2013 IEEE 13th International Conference on Data Mining Workshops

A Semi-Supervised Learning Approach To Differential Privacy

Geetha Jagannathan
Department of Computer Science
George Washington University
Washington, D.C.
geetha@geeiscool.com

Claire Monteleoni
Department of Computer Science
George Washington University
Washington, D.C.
cmontel@gwu.edu

Krishnan Pillaipakkamnatt
Department of Computer Science
Hofstra University
Hempstead, N.Y.
csckzp@hofstra.edu

***Abstract*—Motivated by the semi-supervised model in the data mining literature, we propose a model for differentially-private learning in which private data is augmented by public data to achieve better accuracy. Our main result is a differentially private classifier with significantly improved accuracy compared to previous work. We experimentally demonstrate that such a classifier produces good prediction accuracies even in those situations where the amount of private data is fairly limited. This expands the range of useful applications of differential privacy since typical results in the differential privacy model require large private data sets to obtain good accuracy.**

I. Introduction

Databases that contain sensitive information about individuals need to be safeguarded from malicious access that can compromise privacy. Typical attacks assumed an adversary who can use public information to gain insight into a target individual's information stored in the database. Differential privacy [8] offers protection against such attacks irrespective of any auxiliary information that may be available to an adversary trying to breach privacy. In this work, we consider auxiliary information (i.e. public data) from a different perpective. We ask the question: Can non-private data be used to "boost" the accuracy of differentially-private algorithms? We answer this question in the affirmative. Motivated by the well known semi-supervised model in machine learning literature, we propose a learning model that uses both private and non-private data. We design a data mining algorithm where non-private data is used in conjunction with a small amount of private data to increase the accuracy of differentially-private learners.

Semi-Supervised Learning: In the semi-supervised model [5] of machine learning, a learner has access to both labeled and unlabeled data. Usually, the learner has only a small amount of labeled data available, while the amount of unlabeled data is much larger. This imbalance in the amounts of data is due to the fact that labeling training instances can be an expensive, time-consuming and difficult process that requires human domain experts. Unlabeled data is usually much easier to acquire. In the model of semi-supervised classification the goal is to use both labeled and unlabeled data to create a classifier that is better than one created using the labeled data alone. It has been shown in the machine learning literature that in many cases, unlabeled data used in conjunction with a small amount of labeled data can lead to an increase in the accuracy of classifiers produced by the learning algorithms.

Private and Non-private Data: Prior work in the differential privacy model has assumed that data being analyzed is entirely private. However, this is not always a realistic presumption. Consider the two following scenarios:

1) An organization surveys young customers at a clothing store about their purchasing habits. Individuals for whom privacy is a substantial concern may insist that the organization use their data only in ways that does not reveal anything about them. On the other hand some individuals may be willing to give up their data privacy in exchange for some compensation. The survey organization would then have a database that contains both private as well as non-private data. It would be reasonable to assume that the presence of non-private data can improve the quality of the data analysis they would like to perform.
2) A confidential survey about residents in a community may include information about whether or not each respondent in the database has an annual salary of at least $100,000. A public database (such as a voter registration database) about the same community will not include such confidential information. An organization that wants to mine the private database would perhaps benefit from the public database, even though it is missing an important confidential attribute.

Our Contributions: The main goal of this paper is to provide a learning model that addresses the above mentioned scenarios. We consider the problem of improving the accuracy of a differentially private classifier using non-private data when only a small amount of private data is available. One naive approach is to consider all the non-private data as additional private data and construct a differentially private classifier on the combined data. In doing so, the classifier pays the cost of privacy even when it not required to do so for non-private data. This can result in lowered accuracy. In contrast, our technique initially constructs a differentially private classifier from the private data and then uses the non-

978-0-7695-5109-8/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDMW.2013.131

private data as a post-processing step to "boost" the accuracy of the initial classifier.

Our differentially private classifier is a non-trivial extension of the differentially private random decision tree classifier [13]. However, random decision trees were not designed to deal with unlabeled instances. We extend the random decision tree idea to exploit the availability of (non-private) unlabeled data. We use the unlabeled instances in two ways. First, we use them to control the partitioning of the instance space so that denser regions of the space are partitioned more finely than sparse regions. Second, we use the unlabeled examples to "propagate" the labels from the labeled instances to larger regions of the instance space. Together, these techniques boost the utility of the differentially private classifier without lowering privacy. We experimentally demonstrate that our claims holds even on small and moderate sized datasets. Our goal is to boost the accuracy of the classifier, in contrast to the differentially private boosting algorithm [9] that boosts the accuracy of a class of real valued queries.

We begin in Section II by describing related work. In Section III, we describe our model for differentially-private learning. In Section IV we describe our new differentially private classifier. In Section V, we present the results of our experimental analysis of our classifier on real world data.

II. Related Work

The model of differential privacy introduced by Dwork et al. [8] offers strong privacy guarantees for the analysis of a private data set. Much of this work is surveyed in [7]. Prior work in differential privacy also includes privacy-preserving learning algorithms in the SuLQ framework [2]. Blum et al. [3] gave a computationally inefficient way of constructing a synthetic database useful in any concept class with polynomial VC-dimension. Recently, Mohammed et al. [18] gave a differentially-private anonymization algorithm based on the generalization technique. Generalization replaces a specific value with a more general value to make the information less precise. Jagannathan et al. [13] presented a differentially private classifier based on random decision trees. They heuristically showed that their algorithm achieves good accuracy even for small datasets. Our work in this paper can be considered to be a non-trivial extension of that result to our new learning model.

