TRANSACTIONS ON DATA PRIVACY 5 (2012) 273–295

A Practical Differentially Private Random Decision Tree Classifier

Geetha Jagannathan*, **Krishnan Pillaipakkamnatt****, **Rebecca N. Wright*****

*Department of Computer Science, Columbia University, NY, USA.

**Department of Computer Science, Hofstra University, Hempstead, NY, USA.

***Department of Computer Science, Rutgers University, New Brunswick, NJ, USA.

E-mail: geetha@cs.columbia.edu, csckzp@hofstra.edu, rebecca.wright@rutgers.edu

Abstract. In this paper, we study the problem of constructing private classifiers using decision trees, within the framework of differential privacy. We first present experimental evidence that creating a differentially private **ID3** tree using differentially private low-level queries does not simultaneously provide good privacy and good accuracy, particularly for small datasets.

In search of better privacy and accuracy, we then present a differentially private decision tree ensemble algorithm based on random decision trees. We demonstrate experimentally that this approach yields good prediction while maintaining good privacy, even for small datasets. We also present differentially private extensions of our algorithm to two settings: (1) new data is periodically appended to an existing database and (2) the database is horizontally or vertically partitioned between multiple users.

Keywords. Differential privacy; Classifiers; Ensembles; Distributed data mining

1 Introduction

Data analysis and machine learning have led to the ability to improve customer service, streamline business processes, apportion scarce resources more efficiently, and more. At the same time, due to the widespread availability and use of data, there are significant (and growing) concerns about individual privacy. The difficulty of individual privacy is compounded by the availability of auxiliary information, which renders straightforward approaches based on anonymization or data masking unsuitable.

Recent work in differential privacy [12] has radically changed the research landscape by providing a framework with a strong definition of privacy that addresses how much privacy loss an individual might incur by being in the database as compared to not being in the database, regardless of the auxiliary information that may be available to the database client.

*A preliminary version of parts of this work appears in Proceedings of the ICDM International Workshop on Privacy Aspects of Data Mining, 2009 [20]. Most of this work was carried out while the first author was at Rutgers University.

Using existing differential privacy results, it is possible to create differentially private high-level structures such as decision trees from data using multiple low-level differentially private queries [8], such as simple noisy count queries. Blum et al. [4] gave a private version of **ID3** using a predecessor privacy model to differential privacy. One can then release the resulting decision trees with a provable privacy guarantee. (Differential privacy definitions are given in Section 3.) However, this is not always useful in practice because the privacy guarantee degrades with the number of queries made.

In this paper, we consider the problem of constructing a differentially private decision tree classifier using the general method of creating high-level structures from low-level differentially private queries. We first present experimental evidence that creating a differentially private **ID3** tree using differentially private low-level queries does not simultaneously provide good privacy and good accuracy, particularly for small datasets. Specifically, we present results from the application of this algorithm to realistic data and observe that in order to obtain a reasonable privacy guarantee, the privacy parameter for each individual private sum query needs to be fairly small. Our experiments show that such a differentially private decision tree gives poor prediction for many databases.

Motivated by this poor performance, we instead take an alternative approach based on differentially private ensemble classifiers, which we argue are better suited to the approach of creating high-level differentially private algorithms from differentially private low-level queries. Using random decision trees [16], our algorithm produces classifiers that have good prediction accuracy without compromising privacy, even for small datasets. We note that, in contrast to **ID3** trees, random decision tree classifiers are not suitable for applications in which it is necessary to learn which combinations of attribute-values are most predictive of the class label, because random decision trees do not provide this information. However, they are suitable for applications that require only black-box prediction, similarly to other black-box learners such as neural networks for speech synthesis [33], industrial process control [22], and sales forecasting [36].

We also extend our random decision tree approach to the case where databases are periodically updated by appending new data or the data is partitioned among multiple parties. We present experimental results from the application of our differentially private random decision tree algorithm, and its extensions, to realistic data sets. Our experiments demonstrate that the approach provides good accuracy and privacy even for small data sets. For the case of private updates, the resulting random decision tree classifier handles data updates with only small reductions in accuracy while preserving the privacy guarantee.

To summarize our contributions, we address the problem of constructing differentially private classifiers using decision trees. Our main result is a differentially private classifier using random decision trees. We begin in Section 2 by putting our work into the context of related work. In Section 3, we describe existing differential privacy definitions and theorems that we make use of and describe the basic (non-private) **ID3** and random decision tree algorithms that form the basis of our differentially private algorithms. In Section 4, we consider the use of low-level differentially private sum queries to create differentially private **ID3** decision trees and motivate the necessity of considering a new approach. In Section 5, we show how to construct differentially private random decision trees, our main contribution. We also present extensions of the private random decision tree classifier in settings that involve data updates and settings in which databases are distributed among multiple parties. We present experimental analysis of our differentially private random decision tree algorithms in Section 6.

2 Related Work

Privacy has been a concern since the early days of database technology, and many privacy approaches to data analysis have been considered over the years, including output perturbation (e.g., [1]), data perturbation (e.g., [2]), secure multiparty computation (e.g., [19, 23]), and anonymization-based approaches (e.g., [31, 34, 24]).

Building on work of Dinur and Nissim [7] and Blum et al. [4], Dwork et al. [12] introduced differential privacy, a strong privacy definition that, unlike the methods mentioned earlier (cf. [18]), provides its privacy guarantees even in the presence of arbitrary auxiliary information. Specifically, a data access mechanism (or algorithm) satisfies differential privacy only if a change in any single record of a database has low impact on the output produced. Differential privacy has attracted a substantial amount of attention and further research (much of which is surveyed in [8, 9, 10, 11]).

Based on the composability of differential privacy [4, 12] (described in more detail in Section 3.1), a useful technique for obtaining differential privacy is to issue noisy answers to low-level queries, such as counting the number of rows in the database that satisfy a given predicate. These low-level queries can be used in the construction of differentially private data mining algorithms such as for decision trees. In the context of differentially private classifier learning, our results demonstrate that application of this technique to **ID3** tree construction [28, 4] results in a poor privacy/accuracy trade-off, while random decision trees, introduced by Fan et al. [16], are better suited to the technique. Fan et al. [16, 14, 15] show random decision trees are simple, efficient, accurate, and scale well to large databases.

There has been a substantial amount of work on differentially private classifiers, including differentially private ID3 [17] (discussed in more detail below). Kasiviswanathan et al. [21] show that any PAC-learning algorithm can be converted to a differentially private PAC-learning algorithm, although not necessarily one that runs in polynomial time. Blum et al [5] show that by ignoring computational constraints it is possible to release a non-interactive database useful in any concept class with a polynomial VC-dimension. Chaudhuri and Monteleoni [6] show how to obtain differentially private logistic regression using a method that incorporates perturbation directly into the objective function, rather than using the results of [12]. However their algorithm produces good accuracy only for moderate and large sized datasets. Rubenstein et al. [30] present efficient mechanisms for differentially private support vector machines.

Freidman and Schuster's work [17] is the most closely related to ours. In their work, they provide a method for achieving differential privacy in constructing an ID3 classifier that is more efficient than a direct application of [4, 12]. They demonstrate that their approach provides good results for large data sets and "relatively small" datasets. In contrast to their work, we demonstrate that our RDT approach works well even for substantially smaller datasets, even with much smaller values of ϵ (i.e., providing more privacy). Additionally, in their work, the height of the resulting decision trees is left as a user-specified parameter, while we provide an explicit formula to compute the height of resulting trees. A poor choice of this parameter could negatively affect privacy or utility, so providing such guidance could lead to better results in practice.

