A Practical Differentially Private Random Decision Tree Classifier

Geetha Jagannathan
Department of Computer Science
Rutgers University
New Brunswick, NJ, USA
geetha@cs.rutgers.edu

Krishnan Pillaipakkamnatt
Department of Computer Science
Hofstra University
Hempstead, NY, USA
csckzp@hofstra.edu

Rebecca N. Wright
Department of Computer Science
Rutgers University
New Brunswick, NJ, USA
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 construct privacy-preserving *ID3* decision trees using differentially private sum queries. Our experiments show that for many data sets a reasonable privacy guarantee can only be obtained via this method at a steep cost of accuracy in predictions.**

We then present a differentially private decision tree ensemble algorithm using the random decision tree approach. We demonstrate experimentally that our approach yields good prediction accuracy even when the size of the datasets is small. We also present a differentially private algorithm for the situation in which new data is periodically appended to an existing database. Our experiments show that our differentially private random decision tree classifier handles data updates in a way that maintains the same level of privacy guarantee.

***Index Terms*—Differential Privacy; Classifiers; Ensembles**

I. Introduction

The problem of securing databases from the disclosure of confidential information either directly from the collected data or from the knowledge mined from the collected data has been the subject of research for a very long time [1]. Many models such as the *perturbation* model [2], the k-*anonymity* model [3], [4], and the *secure multiparty computation* [5] (SMC) model have been proposed for data privacy. Until recently, techniques based on perturbation or on k-anonymity and it variants lacked formal proofs of privacy or were applicable only in limited settings. In the SMC setting all parties receive only the final output without any party learning any individual entry of any of the other parties that is not revealed by the output. But although the computation does not reveal the inputs, SMC does not address how much the outputs might reveal about the inputs.

Recent work in private data analysis by Dwork et al. [6] has radically changed the research landscape by defining a model with a strong definition of privacy that addresses how much privacy loss an individual might incur by being in the database. In the most common setting under this new framework of *differential privacy*, the data owner makes the data available through a statistical database on which only aggregate queries are permitted. The goal is to answer queries while preserving the privacy of every individual in the database, irrespective of any auxiliary information that may be available to the database client.

Using existing differential privacy results, it is possible to create high-level structures such as decision trees using multiple low-level differentially private queries [7], [8]. However, a substantial practical problem arises 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.

In this paper, we consider the problem of constructing a differentially private decision tree classifier. We first present experimental evidence that creating a differentially private *ID3* tree using low-level differentially private sum queries does not simultaneously provide good privacy and good accuracy. 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 noisy 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. Our main result is an algorithm for differentially private ensemble classifiers. Using random decision trees [9] our algorithm produces classifiers that have good prediction accuracy without compromising privacy, even for small datasets. In contrast to the privacy-preserving decision tree construction methods presented in [10], [2], these decision trees provide provable privacy guarantees about any individual's contribution to the final result.

We present results from the application of our differentially private random decision tree algorithm to both synthetic and realistic data. Our experiments demonstrate that our approach yields good prediction accuracy even when the size of the datasets are relatively small. We also extend our results to the case where databases that are periodically updated by appending new data and show experimentally that the resulting differentially private random decision tree classifier handles data updates in a way that has small reductions in accuracy while preserving the privacy guarantee.

We begin in Section II by describing the differential pri-

Appears in *Proceedings of the ICDM International Workshop on the Privacy Aspects of Data Mining*, 2009

vacy model in more detail and summarizing relevant existing results. In Section III, we consider the use of low-level differentially private sum queries to create differentially private decision trees and motivate the necessity of considering a new approach. In Section IV, we present an overview of random decision trees, which form the basis of our new approach. In Section V, we show how to construct differentially private random decision trees, our main contribution. We also present a procedure to update private random decision trees as data updates are made. In Section VI, we present experimental analysis of our differentially private random decision tree algorithms.

II. Differential Privacy

The ϵ-*differential privacy* model introduced by Dwork et al. [6] assures that the removal or addition of a single item in a database does not have a substantial impact on the output produced by a private database access mechanism.

