset.

Table 2
Accuracy and standard deviation with varying privacy budgets on mushroom dataset.

ϵ	MaxForest	MaxTree
0.01	60.31(±11.92)	69.60(±12.32)
0.05	84.87(±5.74)	58.12(±9.68)
0.1	90.68(±3.31)	55.46(±8.13)
0.25	91.97(±2.15)	79.27(±11.25)
0.5	93.48(±2.26)	81.11(±11.13)
0.75	94.35(±1.91)	78.43(±11.25)
1.0	94.56(±1.95)	79.16(±11.22)
1.5	94.80(±2.00)	81.19(±10.48)

On the *adult* dataset, the performance of both *MaxTree* and *MaxForest* is better than the competing algorithms even when ϵ is small, as shown in Fig. 8. In addition, Algorithm *MaxTree* performs better than *MaxForest* when $\epsilon = 0.01$. It is because the accuracy enhancement of the ensemble is overcome by the degeneration brought by partitioning a tiny budget.

6.2. The effectiveness of the ensemble

To evaluate the effect of the ensemble step, we compare the accuracy and variance statistics of Algorithm *MaxTree* and *MaxForest*. The number of trees and k are set as the same as in Section 6.1. In comparison, a single decision tree is built in Algorithm *MaxTree*. The privacy budget takes different values as shown in Tables 2 and 3.

Table 3
Accuracy and standard deviation with varying privacy budgets on adult dataset.

ϵ	MaxForest	MaxTree
0.01	76.31(±1.96)	78.88(±1.89)
0.05	80.23(±0.82)	80.29(±0.72)
0.1	81.02(±0.65)	80.64(±0.57)
0.25	81.15(±0.49)	80.13(±0.55)
0.5	81.32(±0.48)	80.57(±0.51)
0.75	81.48(±0.40)	80.81(±0.51)
1.0	81.48(±0.42)	80.93(±0.42)
1.5	81.48(±0.42)	81.09(±0.40)

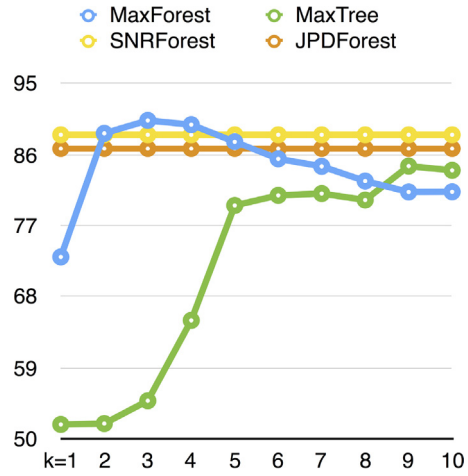

Fig. 9. The effect of varying attribute numbers on mushroom dataset.

Note that Algorithm *MaxTree* builds private decision tree with noisy estimation on internal nodes and leaf nodes, instead of totally random guessing. Therefore unlike the random tree method, Algorithm *MaxTree* can serve as an independent method.

Table 2 and 3 show the results on the *mushroom* dataset and the *adult* dataset, respectively. Based on the variance statistics, both algorithms return unstable results when the privacy budget is small. The variance decreases along with the increasing budget. On dataset *mushroom*, Algorithm *MaxForest* sees a significant reduction in variance, whereas *MaxTree* only shows a relatively small one. On dataset *adult*, both algorithms show high stableness. Algorithm *MaxForest* provides slightly better variance and better accuracy.

Conclude from the above comparison, the ensemble trick improves the classification accuracy and stableness significantly. The idea is that constructing the ensemble model provides the chance to cancel out errors and averages the unstableness. However, there is a tension between model improvement and information loss because constructing the ensemble causes privacy budget further division. Therefore, building the ensemble may bring about unacceptable tiny budgets, thus significantly accuracy degeneration. That is why when the privacy budget is relatively small, Algorithm *MaxForest* shows poorer results than Algorithm *MaxTree*.

6.3. The attribute number effect

In this section, we observe the effect of varying attribute numbers as it affects the budget allocation. Since all the attributes are used to build random trees in Algorithm *JDPForest* and *SNRForest*, we use horizontal lines to illustrate their performance. The number of trees is set to be 10, and the privacy budget is set to be 0.1, which is rather small.