3 Background

Before we present our results, we provide background from the existing literature that we will make use of later.

3.1 Differential Privacy

Differential privacy is a notion of privacy for database access mechanisms (i.e., algorithms that return outputs computed based on access to a database). It captures a notion of individual privacy by assuring that the removal or addition of a single item (i.e., an individual's record) in a database does not have a substantial impact on the output produced by the mechanism. We now provide precise definitions.

Let $\mathcal{D}_1, \ldots, \mathcal{D}_k$ denote domains, each of which could be categorical or numeric. A database D consists of n rows, $\{x_1, x_2, \ldots, x_n\}$, where each $x_i \in \mathcal{D}_1 \times \ldots \times \mathcal{D}_k$. Two databases D_1 and D_2 *differ on at most one element* if one is a proper subset of the other and the larger database just contains one additional row.

Definition 1 ([12])**.** A randomized mechanism $\mathcal{M}$ satisfies *ϵ-differential privacy* if for all databases D_1 and D_2 differing on at most one element, and all $S \in \text{Range}(\mathcal{M})$,

$$Pr[\mathcal{M}(D_1) = S] \leq \exp(\epsilon) * Pr[\mathcal{M}(D_2) = S] \tag{1}$$

The probability is taken over the coin tosses of $\mathcal{M}$.

Smaller values of ϵ correspond to closer distributions, and therefore higher levels of privacy. Let f be a function on databases with range $\mathbb{R}^m$. A now-standard technique by which a mechanism $\mathcal{M}$ that computes a noisy version of f over a database D can satisfy *ϵ-differential privacy* is to add noise from a suitably chosen distribution to the output $f(D)$. The magnitude of the noise added to the output depends on how much change in f can be caused by a single change to the database, defined as follows:

Definition 2 ([12])**.** The *global sensitivity* of a function f is the smallest number $S(f)$ such that for all D_1 and D_2 which differ on at most one element,

$$\| f(D_1) - f(D_2) \|_1 \leq S(f) \tag{2}$$

Let $\text{Lap}(\lambda)$ denote the Laplace distribution with mean 0 and standard deviation $\sqrt{2}\lambda$.

Theorem 3 ([12])**.** *Let f be a function on databases with range $\mathbb{R}^m$. Then, the mechanism that outputs $f(D) + (Y_1, \ldots, Y_m)$, where Y_i are drawn i.i.d from $\text{Lap}(S(f)/\epsilon)$, satisfies ϵ-differential privacy.*

Using this method, smaller values of ϵ imply that more noise is added when the results are returned.

Theorem 4 (Composition Theorem [26])**.** *The sequential application of mechanisms $\mathcal{M}_i$, each giving ϵ_i-differential privacy, satisfies $\sum_i \epsilon_i$-differential privacy.*

Theorem 4 implies that differential privacy is robust under composition, but with an additive loss in privacy for each query made. While this provides an important tool for analyzing algorithms that make multiple queries to the data, a substantial practical problem can arise when high-level structures are created this way. Specifically, if an algorithm makes a large number m of such low-level queries, the privacy guarantee for the high-level structure is reduced by a factor of m. Because of this, for many databases and high-level structures, acceptable levels of privacy in the end result via these methods can only be obtained by sacrificing utility in the high-level structure.

Furthermore, as noted by Dwork [8], the technique of obtaining differential privacy by adding noise proportional to $S(f)/\epsilon$ frequently yields accurate results only when large

data sets are used. For example, consider a histogram query, for which $S(f) = 1$. If the privacy parameter ϵ is set to 0.01, the standard deviation for the Laplace noise added to each component of the output would be approximately 141, assuming only a single query is made. If q such queries are expected to be made, then to use the composition theorem (Theorem 4) to obtain the same privacy guarantee ϵ, the noise added to each query result should be approximately $141q$. If a data set contains only a few hundred or a few thousand rows, this amount of noise would completely overwhelm the underlying "signal." As we demonstrate in Sections 4 and 5, choosing data analysis methods that make fewer queries can have a substantial improvement on the resulting accuracy, particularly when the data set is small.

When queries can be made in parallel (i.e., the results from one query are not needed in order to determine which query to make next), the privacy guarantee can be made much stronger, in that there is no increase required in the differential privacy parameter ϵ.

Theorem 5 (Parallel Composition Theorem [25])**.** *Let mechanisms $\mathcal{M}_i$ each satisfy ϵ-differential privacy. Let D_i be arbitrary disjoint subsets of the database D. Then the composed mechanism consisting of each $\mathcal{M}_i$ applied to each D_i satisfies ϵ-differential privacy.*

We also note that Dwork et al. [13] recently gave a relaxed definition of differential privacy called (ϵ, δ)-differential privacy. In this definition, a mechanism M satisfies (ϵ, δ) differential privacy if $P(M(D_1) = S) \leq \exp(\epsilon)P(M(D_2) = S) + \delta$, for neighboring databases. With this relaxed definition of differential privacy, they present a new sequential composition theorem that does not require as large an increase in ϵ for sequential composition, but only applies when $\delta > 0$. We do not consider (ϵ, δ)-differential privacy in this paper; we believe that if we were to hold ϵ fixed and consider small values of δ, it could be used to provide modest accuracy improvements in both the **ID3** results we present and the random decision tree results we present.

Throughout the paper, we assume that the size n of the data set is public. In a situation where n is not public, a differentially private query can be made to obtain an estimate of the database size.

3.2 ID3 Decision Trees

The **ID3** decision tree learner [29] is a top-down recursive algorithm to create decision tree classifiers from labeled data. For reference, we present the original **ID3** algorithm in Algorithm 1. We will adapt it to differential privacy in Section 4.

3.3 Random Decision Trees

In most machine learning algorithms, the best approximation to the target function is assumed to be the "simplest" classifier that fits the given data, since more complex models tend to overfit the training data and generalize poorly. Ensemble methods such as Boosting and Bagging [32] combine multiple "base" classifiers to obtain new classifiers. It has been observed that ensemble methods can have significantly lower generalization error than any of the base classifiers on which they are based [32]. The base classifiers used in ensemble methods are usually "conventional" classifiers such as decision trees produced by C4.5, which are computationally expensive. The final step of combining these base classifiers can also be computationally intensive.

Algorithm 1 The **ID3** decision tree learning algorithm.

Algorithm ID3
Input: R , a set of independent attributes,
C, the class attribute, and
S, a training set.
Output: A decision tree

if S is empty **then**
return a leaf node with value Failure
end if
if S consists of records all with the same values for the class attribute **then**
return a leaf node with that value
end if
if R is empty **then**
return a leaf node with value the most frequent of the values
of the class attribute that are found in records of S
end if
D = Attribute in R with largest **InformationGain**(D, S, C)
Let $\{d_j | j = 1, 2, \ldots, m\}$ be the values of attribute D
Let $\{S_j | j = 1, 2, \ldots, m\}$ be the subsets of S consisting
respectively of records with value d_j for attribute D
return a tree with root labeled D and arcs labeled
$d_1, d_2, \ldots, d_m$ going respectively to the trees
ID3$(R - \{D\}, C, S_1)$, **ID3**$(R - \{D\}, C, S_2)$, ..., **ID3**$(R - \{D\}, C, S_m)$

Subroutine **InformationGain** (D, S, C)
return $(\mathbf{Entropy}(S, C) - \mathbf{AttributeEntropy}(D, S, C))$