Let $\mathcal{D}_1, \ldots, \mathcal{D}_k$ denote domains, each of which could be categorical or numeric. Our 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 in at most one element* if one is a proper subset of the other and the larger database just contains one additional row [8].

Definition 1 ([8]): 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 $S \in$ Range($\mathcal{M}$),

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

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

Note that smaller values of ϵ correspond to higher levels of privacy. Let f be a function on databases with range $\mathbb{R}^m$. A standard technique by which a mechanism $\mathcal{M}$ that computes a noisy version of f over a database D can achieve 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 the sensitivity of f, defined as follows:

Definition 2 ([8]): 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,

$$\| f(D_1) - f(D_2) \|_1 \leq S(f) \qquad (2)$$

Let $Lap(\lambda)$ denote the Laplacian distribution with mean 0 and standard deviation $\sqrt{2}\lambda$. The following theorems are proven in [6], [11].

Theorem 1 ([6]): Let f be a function on databases with range R^m. Then, the mechanism that outputs $f(D) + (Y_1, \ldots, Y_m)$, where Y_i are drawn i.i.d from $Lap(S(f)/\epsilon)$, achieves ϵ-differential privacy.

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

Theorem 2 ([11]): The sequential application of mechanisms $\mathcal{M}_i$, each giving ϵ_i-differential privacy, gives $\sum_i \epsilon_i$-differential privacy.

Theorem 2 implies that differential privacy is robust under composition, but with an additive loss of privacy for each query made.

Data set	# rows	# queries	Accuracy orig. *ID3*	Accuracy Private
Nursery	12960	176	98.19%	18.73%
Cong. Votes	435	97	94.48%	2.53%
Mushroom	8124	267	100%	1.48%

TABLE I
IMPLEMENTING A PRIVACY-PRESERVING VERSION OF *ID3* USING LOW-LEVEL DIFFERENTIALLY PRIVATE QUERIES PRODUCES VERY POOR ACCURACY ON MANY WIDELY USED DATA SETS.

III. ID3 Trees for Private Sum Queries

Most of the work in the differential privacy framework addresses the problem of issuing noisy answers to low-level queries. These low-level queries, such as count queries, can be used in the construction of differentially private data mining algorithms such as for decision trees [7]. However, a substantial practical problem arises when higher level structures are created using low-level queries as described in [7]. By Theorem 2, 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 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 paper, we study the resulting utility and privacy that occurs when low-level noisy queries are used to construct high-level structures. We first examine the construction of differentially private *ID3* decision trees using noisy sum queries as in [7]. In order to obtain an acceptable privacy guarantee in the final tree, the privacy parameter needs to be fairly small in each noisy sum query used in the computation of information gain. 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 has poor accuracy.

We implemented the private *ID3* algorithm and ran our experiments on three datasets from the UCI Machine Learning Repository [12]—namely the **Nursery** dataset, the **Congressional Voting Records** dataset and the **Mushroom** dataset. Accuracy results are given in Table I, which is the percentage of test instances that were classified correctly. (In these cases, the vast majority of the test instances were left unclassified by the trees created, though some instances were also misclassified.) In all of these datasets, the overall privacy parameter of the *ID3* tree is set to $\epsilon' = 1$. The privacy parameter ϵ for each noisy sum query is given by ϵ'/q, where q is the number of queries made. For example, in the case of the **Mushroom** dataset, the value of ϵ for each noisy query is approximately 0.0037. As can be seen from Table I, for all three datasets the resulting private *ID3* algorithm has poor utility.

To overcome this poor performance, we take a different approach. We construct a differentially private ensemble classifier using the approach of random decision trees [9]. Instead of adding noise to each of the queries made, this approach computes the entire decision tree and adds noise at the end

to the leaves. We show that this gives good utility without compromising privacy. We describe this method in detail in Section V after first introducing random decision trees.