The results of varying attribute numbers on dataset *mushroom* and *adult* are shown in Figs. 9 and 10, respectively. As can be

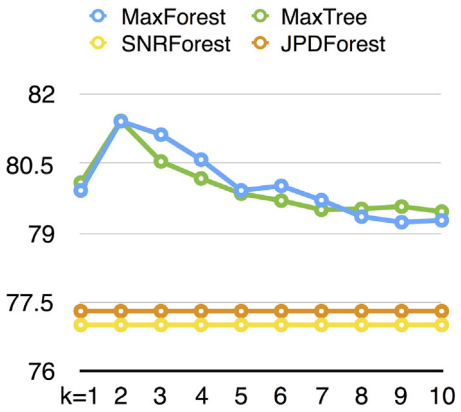

Fig. 10. The effect of varying attribute numbers on adult dataset.

seen from the results, there is an improvement in accuracy when the number of attributes is promoted. However, the performance reaches a peak around 3, which is relatively small since the total budget is small. That implies when the noise is already large, building a deeper tree does not help the accuracy promotion. Therefore, shallower trees are advised in this case.

Based on the above experiments, we show that our algorithm advances its competing algorithms in classification accuracy. Through constructing the ensemble of private decision trees, both accuracy and stableness are improved.

7. Conclusion

In this paper, we give the statistical query analysis in building decision trees. We also propose the algorithms to build private decision trees and ensemble under differential privacy. In the process of building a private tree, internal nodes are selected based on the noisy maximal vote. We also design a budget allocation strategy so that less noise will be added in larger depth to balance between the true counts and noise. For leaf nodes, the vote of each class is masked with Laplacian noise. With the two procedures above, we make sure that growing a private decision tree preserves differential privacy. Furthermore, to boost the accuracy and reduce the variance, we propose the ensemble model. The final classification result is set to be the label vote of multiple private trees. Extensive experiments show that the proposed algorithm provides satisfactory accuracy and stableness on real datasets.

Acknowledgements

The authors would like to thank the editor and reviewers for their insightful comments and suggestions, which helped improve the paper significantly. The work was supported in part by, the Fundamental Research Funds for the Central Universities (No. 30916015104), Chinese National Natural Science Foundation under grants 91646116, National Key Research and Development Program (No. 2016YFE0108000), the Key Research and Development Program (Nos. SBE2017741114 and SBE2017030519), and the Scientific and Technological Support Project (Society) of Jiangsu Province (No. BE2016776).