Subroutine **Entropy** (S, C)
Let $\{c_j | j = 1, 2, \ldots, k\}$ be the values of the class attribute C
Let $\{S_j | j = 1, 2, \ldots, k\}$ be the subsets of S consisting
respectively of records with value c_j for attribute C
return $\left(-\sum_{j=1}^{k} \frac{|S_j|}{|S|} \log \frac{|S_j|}{|S|} \right)$

Subroutine **AttributeEntropy** (D, S, C)
Let $\{d_j | j = 1, 2, \ldots, m\}$ be the values of the attribute D
Let $\{S_j | j = 1, 2, \ldots, m\}$ be the subsets of S consisting
respectively of records with value d_j for attribute D
return $\left(\sum_{j=1}^{m} \frac{|S_j|}{|S|} \text{Entropy}(S_j) \right)$

However, Fan et al. [16] argue that neither of these steps (creating the classifiers and combining them) need be computationally burdensome to obtain classifiers with good performance. They present a fast and scalable ensemble method that performs better than the base classifiers, and frequently as well as the well-known ensemble classifiers. Counter-intuitively, their ensemble classifier uses base classifiers that are created from randomly chosen decision trees, in which attributes for decision tree nodes are chosen at random instead of using a carefully defined criterion, such as **ID3**'s information gain criterion. The structure of the decision tree (that is, which attributes are in which internal nodes of the decision tree) is determined even before any data is examined. Data is then examined to modify and label the random tree. The end result, based on creating an ensemble of random decision trees, is an algorithm that scales well for large databases.

The algorithm to build a single random decision tree is shown in Algorithm 2. The random decision tree classifier is an ensemble of such random decision trees. The algorithm first recursively creates the structure of the tree (**BuildTreeStructure**). Then it updates the statistics (**UpdateStatistics**, **AddInstance**) at the leaves by "filtering" each training instance through the tree. Each leaf node of the tree holds T counters, $\alpha[1], \ldots, \alpha[T]$, where T is the number of possible labels for training instances. After all the examples have been incorporated into the tree, the algorithm prunes away all internal and leaf nodes that did not encounter any of the examples in the training set. As presented, the algorithm works only for categorical attributes, but it can easily be extended to continuous-valued attributes by choosing random thresholds for each chosen attribute.

There are two important parameters to specify when using this ensemble method, namely (i) the height h of each random tree, and (ii) the number N of base classifiers. Using simple combinatorial reasoning, Fan et al. [16] suggest that a good choice for the height is $h = m/2$, where m denotes the number of attributes. They also find that a value for N as small as 10 gives good results.

Once the classifier is computed, when a test instance needs to be classified, the posterior probability is output as the weighted sum of the probability outputs from the individual trees, as shown in Algorithm 3.

The advantage of the random decision tree algorithm is its training efficiency as well as its minimal memory requirements. The algorithm uses only one pass over the data to create each random decision tree. For our purposes, an additional advantage of random decision trees is that the algorithm to construct each tree makes a relatively small number of "low-level" queries to the data, and most of the structure of a random decision tree is created without examining the data. As we show in Sections 5 and 6, this renders the algorithm suitable to the method of differentially private construction of high-level data structures via differentially private low-level queries.

4 Differentially Private ID3 Trees from Private Sum Queries

Much of the work in the differential privacy framework addresses the problem of issuing noisy answers to low-level queries, such as counting the number of rows in the database that satisfy a given predicate. These low-level queries can be used in the construction of differentially private data mining algorithms such as for decision trees. However, a substantial practical problem can arise when higher level structures are created using low-level queries. By Theorem 4, if an algorithm makes q such queries each with a privacy parameter ϵ, the overall privacy guarantee of the structure is $\epsilon' = q\epsilon$. In practice, for ϵ' to be reasonable, one must choose ϵ to be fairly small, which increases the amount of noise

Algorithm 2 Random Decision Tree (RDT) Algorithm

Algorithm Random Decision Tree (**RDT**)
Input: D, the training set, and
X, the set of attributes.
Output: A random decision tree R

```
R = BuildTreeStructure(X)
UpdateStatistics(R, D)
Prune subtrees with zero counts
return R

Subroutine BuildTreeStructure(X)
  if X = ∅ then
    return a leaf node
  else
    Randomly choose an attribute F as testing attribute
    Create an internal node r with F as the attribute
    Assume F has m valid values
    for i = 1 to m do
      c_i = BuildTreeStructure(X − {F})
      Add c_i as a child of r
    end for
  end if
  return r

Subroutine UpdateStatistics(r, D)
  for each x in D do
    AddInstance(r, x)
  end for

Subroutine AddInstance(r, x)
  if r is not a leaf node then
    Let F be the attribute in r
    Let c represent the child of r that corresponds to the value of F in x
    AddInstance(c, x)
  else
    /* r is a leaf node */
    Let t be the label of x
    Let α[t] = # of t-labeled rows that reach r
    α[t] ← α[t] + 1
  end if
```

Algorithm 3 Computing the probability for each possible label for a test instance

Algorithm Classify
Input: $\{R_1, \ldots, R_N\}$, an ensemble of random decision trees, and
x, the row to be classified.
Output: Probabilities for all possible labels

For a tree R_i, let ℓ_i be the leaf node reached by x
Let $\alpha_i[t]$ represent the count for label t in ℓ_i

$$P(t|x) = \sum_{i=1}^{N} \alpha_i[t] / \left(\sum_{\tau} \sum_{i=1}^{N} \alpha_i[\tau] \right)$$

return probabilities for all t

added to each low-level query. This can have a significant negative impact on the utility of the high-level structure the user wants to compute.

In this section, we study the resulting utility and privacy that occurs when differentially private construction of **ID3** decision trees is carried out via the method of using low-level noisy queries to construct the high-level structures. Specifically, we obtain a differentially private **ID3** algorithm by replacing accesses to the training set S in the the standard algorithm (shown in Algorithm 1) with equivalent queries to a differentially private interactive mechanism. In this case, the "high-level structure" is the resulting decision tree, while the "low-level queries" are noisy sum queries. Blum et al. [4] gave a private version of **ID3** using a predecessor privacy model to differential privacy. Though we follow the same general approach, our work does not modify their algorithm to achieve differential privacy. Instead, we show below the various modifications we made to the original **ID3** algorithm to achieve differential privacy. While [4] proved theoretical guarantees about the accuracy of the algorithm, in this paper we show empirically that [4] has low accuracy on small and moderate sized real datasets.

The affected statements of **ID3** and how they are modified are as follows:

1. The conditions of the first two **If** statements in function **ID3** access the training set S. These can be evaluated by using two queries, each of global sensitivity 1:

   (a) one query that asks for the number of rows in S, and

   (b) one query that asks for the count in S of each value of the class attribute C.

   Note that although it would be possible to use the sum of the values obtained for the second query as the size of S (thus avoiding the use of the first query), this sum would have higher variance than the result for a direct query for the size of S.

2. The return statement in the third **If** of function **ID3** refers to the most frequently occurring class value in S. This can be computed using answers to the queries of 1(b) above.

3. The entropy computation in the return statement of the function Entropy needs to know the size of S and sizes of its subsets S_j, for $1 \leq j \leq k$. These can be computed using differentially private queries of global sensitivity 1.

Data set	# rows	# queries	Accuracy original **ID3**	Accuracy Most Freq Class	Accuracy Private **ID3**
Nursery	12960	36	98.19%	33%	46%
Cong. Votes	435	136	94.48%	61%	40%
Mushroom	8124	253	100%	52%	56%

Table 1: Implementing a privacy-preserving version of **ID3** using low-level differentially private queries produces classifiers with poor accuracy on many widely used data sets.