IV. Random Decision Trees: An Overview

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 [13] 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 [13]. 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.

However, Fan et al. [9] 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. Counterintuitively, 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. 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 Figure 1. The algorithm as presented works only for categorical attributes, though it can easily be extended to continuous-valued attributes by choosing random thresholds for the chosen attribute. The algorithm recursively creates the structure of the tree (**BuildTreeStructure**), and then 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. The running time of the algorithm is linear in the size of the database.

The random decision tree classifier is an ensemble of such random decision trees. When a test instance needs to be classified, the posterior probability is output as the weighted sum of the the probability outputs from the individual trees (see Figure 2).

There are two important parameters to be chosen when using this ensemble method, namely (i) the height h of each random tree, and (ii) the number N of base classifiers. Using

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 $X = \Phi$ **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 $\alpha[t]$ = # of t-labeled rows that reach r
$\alpha[t] \leftarrow \alpha[t] + 1$
end if

Fig. 1. Random Decision Tree Algorithm

simple combinatorial reasoning, Fan et al. [9] 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 low as 10 gives good results.

The advantage of creating a random tree is its training efficiency as well as its minimal memory requirements. The algorithm uses only one pass over the data to create a random decision tree. In a series of papers, Fan et al. [14], [15] show that the random decision tree algorithm is simple, efficient and accurate. They surmise that the reason for the superiority

Algorithm Classify

Input: $\{R_1, \ldots, R_N\}$, an ensemble of RDTs, 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

Fig. 2. Computing the probability for each possible label for a test instance

of their ensemble method is that it optimally approximates for each example its true probability of being a member of a given class—that is, the random decision tree ensembles form efficient implementations of Bayes Optimal Classifiers.

V. Differentially Private Random Decision Trees

As was discussed in Section IV, the structure of a random decision tree is created without examining the data. Only the counters in the leaves of the tree depend on the input database. This makes the random decision tree algorithm a potentially good candidate for a differentially private mechanism. However, the given randomized decision tree algorithm does not satisfy the requirements of differential privacy because of the way the pruning step is carried out, as demonstrated by the following counterexample. Consider the databases D_1 and D_2 (that differ in at most one element) in Figure 3, shown along with an initially-empty randomly generated tree structure. The trees R_1 and R_2 that result from the *deterministic* steps of **UpdateStatistics** and pruning are shown in Figure 4. The probability of the tree R_1 being generated from database D_2 is zero.

In this section, we present a modified form of the algorithm which does satisfy ϵ-differential privacy.

A. Private Random Decision Tree Algorithm

We now present an algorithm for creating a differentially private random decision tree. It is a modification of the original random decision tree algorithm. We begin by eliminating the pruning step that removes "empty" tree nodes. The algorithm thus creates a tree 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 MT 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 (i) releasing the structure of the tree followed by (ii) this vector of counts. Since the attributes in the tree nodes are chosen completely at random (even before the data is examined), the structure contains no information about the data. (This is unlike in conventional decision trees such as *ID3*, where the presence of an attribute in the root indicates its relative predictive capability.) As we show below in Theorem 3, the leaf vector has a global sensitivity of 1. (Recall that global sensitivity is defined in Section II.) It follows from Theorem 1 that adding Lap($\frac{1}{\epsilon}$) noise to each component of V and releasing the resulting noisy vector satisfies ϵ-differential privacy. Note that the values in the leaf vector will no longer be integer counts. The resulting algorithm, shown in Figure 5, produces a single differentially private random decision tree. The data owner releases an ensemble of differentally private random decision trees obtained by repeated application of this algorithm.

Theorem 3: The **Private-RDT** algorithm is ϵ-differentially private.

Proof (sketch): 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}. Since 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 1, taken with the facts that the sensitivity of the noiseless leaf vectors is 1 and the noise added is Lap($1/\epsilon$). ∎

B. Updating Random Decision Trees

The **Private-RDT** algorithm assumes that all data is available at one go. In the real world data usually arrives in batches. We consider a situation 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. This kind of *incremental learning* is frequently a challenging problem, even when privacy is not an issue. The solution of rebuilding a classifier from scratch can be a time consuming proposition for large datasets.