References

[1] N.R. Adam, J.C. Worthmann, Security-control methods for statistical databases: a comparative study, ACM Comput. Surv. 21 (4) (1989) 515–556.
[2] S.R. Ganta, S.P. Kasiviswanathan, A. Smith, Composition attacks and auxiliary information in data privacy, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 265–273.
[3] L. Sweeney, k-anonymity: a model for protecting privacy, Int. J. Uncertain. Fuzziness Knowl. Based Syst. 10 (5) (2002) 557–570.
[4] C. Dwork, Differential privacy, in: 33rd International Colloquium on Automata, Languages and Programming, Part II (ICALP 2006), Springer Verlag, 2006, pp. 1–12.
[5] C. Dwork, Differential privacy: a survey of results, in: International Conference on Theory and Applications of Models of Computation, Springer, 2008, pp. 1–19.
[6] R. Bassily, A. Smith, A. Thakurta, Private empirical risk minimization: efficient algorithms and tight error bounds, in: 2014 IEEE 55th Annual Symposium on in Foundations of Computer Science (FOCS), IEEE, 2014, pp. 464–473.
[7] K. Chaudhuri, C. Monteleoni, A.D. Sarwate, Differentially private empirical risk minimization, J. Mach. Learn. Res. 12 (2011) 1069–1109.
[8] H. Li, L. Xiong, X. Jiang, J. Liu, Differentially private histogram publication for dynamic datasets: an adaptive sampling approach, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, Melbourne, 2015, pp. 1001–1010.
[9] L. Chen, T. Yu, R. Chirkova, Wavecluster with Differential Privacy, 2015, arXiv preprint arXiv:1508.00192.
[10] F. McSherry, I. Mironov, Differentially private recommender systems: building privacy into the net, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 627–636.
[11] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 308–318.
[12] F.D. McSherry, Privacy integrated queries: an extensible platform for privacy-preserving data analysis, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, ACM, 2009, pp. 19–30.
[13] A. Blum, C. Dwork, F. McSherry, K. Nissim, Practical privacy: the SuLQ framework, in: Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2005, pp. 128–138.
[14] I. Roy, S.T. Setty, A. Kilzer, V. Shmatikov, E. Witchel, Airavat: security and privacy for MapReduce NSDI, 10, 2010, pp. 297–312.
[15] P. Mohan, A. Thakurta, E. Shi, D. Song, D. Culler, GUPT: privacy preserving data analysis made easy, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012, pp. 349–360.
[16] K. Nissim, S. Raskhodnikova, A. Smith, Smooth sensitivity and sampling in private data analysis, in: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, ACM, 2007, pp. 75–84.
[17] P.K. Fong, J.H. Weber-Jahnke, Privacy preserving decision tree learning using unrealized data sets, IEEE Trans. Knowl. Data Eng. 24 (2) (2012) 353–364.
[18] W. Du, Z. Zhan, Using randomized response techniques for privacy-preserving data mining, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 505–510.
[19] Z. Teng, W. Du, A hybrid multi-group privacy-preserving approach for building decision trees, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2007, pp. 296–307.
[20] E. Suthampan, S. Maneewongvatana, Privacy preserving decision tree in multi party environment, in: Asia Information Retrieval Symposium, Springer, 2005, pp. 727–732.
[21] J. Vaidya, C. Clifton, Privacy-preserving decision trees over vertically partitioned data, in: IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2005, pp. 139–152.
[22] J. Vaidya, C. Clifton, M. Kantarcioglu, A.S. Patterson, Privacy-preserving decision trees over vertically partitioned data, ACM Trans. Knowl. Discov. Data 2 (3) (2008) 14.
[23] F. Emekçi, O.D. Sahin, D. Agrawal, A. El Abbadi, Privacy preserving decision tree learning over multiple parties, Data Knowl. Eng. 63 (2) (2007) 348–361.
[24] J. Dowd, S. Xu, W. Zhang, Privacy-preserving decision tree mining based on random substitutions, in: Emerging Trends in Information and Communication Security, Springer, 2006, pp. 145–159.
[25] W. Du, Z. Zhan, Building decision tree classifier on private data, in: Proceedings of the IEEE International Conference on Privacy, Security and Data Mining, vol. 14, Australian Computer Society, Inc., 2002, pp. 1–8.
[26] A. Friedman, A. Schuster, R. Wolff, k-anonymous decision tree induction, in: Knowledge Discovery in Databases: PKDD, 2006, pp. 151.
[27] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of Cryptography, Springer, 2006, pp. 265–284.
[28] A. Friedman, A. Schuster, Data mining with differential privacy, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 493–502.
[29] F. McSherry, K. Talwar, Mechanism design via differential privacy, in: 48th Annual IEEE Symposium on Foundations of Computer Science, 2007. FOCS'07, IEEE, 2007, pp. 94–103.
[30] S. Rana, S.K. Gupta, S. Venkatesh, Differentially private random forest with high utility, in: 2015 IEEE International Conference on Data Mining (ICDM), IEEE, 2015, pp. 955–960.
[31] G. Jagannathan, K. Pillaipakkamnatt, R.N. Wright, A practical differentially private random decision tree classifier, in: IEEE International Conference on in Data Mining Workshops, 2009. ICDMW'09, 2009, pp. 114–121.

[32] S. Fletcher, M.Z. Islam, A differentially private random decision forest using reliable signal-to-noise ratios, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2015, pp. 192–203.
[33] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[34] X. Xiao, Y. Tao, Output perturbation with query relaxation, Proceedings of the VLDB Endowment 1 (1) (2008) 857–869.
[35] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[36] M. Lichman, UCI Machine Learning Repository, 2013, Available: http://archive.ics.uci.edu/ml.
[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.