4. The weighted entropy computation in the return statement of the function Attribute-Entropy needs to know the size of S and the sizes of the subsets S_j, for $1 \leq j \leq m$. This can be obtained used a query of global sensitivity 1.

Note that each of the subsets of the training set S used in the algorithm can be specified by some set of attribute-value pairs.

We can use the Parallel Composition Theorem (Theorem 5) at each level of the **ID3** tree, as the data set at any node is disjoint from the data sets at other nodes in the same level of the tree. However, we need to use the sequential Composition Theorem (Theorem 4), for the queries at different levels of the tree. If each of the queries to S at different levels of the tree provides ϵ differential privacy and there are k such queries, then this approach provides ϵk-differential privacy. Unfortunately, the number k of queries made of S via this approach can be rather large. In order for the privacy guarantee ϵk in the final tree to be acceptably small, the privacy parameter ϵ for each noisy sum query may need to be quite small. However, this requires a large amount of noise to be added to each query result, effectively destroying the correlation between the attributes in the database. As we observe experimentally, the resulting tree can have very poor accuracy.

We implemented the private **ID3** algorithm and ran our experiments on three datasets from the UCI Machine Learning Repository [3]—namely the **Nursery** dataset, the **Congressional Voting Records** dataset and the **Mushroom** dataset. In Table 1, we show the accuracy results—i.e., the percentage of test instances that were classified correctly. As described above, in order to determine the differential privacy guarantee of the algorithm, we use both the parallel composition theorem (for the full set of queries at a given level of the tree) and the sequential Composition Theorem (for the set queries for multiple levels). Specifically, we set the overall differential privacy parameter of the overall **ID3** tree algorithm to $\epsilon' = 0.5$. For each dataset, we then use the number of attributes as the the maximum number of sequential queries the algorithm might need to make for that dataset (because the actual number of queries is not known in advance). Based on that, we set the privacy parameter ϵ for each vector of noisy sum queries at a single level of the tree by ϵ'/q, where q is the maximum number of queries made. For example, in the case of the **Mushroom** dataset, the value of ϵ for each noisy query is approximately 0.002. As can be seen from Table 1, on average, the differentially private **ID3** algorithm is not much more accurate than simply predicting the most frequently occurring class. This poor performance motivates the need for an alternate approach to differentially private classifier construction, which we describe in Section 5.

5 Differentially Private Random Decision Trees

As discussed in Section 3.3, random decision trees are suited for adaptation to differential privacy. Considering the construction of a single random decision tree, because the attributes in the tree nodes are chosen completely at random (and even before the data is examined), these choices yield no information about the data. In contrast, in conventional decision trees such as **ID3**, the presence of an attribute in the root indicates its relative predictive capability. In a random decision tree, with one exception we discuss shortly, the only part of the resulting classifier that depends on the input database is the counters in the leaves of the trees. This makes the random decision tree algorithm a good candidate to consider for creating a differentially private mechanism based on differentially private low-level queries to determine the value of the counters. However, as just noted, there is one exception to the structure not depending on the data. In addition to the value of the counters, the given structure of the tree depends on the values of the data due to the pruning step.

In this section, we consider a modified form of the algorithm that satisfies ϵ-differential privacy. As compared to the original algorithm, it has two differences: it eliminates the pruning step and replaces the count queries for the counters with differentially private count queries. We also consider extensions to batch data processing and distributed databases. We provide experimental results of these algorithms in Section 6.

5.1 Private Random Decision Tree Algorithm

We now describe in more detail our algorithm for creating a differentially private random decision tree, which is a modification of the original random decision tree algorithm (shown in Algorithm 2). We begin by eliminating the pruning step that removes "empty" tree nodes. Eliminating the pruning step has the effect that the tree structures produced by the resulting algorithm do not depend at all on the data. This also results in the algorithm creating trees in which all of the leaves are at the same level; the leaf nodes of a random decision tree, then, effectively form a *leaf vector* V of $M \cdot T$ integers, where M is the number of leaf nodes and T is the number of possible labels for instances in the training data. This vector of "counts" is updated by the **UpdateStatistics** function. Effectively, releasing a random decision tree amounts to releasing (i) the structure of the tree and (ii) this vector of counts. As we show below in Theorem 6, the leaf vector has a global sensitivity of 1, as defined in Section 3.1. It therefore follows from Theorem 3 that adding $\mathrm{Lap}(1/\epsilon)$ noise to each component of V and releasing the resulting noisy vector satisfies ϵ-differential privacy. The resulting algorithm, shown in Algorithm 4, produces a single differentially private random decision tree. The data owner releases an ensemble of differentially private random decision trees obtained by repeated application of this algorithm. In other words, using Theorem 3 and Algorithm 4, we can build an ϵ-differentially private ensemble of N random decision trees by adding $\mathrm{Lap}(N/\epsilon)$ noise to each leaf vector. It takes $O(n \log n)$ time to construct a single differentially private random decision tree, where n is the size of the dataset.

The height we use for a database depends on:

1. the average number of values taken by the attributes of the data set (denoted by b),
2. the number of rows in the database (denoted by n).

Clearly, b is close to the average branching factor of a random decision tree. Among other considerations, we want to ensure that even if the rows are evenly distributed among the

Algorithm 4 Privacy-Preserving RDT Algorithm

Algorithm Private-RDT
Input: D, the training set, and
X, the set of attributes.
Output: A random decision tree R

R = **BuildTreeStructure**(X)
UpdateStatistics(R, D)
Add $\mathrm{Lap}(1/\epsilon)$ to each component of the leaf vector.
return R

leaves of a random decision tree, the leaves will not all be very sparse, because sparse leaf counts are more susceptible to noise added by the **Private-RDT** algorithm. Hence, it is not advisable to choose a height h such that $b^h \gg n$. If the rows of the database tend to clump together into a small number of leaves, that could be acceptable, at least if the test set is likely to be similarly distributed. However, if the height is too small there would be too few clumps, and the leaves would lose the power of discrimination. A compromise is to choose a value of h close to $\log_b n$. We suggest the height $h = \min(\lfloor k/2 \rfloor, (\lfloor \log_b n \rfloor - 1))$ where k is the number of attributes: this is the value we use in our experiments in Section 6. (As described previously, if the size n of the data set is not known, it can be approximated via a differentially private noisy query.) As a result, the size of the leaf vector is linear in the size of the data set.

Theorem 6. *The* **Private-RDT** *algorithm is ϵ-differentially private.*

Proof. Let A denote the **Private-RDT** algorithm. For a tree R, we denote the noisy leaf vector of R by $\lambda(R)$.

Consider a fixed random decision tree structure into which no examples have yet been incorporated. Let D_1 and D_2 be two databases differing in at most one element that generate leaf vectors V_1 and V_2 respectively on the tree (before noise is added). The global sensitivity for the leaf vector of that tree is 1, because V_1 and V_2 must differ in exactly one component by a value of 1.

We need to show that for any tree R, the ratio $\frac{P(A(D_1)=R)}{P(A(D_2)=R)}$ is bounded from above by e^ϵ. Here $A(D)$ denotes the output produced by the randomized algorithm A on input D. Because the structure of the random decision tree is generated even before the data is examined, it suffices for us to show that $\frac{P(\lambda(A(D_1))=V)}{P(\lambda(A(D_2))=V)}$ is bounded by e^ϵ, for any leaf vector V. This immediately follows from Theorem 3, taken with the facts that the sensitivity of the noiseless leaf vectors is 1 and the noise added is $\mathrm{Lap}(1/\epsilon)$. □

In order to determine an ensemble of trees to release in a non-private setting, the data owner can determine and release an ensemble with maximal accuracy. However, this potentially violates differential privacy, since it depends on the data. In the differential privacy setting, therefore, the **Private-RDT** algorithm should be used to produce a random ensemble of random decision trees, which are then released.