In the context of differential privacy, a second problem arises. In order to make use of the composition theorem, the updated classifier must provide a lower privacy guarantee than the initial classifier does. Subsequent updates will worsen the privacy guarantee even further. In this section, we show how private random decision trees can handle data updates is a way that does not lower the privacy guarantee. The tradeoff is a marginal reduction in prediction accuracy when 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 instances of the database. Let r_1 be a private random decision tree built using D_1. We present here some possible approaches to update r_1 to include D_2, and associated problems:

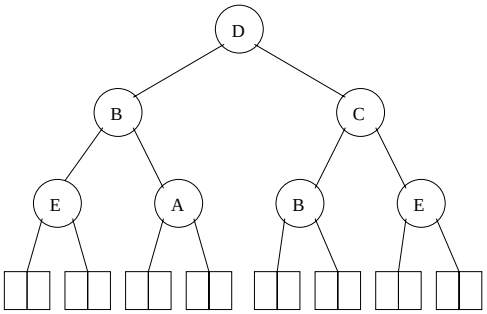
- One possible option is to incorporate the instances of D_2 into r_1 and then re-release this updated r_1. However, this

A	B	C	D	E	class
1	0	1	0	0	T
0	0	1	0	0	F
1	0	0	0	0	T
0	1	0	0	1	F
1	1	1	0	0	T
1	1	1	0	1	F
1	0	1	0	1	T
0	0	1	1	0	T

D_1

A	B	C	D	E	class
1	0	1	0	0	T
0	0	1	0	0	F
1	0	0	0	0	T
0	1	0	0	1	F
1	1	1	0	0	T
1	1	1	0	1	F
1	0	1	0	1	T

D_2



Tree structure

Fig. 3. RDT Algorithm is not private: Example databases that differ in at most one element and randomly generated tree structure

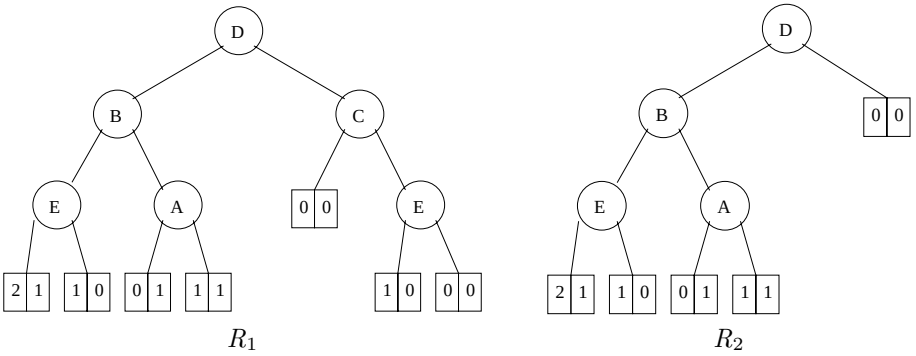


Fig. 4. RDT Algorithm is not private: Final trees for D_1 and D_2

can result in a breach of privacy.

- A second option is to clear out the leaf vector for r_1, and then run **UpdateStatistics** with $D_1 \cup D_2$. Then add Lap$(1/\epsilon)$ noise to the leaf vector. By the composition theorem, the resulting private random decision tree will have a privacy guarantee of 2ϵ. In general, k such updates will raise the privacy parameter to $k\epsilon$.
- A third option is to build a new random decision tree from scratch using $D_1 \cup D_2$, and then add noise to the leaf vector. As before, the private guarantee erodes to 2ϵ. For example, if the **Private-RDT** algorithm was run twice with a parameter of $\epsilon = 0.5$, once with the old data alone, and once with the union of the old and new data, the final privacy guarantee would be $\epsilon = 1$.