Algorithm 5 Updating Random Decision Trees

Algorithm Update Random Decision Tree
Input: D_2, representing new data
r_1, the private random decision tree built using old data D_1.
Output: Updated tree r_2 for $D_1 \cup D_2$

Create a clone r_1' of r_1 and clear out the leaf vector.
Use **UpdateStatistics** to insert the rows of D_2 into r_1'.
Add Laplacian noise to the leaf vector of r_1'.
Add the leaf vector of r_1' to the leaf vector of r_1 and release updated random decision tree r_2.
return r_2

5.2 Updating Random Decision Trees

The **Private-RDT** algorithm assumes that all data is available at once. However, in many important real-world applications, data arrives in batches. In the case where data is periodically appended to an existing database, a classifier built on the combined data is likely to be preferred to a classifier built on the new data alone, while rebuilding a classifier from scratch can be a time consuming proposition for large datasets.

Incremental learning seeks to efficiently update learned classifiers as new batches of data arrive. Incremental learning can be challenging even when privacy is not an issue. In the context of differential privacy, an additional challenge arises. Specifically, if the privacy guarantee of the result of multiple increments is to be proved via use of the composition theorem (Theorem 4), then each update to a classifier results in lowering the privacy guarantee provided. In this section, we show how private random decision trees can handle data updates is a way that does not suffer from this problem. The tradeoff is a potential reduction in prediction accuracy, as compared with building a random decision tree directly from the combined data.

Let D_1 and D_2, respectively, represent the old and the new data. Let r_1 be a private random decision tree built using D_1. Here $D_1 \cup D_2$ denotes the entire set of data that have arrived in both batches. The procedure for handling data updates is described in Algorithm 5. This algorithm satisfies differential privacy with the privacy parameter ϵ since the leaf vector of r_1' is based on D_2 alone, and not on the data D_1. We present experiments in Section 6 to show that, for the data sets we consider, the accuracy of r_1 is not substantially reduced after a small number of updates. After a few iterations, the accuracy of the random decision tree ensemble will be reduced. At that stage, one could build a new ensemble from scratch and then return to more efficient updates.

5.3 Private Random Decision Trees for Distributed Databases

While much of knowledge discovery happens within an organization, it is quite common to use data from multiple sources in order to yield more precise or useful knowledge. Data may be distributed among multiple databases owned by different entities, and pooling them at a centralized location may not be possible or desirable due to limitations in computational and communication resources as well as to privacy constraints. Distributed data mining provides algorithms to perform data mining in a distributed setting without pooling the data into one location [27]. In this section, we consider the problem of constructing

decision tree classifiers in a differentially private manner when the database is horizontally or vertically partitioned between multiple parties.

Private random decision trees on horizontally partitioned data. A (virtual) database table is *horizontally partitioned* among a number of parties if each party has a database table with the same set of attributes. The virtual table is the union of all these tables. We assume that the parties have no rows in common and that they have agreed on ϵ and on the order in which the data will be processed.

In this case, the parties can use essentially the same algorithm as described for updating random decision trees in Section 5.2. Each partition of the database is treated as a batch of data to be used in an update. The tree structure of each tree in the ensemble does not change after the first party, so it need not be announced again each time. Instead, once the first party publishes the ensemble based on its own data, each party simply publishes the noisy counts of its data according to the leaf vector of each tree in the ensemble. Depending on the preference of the parties and the relative cost of computation vs. communication, either one party can then process the noisy counts in sequence as in Algorithm 5, or each party can do so locally.

Because each party constructs differentially private classifiers from datasets that have no rows in common, the union of the classifiers from all the parties together is also differentially private. This approach has the following advantages:

- Because each party constructs and publishes a classifier from only its own portion of data, the communication overhead is no more than if each party were to publish its own portion of the data.
- A client who wishes to construct a private classifier based on data from only some of the parties can do a local computation using the noisy counts from only those parties.

Private random decision trees for vertically partitioned databases. A virtual database table is *vertically partitioned* among a number of parties if each party has data for some fixed set of attributes about individuals that are known to all the parties. Each party here holds a different set of attributes, except for a common identifier attribute which relates a row in one database table with a row in another. The virtual database table is the "join" of all the distributed tables. We consider the case of two parties in this section; the extension to multiple parties is straightforward.

Let D_1 and D_2 denote vertical partitions of database D, where the first partition is owned by, say, Alice and the second by, say, Bob. Let X_1 denote the set of attributes for database D_1 and X_2 denote the attributes for database D_2. We assume that both parties have the same class attribute. We also assume that the new instance to be classified contains the entire set of attributes of the database D; this assumption can be relaxed with a small loss in utility. The construction is described in Algorithm 6.

Because each party constructs differentially private classifiers from datasets that have no attributes in common, the union of the classifiers from all the parties together is also differentially private.

6 Experimental Results

In this section, we present our experimental results showing that the private random decision tree algorithm achieves good utility in terms of prediction accuracy. We ran three

Algorithm 6 Private RDT for vertically partitioned data

Algorithm Update Random Decision Tree

Input: D_1 and X_1 denote Alice's partition of the database and attributes respectively
D_2 and X_2 denote Bob's partition of the database and attributes respectively

Output: Alice outputs ensemble A
Bob outputs ensemble B

Alice computes an ensemble A of random decision trees using algorithm **Private-RDT** with inputs D_1 and X_1. Alice releases the ensemble A.
Bob computes an ensemble B of random decision trees using algorithm **Private-RDT** with inputs D_2 and X_2. Bob releases the ensemble B.
To classify a new instance, run the algorithm **Classify** on the union of the two ensembles A and B.
return Alice returns A and Bob returns B

sets of experiments. First, we ran experiments to measure the accuracy of private random decision tree ensembles for various values of the privacy parameter ϵ. Second, we ran experiments to observe the change in the accuracy of random decision tree ensembles when there are batch updates to the data, which also apply to the case of horizontally partitioned data. Third, we ran experiments to measure the utility of the algorithm for vertically partitioned data. All our implementations are in Java using the Weka machine learning framework [35].

6.1 Accuracy of Private Random Decision Tree Ensembles

The experiments were run on data sets available from the UCI Machine Learning Repository [3]. We restricted ourselves to data sets with only categorical attributes. Extending the implementation to continuous attributes should only take a small amount of additional effort.

Experimental Setup In our experiments, we considered a variety of values of ϵ (specifically, $\epsilon \in \{5, 4, 3, 2, 1, 0.75, 0.5, 0.25, 0.1, 0.01\}$) in order to test the effect of ϵ on the resulting prediction accuracy. The particular choice of ϵ that is appropriate for a given setting is specific to the application and must be decided by the data owner based on utility and privacy requirements and tradeoffs.

We performed our experiments on three data sets from the UCI Machine Learning Repository, namely the **Nursery**, **Mushroom** and **Congressional Voting Records** data sets, which range from moderately sized to quite small, as it is precisely the application for small data sets that motivated us to seek an alternative approach. See Table 2 for data characteristics. In the **Mushroom** database, we removed the attribute that has missing entries. In the **Congressional Voting Records** database, we replaced each missing vote with the majority vote for that bill.

We must also specify the number of trees in the ensemble. In the non-private version of random decision trees, increasing the number of trees in the ensemble increases the accuracy of predictions. Our experiments indicate that, for our data sets, using as few as five decision trees in an ensemble on average produces acceptable accuracy. However, the variance in accuracy between runs is lower when more trees are used. An ensemble with

Data set	# attribs	# rows	# Class labels
Nursery	8	12960	3
Mushroom	22	8124	2
Cong. Votes	16	435	2

Table 2: Experimental Data Characteristics

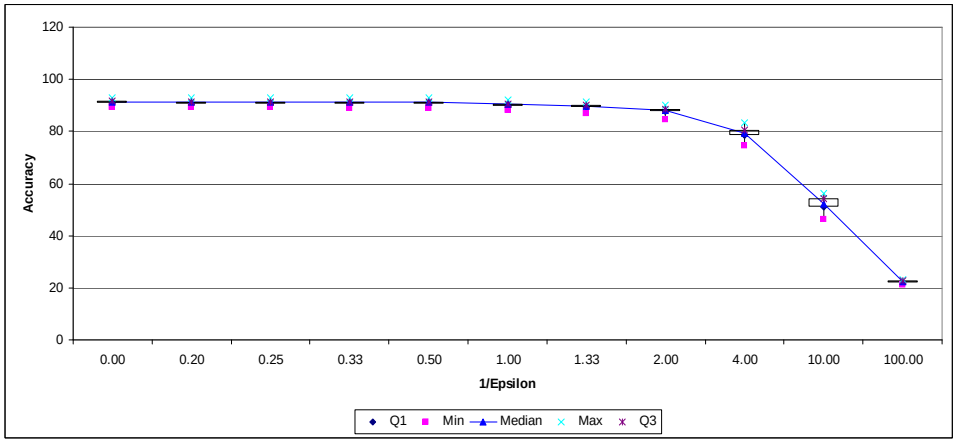
10 or more trees has better accuracy on average, with lower variance. On the other hand, since one count query is required per random decision tree, creating q trees implies that the per-query privacy parameter needs to be set to ϵ/q. This increases the amount of noise added per query, which negatively impacts prediction accuracy. The **Congressional Voting Records** data set has only 435 rows. Increasing the number of trees beyond five yielded poor results for that data set. For the other data sets, we set the number of trees to 10. Finally, our initial experiments indicated that setting the height of the generated random trees to $k/2$, where k is the number of attributes, does not produce optimal results. As discussed in Section 5.1, we set the height of the tree for a database to be $h = \min(\lfloor k/2 \rfloor, (\lfloor \log_b n \rfloor - 1))$ where b denotes the average number of values taken by the attributes of the data set and n is the number of rows in the database.

To get a sense of the variation of the prediction accuracy, we present in each case summary statistics over 10 runs of private ensembles of 10 random decision trees (with the exception of the **Congressional Voting Records** data set), for each value of $\epsilon \in \{5, 4, 3, 2, 1, 0.75, 0.5, 0.25, 0.1, 0.01\}$. For the sake of consistency, we used the same set of ensembles of tree structures for every value of ϵ. Prediction accuracy is based on the stratified cross-validation technique available in Weka. The accuracy for each ensemble and each value of ϵ is computed by averaging over 10 runs. We re-sampled the Laplace distribution for each run to add noise to the leaf vector. For purposes of comparison, we also show the prediction accuracy of a non-private implementation of random decision trees. In this case, we set ϵ to ∞. We did not prune this tree, as the performance of the tree is high even without pruning. (See Figures 1(a), 2(a), and 3(a).) We did not compare our results with the standard random decision tree ensemble algorithm with pruning.

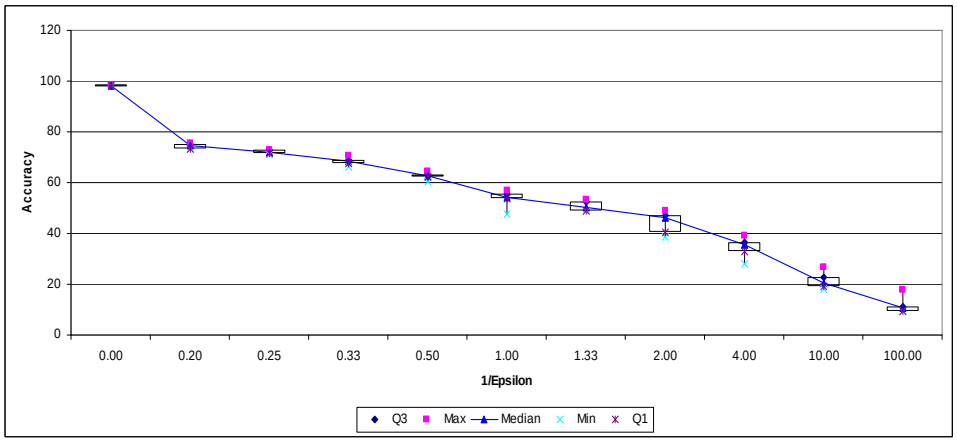
Results We use box plots to graphically display a five-number summary of the accuracy of **Private-RDT** on each data set and each value of ϵ, consisting of the observed minimum and maximum accuracies, the lower and upper quartiles, and the median. We present in Figures 1(a), 2(a), and 3(a) the box plots of the results of our experiments on the **Nursery**, **Mushroom** and **Congressional Voting Records** data sets. In these and other figures, we represent the privacy parameter as $1/\epsilon$ so that the values on the X-axis increase from left to right.

In our experiments of the **Private-RDT** algorithm, we observe that lower values of ϵ generally result in lower average accuracy of predictions. This is as expected, because the amount of noise added is inversely proportional to ϵ. The drop in average accuracy is gradual and not precipitous, but by $\epsilon = 0.01$, the amount of added noise overwhelms **Private-RDT**. With $\epsilon = 5$, the reduction in accuracy as compared to the non-private version of the algorithm (i.e., $\epsilon = \infty$), though noticeable, is not substantial.

For comparison, we also present in Figures 1(b), 2(b), and 3(b) the accuracy of the differentially private **ID3** algorithm for each data set and each of the values of ϵ. In comparing the accuracy of private **ID3** to **Private-RDT**, we can see that the difference in accuracy for the



(a)



(b)

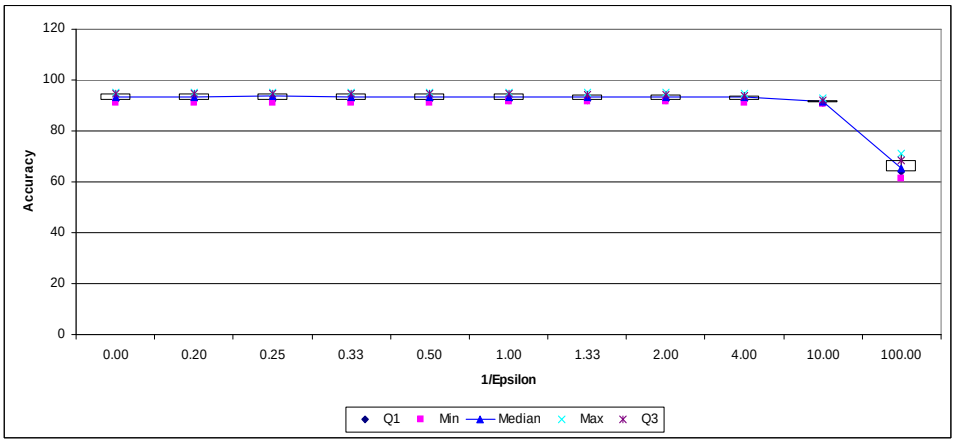
Figure 1: (a) Accuracy on the Nursery data set from the UCI Repository using **Private-RDT**. Displayed are the accuracy values for $1/\epsilon \in \{0, 0.2, 0.25, 0.33, 0.5, 1, 1.33, 2, 4, 10, 100\}$. (b) Accuracy on the Nursery data set from the UCI Repository for differentially private **ID3**.