We can avoid these privacy related issues using the following procedure:

1) Create a clone r'_1 of r_1 and clear out the leaf vector.
2) Use **UpdateStatistics** to insert the rows of D_2 into r'_1.
3) Add Laplacian noise to the leaf vector of r'_1.
4) Add the leaf vector of r'_1 to the leaf vector of r_1 and release this updated random decision tree.

Since the leaf vector of r'_1 is based on D_2 alone, and not on any instance of D_1, the updated random decision tree continues to satisfy the privacy parameter of ϵ. We present experiments in Section VI to show that the accuracy of r_1 is not substantially reduced after a single update. After a few iterations, the accuracy of the 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.

VI. Experiments

In this section, we present our experimental results that show that the private random decision tree algorithm achieves good utility in terms of prediction accuracy. We ran two 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. We implemented our algorithms in Java using the Weka machine learning framework [16].

A. Accuracy of Private Random Decision Tree Ensembles

The experiments were run on data sets available from the UCI Machine Learning Repository [12] and from synthetic data that we generated. We restricted ourselves to data sets with only categorical attributes, although extending the implementation to continuous attributes should only take a small amount of additional effort.

Experimental Setup: As noted by Dwork [8], the technique of obtaining differential privacy by adding noise proportional to $\frac{S(f)}{\epsilon}$ 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 Laplacian 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 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." Since our largest realistic data set contains no more than 13,000 rows, and our largest synthetic data set contains only 16,000 rows, we used larger values of ϵ, namely 0.5, 0.75 and 1. As ϵ decreases, the amount of noise to be added must increase, resulting in a decrease in prediction accuracy. The particular choice of ϵ is specific to the application and will be decided by the data owner based on the utility/privacy requirements.

Another important parameter to consider is 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 produces acceptable accuracy on average, although the variance in accuracy between runs is higher than when using more trees. An ensemble with 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 $\frac{\epsilon}{q}$. This increases the amount of noise added per query, which negatively impacts prediction accuracy. Again, the size of the data set matters. The **Congressional Voting Records** data set has only 435 rows. Increasing the number of trees beyond five yielded poor results. For the other data sets, we set the number of trees to 10. Finally, our experiments indicate that setting the height of the generated random trees to $k/2$, where k is the number of attributes, is not always optimal.

We performed our experiments on three data sets from the UCI Machine Learning Repository, namely the **Nursery**, **Mushroom** and **Congressional Voting Records** data sets and on three synthetic data sets. Each synthetic data set was generated from a handmade Boolean decision tree. One of these trees is presented in Figure 12. We added noise to these data sets by flipping each class label with a probability of 0.05 to make these synthetic data sets more realistic. (This has no bearing on privacy.) See Table II for data characteristics.

To get a sense of the variation of the prediction accuracies, we present in each case the average prediction accuracy and the maximum accuracy 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 $\epsilon \in \{0.5, 0.75, 1\}$. For comparison, we also show the average prediction accuracy of a non-private implementation of random decision trees (we set ϵ to ∞ and the tree is not pruned). Prediction accuracy is based on the stratified cross-validation technique available in Weka. We removed from the **Mushroom** database 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 did not compare our results with the standard random decision tree

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 Lap($\frac{1}{\epsilon}$) to each component of the leaf vector.
return R

Fig. 5. Privacy-Preserving RDT Algorithm

Data set	# attribs	# rows	# Class labels
Nursery	8	12960	3
Mushroom	22	8124	2
Cong. Votes	16	435	2
Set 1	10	16000	2
Set 2	6	14000	2
Set 3	7	14000	2

TABLE II
EXPERIMENTAL DATA CHARACTERISTICS

ensemble algorithm with pruning. In a non-private setting, the data owner can release an ensemble with maximal accuracy. But such a release would leak information. In our setting, the data owner releases a random ensemble of random decision trees.

Results: We present in Figures 6, 7 and 8 the results of our experiments on the **Nursery**, **Mushroom** and **Congressional Voting Records** data sets. Figures 9, 10 and 11 present the results of our experiments on three synthetic data sets. In almost all experiments on the **Private-RDT** algorithm, whether on realistic data from the UCI Repository or on synthetic data, we see that lower values of ϵ generally result in lowered average accuracy of predictions. This is as expected, since the amount of noise added is inversely proportional to ϵ. However, the drop in average accuracy is gradual and not precipitous.

For the three data sets from the UCI Repository, the reduction in accuracy from $\epsilon = \infty$ to $\epsilon = 1$, though noticeable, is not substantial. The **Mushroom** data set appears to be the least affected by the addition of noise, while the **Congressional Voting Records** data set appears to be the most sensitive. The latter can be partly explained by the smaller size of the data set.

For the three synthetic data sets, just as is the case of the data sets from the UCI repository, there is a gradual reduction in accuracy with the decrease in ϵ. However, in two of these synthetic data sets, the reduction in average accuracy from the noiseless runs ($\epsilon = \infty$) to the ones with noise added is a little more steep. Also, there appears to be a larger variance in the maximum accuracy compared to the UCI data sets.

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

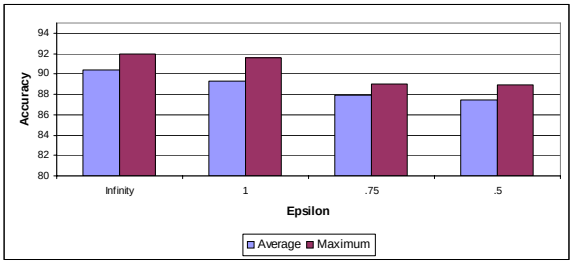


Fig. 6. Accuracy on the Nursery data set from the UCI Repository. Displayed are the average and maximum accuracy for $\epsilon \in \{0.5, 0.75, 1, \infty\}$.

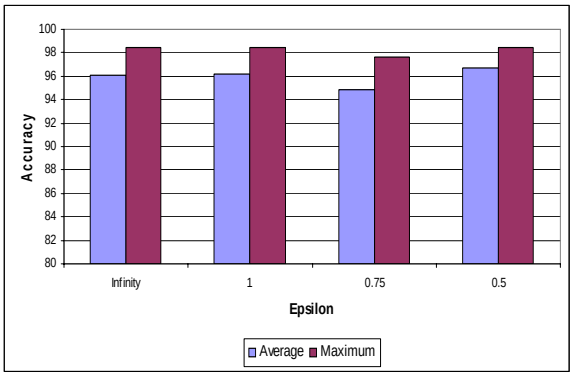


Fig. 7. Accuracy on the Mushroom data set from the UCI Repository.

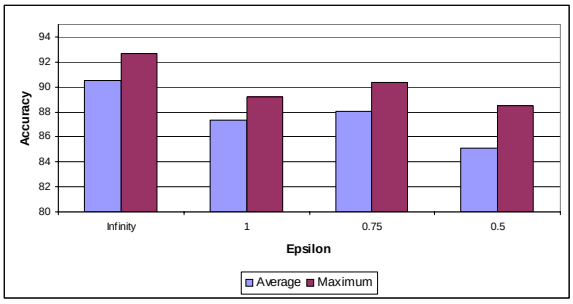


Fig. 8. Accuracy on the Congressional Voting Records data set from the UCI Repository.

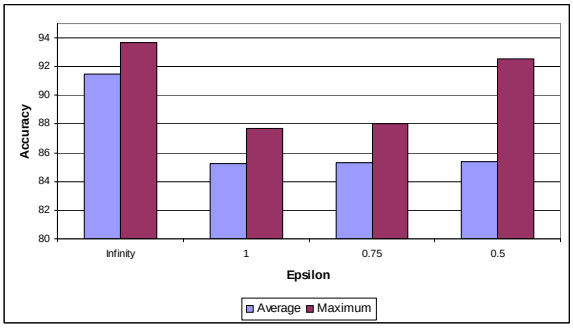


Fig. 9. Accuracy on synthetic data set 1. The decision tree for this data set (not shown) has 10 attributes.