Mushroom data set is lower than the difference in accuracy for the smaller **Congressional Voting Records** data set. In Figures 2(b) and 3(b), for example at $\epsilon = 1$, the difference in accuracy for the mushroom data set is around 20% whereas the difference in accuracy for the congressional voting data set is around 40%. This provides evidence that the difference in performance between **ID3** and **RDT** shrinks as the size of the data set increases.

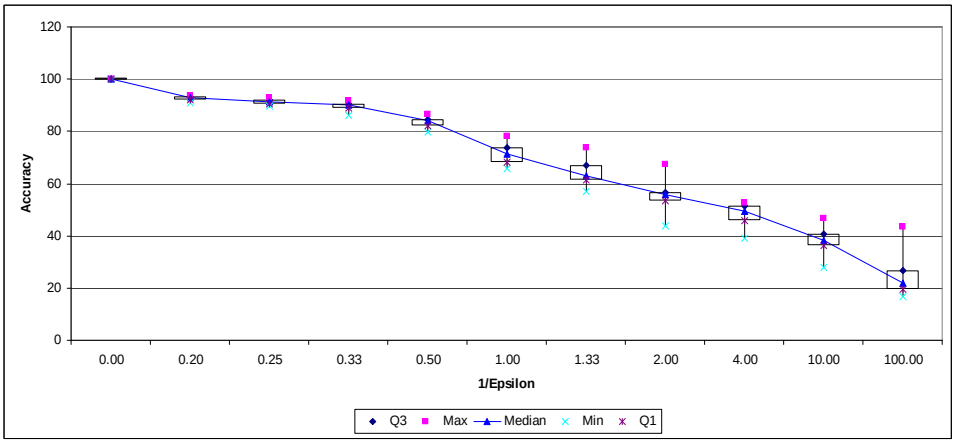
Overall, these figures show that the private random decision tree ensemble algorithm has good accuracy, even for relatively small data sets.

6.2 Updating Random Decision Trees

We ran the algorithm presented in Section 5.2 on each of the large data sets from the earlier experiments. We split each data set into m equal parts where $m = 2, 4, 6, 8, 10$. We used the privacy parameter $\epsilon = 0.5$. We present the results obtained on **Nursery** and **Mushroom**

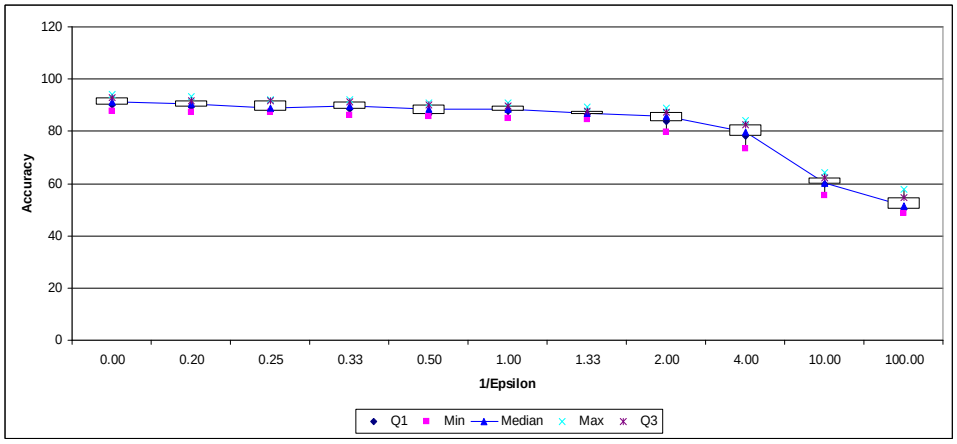


(a)

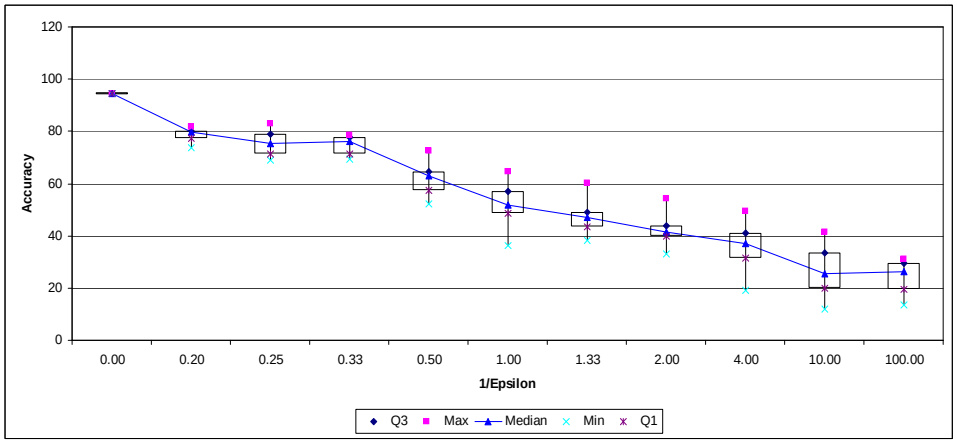


(b)

Figure 2: (a) Accuracy on the Mushroom data set from the UCI Repository using **Private-RDT**. (b) Accuracy on the Mushroom data set from the UCI Repository for differentially private **ID3**.

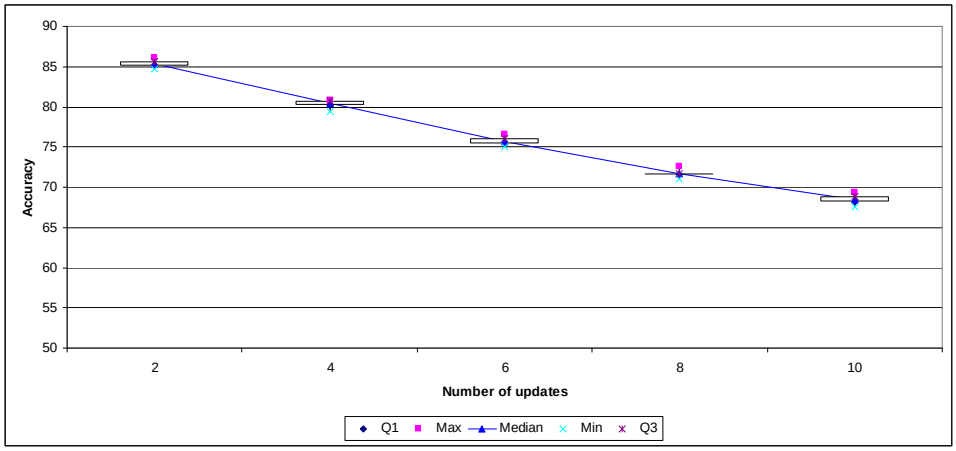


(a)

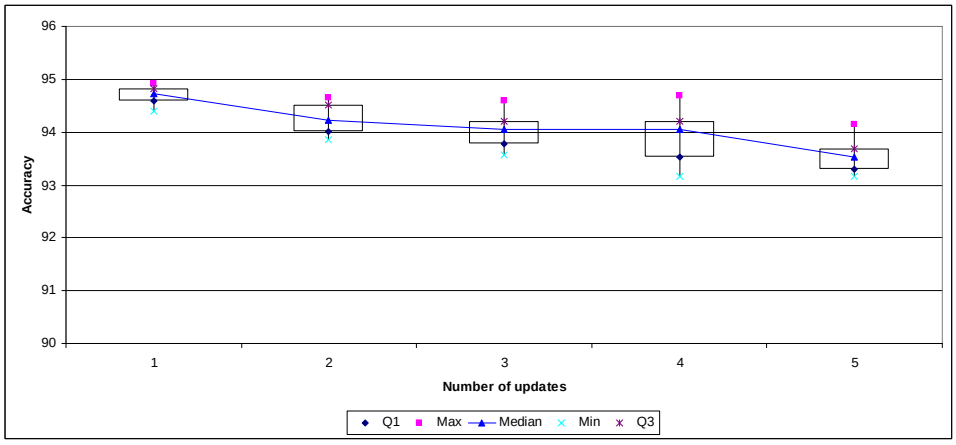


(b)

Figure 3: (a) Accuracy on the Congressional Voting Records data set from the UCI Repository using **Private-RDT**. (b) Accuracy on the Congressional Voting Records data set from the UCI Repository for differentially private **ID3**.