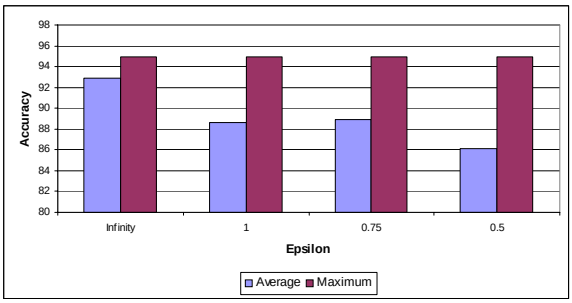


Fig. 10. Accuracy on synthetic data set 2. See Figure 12 for underlying decision tree.

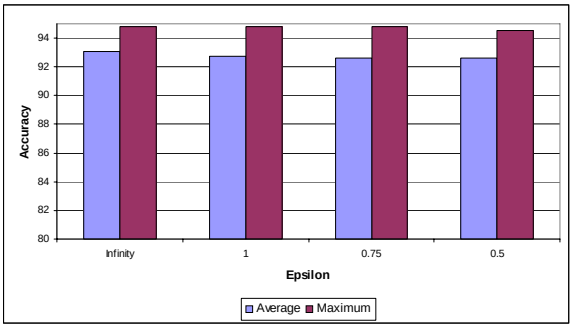


Fig. 11. Accuracy on synthetic data set 3. The decision tree for this data set (not shown) has 7 attributes.

B. Updating Random Decision Trees

We ran the algorithm presented in Section V-B on each of the large data sets from the earlier experiments. We split each data set into two equal halves. The first half was used as D_1 and the second as D_2. We present the results obtained on **Nursery**, **Mushroom** and on the synthetic data set 1 in Figures 13, 14 and 15 respectively. In each case we compare the accuracy of the ensemble produced by the update algorithm to the ensemble produced by **Private-RDT** on the union of old and new data.

Our experiments on the private random decision tree update algorithm also show that there is a gradual reduction in accuracy with decreasing values of ϵ. More importantly, the difference in accuracy between this algorithm and the **Private-RDT** algorithm run on the union of old and new data is low.

The value of ϵ for the **Private-RDT** algorithm run on the union of old and new data would be twice the value displayed in Figures 13–15 because of Theorem 2 (assuming that the curator had previously released an ensemble based on the old data alone). For example, if the **Private-RDT** algorithm was run twice with $\epsilon = 0.5$, once with the old data alone, and once with the union of the old and new data, the final privacy guarantee would be $\epsilon = 1$. Therefore, to get a fair comparison, one should compare the results for $\epsilon = 1$ for the private random decision tree update algorithm against $\epsilon = 0.5$ for the **Private-RDT** algorithm on the unioned data. The results from the experiments run on the private random decision tree updating algorithm indicate that the algorithm substantially preserves accuracy with the released trees retaining their

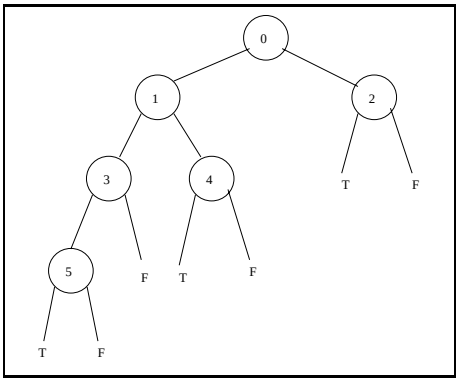


Fig. 12. The decision tree used to generate synthetic data set 2.