(a)



(b)

Figure 4: (a) Performance of the update algorithm on the Nursery data set. (b) Performance of the update algorithm on the Mushroom data set.

in Figure 4. The accuracy of each ensemble is computed by averaging over 10 runs. Our experimental results indicate that for a small number of updates, our algorithm produces random decision trees with good prediction accuracy. Our experiments on the private random decision tree update algorithm also show that there is a gradual reduction in accuracy when the number of updates increases.

These results also apply to creating random decision trees from horizontally partitioned data. Specifically, results presented for m updates apply to the case of $m + 1$ parties.

6.3 Private Random Decision Trees on Vertically Partitioned Data

To test the utility of the ensembles produced for vertically partitioned data, we vertically partitioned our datasets by randomly dividing the set of attributes into two disjoint sets. We ran the **Private-RDT** algorithm on each partition and computed the union of the ensembles output for each partition. The accuracy of each ensemble, for each ϵ, is computed

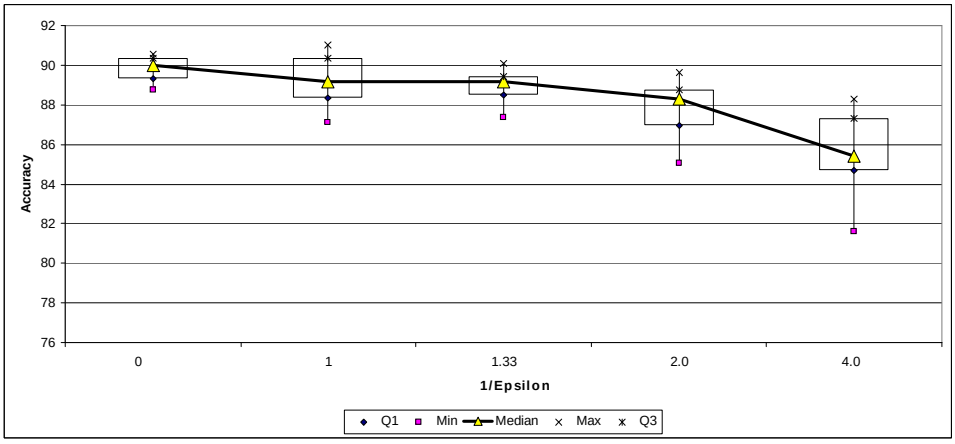


Figure 5: Performance of the vertical partitioned database algorithm on randomly partitioned **Congressional Voting Records** dataset.

by averaging over 10 runs. Figure 5 presents the accuracy for the **Congressional Voting Records** database.

We note that in cases where the variance is high (as happens in the **Congressional Voting Records** data and can be seen in comparing Figures 3(a) and 5), the additional noise added in processing partitioned data can actually result in higher accuracy than when processing the whole dataset at once. However, this should be viewed as an unpredictable side effect of the high variance, rather than a method for systematically obtaining higher accuracy.

7 Conclusions

We presented a differentially private decision tree classifier using the random decision tree approach. Our use of random decision trees for differential privacy is motivated by a study of differentially private **ID3** trees. We experimentally showed that our approach yields good prediction accuracy even when the size of the database is small. This is possible because the classifier is built from only a small number of queries to the database (as compared to a straightforward differentially private adaptation of **ID3** tree classifiers).

We extended our approach to constructing decision tree classifiers in a differentially private way when new data is periodically appended to an existing database and when data is horizontally or vertically partitioned between multiple parties. In all of these cases, we experimentally demonstrated that our differentially private random decision tree algorithm produces classifiers with good utility even for small databases.

Acknowledgments

We would like to thank the anonymous referees for their valuable comments. Geetha Jagannathan and Rebecca N. Wright were partially supported by NSF award CNS-0822269.

References

[1] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: A comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. volume 29, pages 439–450, New York, NY, USA, 2000. ACM.

[3] A. Asuncion and D. Newman. UCI Machine Learning Repository, 2007.

[4] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *PODS '05: Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138, 2005.

[5] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC '08: Proceedings of the 40th annual ACM Symposium on Theory of computing*, pages 609–618, 2008.

[6] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS 08': Proc. Neural Information Processing Systems Foundations*, pages 289–296, 2008.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS '03: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, 2003.

[8] C. Dwork. Differential privacy: A survey of results. In *TAMC : Theory and Applications of Models of Computation, 5th International Conference*, pages 1–19, 2008.

[9] C. Dwork. The differential privacy frontier (extended abstract). In *TCC*, pages 496–502, 2009.

[10] C. Dwork. Differential privacy in new settings. In *SODA*, pages 174–183, 2010.

[11] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[13] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010.

[14] W. Fan. On the optimality of probability estimation by random decision trees. In *AAAI'04: Proceedings of the 19th National Conference on Artificial Intelligence*, pages 336–341. AAAI Press / The MIT Press, 2004.

[15] W. Fan, E. Greengrass, J. McCloskey, P. S. Yu, and K. Drummey. Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 154–161, Washington, DC, USA, 2005. IEEE Computer Society.

[16] W. Fan, H. Wang, P. Yu, and S. Ma. Is random model better? On its accuracy and efficiency. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 51, 2003.

[17] A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD*, pages 493–502, 2010.

[18] S. Ganta, S. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[19] O. Goldreich. *Foundations of Cryptography, Vol II*. Cambridge University Press, 2004.

[20] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A practical differentially private random decision tree classifier. In *ICDMW '09: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, pages 114–121, Washington, DC, USA, 2009. IEEE Computer Society.

[21] S. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *FOCS '08: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer*

Science, pages 531–540, Oct. 2008.

[22] B. Lennox, G. A. Montague, A. M. Frith, C. Gent, and V. Bevan. Industrial application of neural networks – an investigation. *Journal of Process Control*, 11(5):497 – 507, 2001.

[23] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[24] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[25] F. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[26] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2007.

[27] B. Park and H. Kargupta. Distributed data mining: Algorithms, systems, and applications. *In The Handbook of Data Mining, edited by N. Ye*, pages 341–358, 2003.

[28] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[29] J. R. Quinlan. *Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems*, pages 349–361. Morgan Kaufmann Publishers Inc., USA, 1993.

[30] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *CoRR*, abs/0911.5708, 2009.

[31] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.

[32] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[33] M. Scordilis and J. Gowdy. Speech synthesis of phonemic triplets through a neural network-controlled formant synthesizer. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, page 1007, July 1991.

[34] L. Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[35] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[36] D. Yip, E. Hines, and W. Yu. Application of artificial neural networks in sales forecasting. In *Neural Networks,1997., International Conference on*, volume 4, pages 2121 –2124, June 1997.