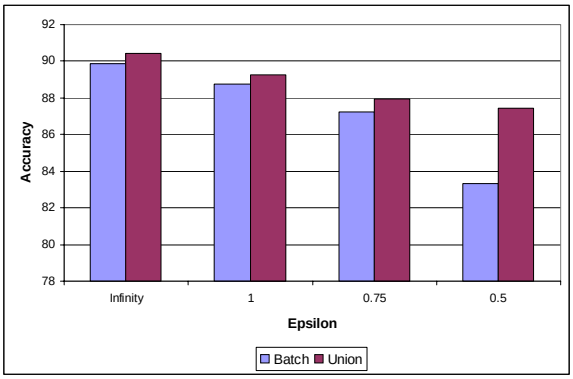


Fig. 13. Performance of the update algorithm on the Nursery data set.

original (pre-update) levels of privacy.

ACKNOWLEDGMENTS

We wish to thank the anonymous referees for their valuable comments. Geetha Jagannathan and Rebecca N. Wright were funded by NSF grant CCR-0331584.

REFERENCES

[1] N. R. Adam and J. C. Worthmann, "Security-control methods for statistical databases: A comparative study," *ACM Comput. Surv.*, vol. 21, no. 4, pp. 515–556, 1989.

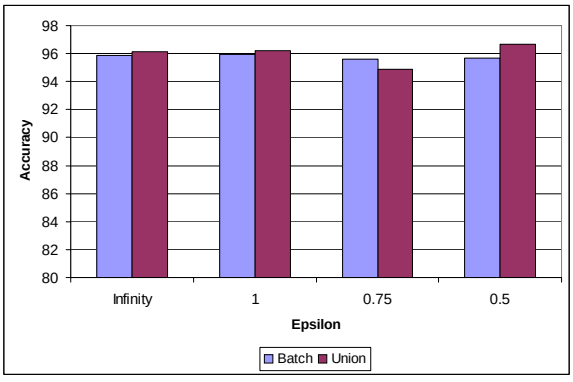


Fig. 14. Performance of the update algorithm on the Mushroom data set.

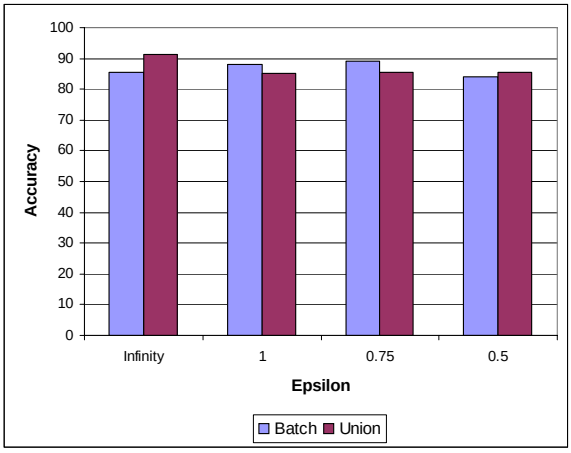


Fig. 15. Performance of the update algorithm on synthetic data set 1.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM SIGMOD*, May 2000, pp. 439–450.
[3] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Trans. Knowl and Data Engng.*, vol. 13, pp. 1010–1027, 2001.
[4] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.
[5] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. New York, NY, USA: Cambridge University Press, 2004.
[6] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC '06*, 2006, pp. 265–284.
[7] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: The sulq framework," in *PODS '05*, 2005, pp. 128–138.
[8] C. Dwork, "Differential privacy: A survey of results," in *TAMC*, 2008, pp. 1–19.
[9] W. Fan, H. Wang, P. Yu, and S. Ma, "Is random model better? on its accuracy and efficiency," in *ICDM '03*, 2003, p. 51.
[10] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *J. Cryptol.*, vol. 15, no. 3, pp. 177–206, 2002.
[11] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *FOCS '07*, 2007, pp. 94–103.
[12] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html
[13] 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*, vol. 26, no. 5, pp. 1651–1686, 1998.
[14] W. Fan, "On the optimality of probability estimation by random decision trees," in *AAAI*, 2004, pp. 336–341.
[15] W. Fan, E. Greengrass, J. McCloskey, P. Yu, and K. Drummey, "Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches," pp. 154–161, 2005.
[16] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.