Friedman and Schuster [11] presented a differentially private ID3 algorithm that gives better accuracies than the straightforward construction of a differentially private ID3. However when tested on real world data sets their algorithm provided good accuracies only on large or moderate-sized datasets. Lee and Clifton [16] gave a new formulation called differential identifiability. This framework provides the strong privacy guarantees of differential privacy but also lets the privacy makers set the parameters based on the established privacy concept of individual identifiability. Karwa and Slavkovic [15] presented a differentially private algorithm that releases a graphical degree partition of a graph. Recently, Jain and Thakurta [14] addressed the problem of differentially private learning where the training features are accessed only through a kernel function. In [12], the authors preprocessed the counts by grouping and sampling them. This process reduced the Laplacian noise added to the counts.

III. Our Model

Let D_{priv} denote the database owned by the curator for which privacy needs to be preserved (the *private subset*). We also assume that the curator has access to a dataset D_{npriv} for which privacy need not be preserved (the *non-private subset*). We use the term "non-private" instead of "public" because we intend to convey that privacy is not required for D_{npriv} even though that dataset may not be publicly available.

Let $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ denote the set of domains, each of which could be categorical or numeric. Let $D = D_{priv} \cup D_{npriv}$ denote the database owned by the curator. D_{priv} consists of m rows denoted as $\{x_1, x_2, \ldots, x_m\}$, where each $x_i \in \mathcal{D}'_1 \times \ldots \times \mathcal{D}'_s$ where $\mathcal{D}'_1, \ldots, \mathcal{D}'_s \in \mathcal{D}$. Similarly, D_{npriv} consists of ℓ rows denoted as $\{z_1, z_2, \ldots, z_\ell\}$, where each $z_i \in \mathcal{D}''_1 \times \ldots \times \mathcal{D}''_t$ where $\mathcal{D}''_1, \ldots, \mathcal{D}''_t \in \mathcal{D}$.

Extending [8] to the new model defined above, two databases D_1 and D_2 *differ in at most one element* if the private subset of one is a proper subset of the private subset of other and the larger database just contains one additional row. D_1 and D_2 are then said to be neighboring databases. Since differential privacy needs to be preserved only for the dataset D_{priv}, the notion of neighboring datasets applies only to D_{priv} and not to D_{npriv}.

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 $S \subseteq$ Range($\mathcal{M}$), $Pr[\mathcal{M}(D_1) \in S] \leq \exp(\epsilon) * Pr[\mathcal{M}(D_2) \in S]$. The probability is taken with respect to the randomized algorithm $\mathcal{M}$.

Let $f : X \rightarrow R^d$, where X is the set of databases. Dwork et al. [8] gave a technique for a mechanism $\mathcal{M}$ that computes the function f to achieve differential privacy. This is done by adding a noise proportional to the sensitivity of f, drawn from a suitably chosen distribution, to the output of $f(D)$, where $D \in X$.

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, $\| f(D_1) - f(D_2) \|_1 \leq S(f)$.

Let $Lap(\lambda)$ stand for the Laplacian distribution with mean 0 and standard deviation $\sqrt{2}\lambda$. The following theorems are proven in [8], [17].

Theorem 1 ([8]): For all functions f, the mechanism that outputs $f(D) + (Y_1, \ldots, Y_d)$ achieves ϵ-*differential privacy*. Here Y_i are drawn i.i.d from $Lap(S(f)/\epsilon)$.

This mechanism is referred to as the Laplace mechanism. Smaller values of ϵ require that more noise be added when query results are returned.

Theorem 2 ([17]): The sequential application of mechanisms $\mathcal{M}_i$, each giving ϵ_i-differential privacy gives $\sum_i \epsilon_i$-*differential privacy*.

Our model, as described, is fairly general in that either or both of D_{priv} and D_{npriv} could be labeled or unlabeled. However, in the context of the algorithm presented in this paper, D_{priv} is assumed to be labeled data while D_{npriv} is unlabeled data. Also, a single row in two neighboring databases could differ in the example, the label or both. Our algorithm also works for multi-class labels.

IV. A Differentially Private Semi-Supervised Classifier

In this section, we present our new differentially private classifier in the learning model described in Section III. Our classifier (**RDT# Classifier**), based on the random decision tree classifier [10] reduces to the random decision tree algorithm in [13] when restricted to labeled data alone. A random decision tree classifier is an ensemble method that uses random decision trees as base classifiers.

A. Differentially Private Random Decision Trees

Random decision tree classifiers were originally considered in the non-private setting for their efficiency in dealing with large databases [10]. Recently, it has been observed that they are also useful in the differential privacy model for small and moderate sized databases [13].

Unlike conventional decision trees, such as those created by algorithms such as ID3, C4.5 or CART, a random decision tree is created by choosing test attributes for the decision tree's nodes completely at random. The selection of an attribute for testing at a node does not depend on any splitting criterion that measures the attribute's predictive capabilities. The entire structure of such a random decision tree can be precomputed using the attribute list alone, before any training data is examined. This is a key characteristic of the random decision tree that helps in the construction of accurate differentially-private classifiers. The training instances are incorporated into the structure to compute the distribution of class labels at all the leaves of the tree. The differentially private random decision tree ensures privacy by invoking the Laplace mechanism. Since the summary stored in the leaves of a random decision tree has low sensitivity, the leaves need only a small amount of Laplacian "noise" added. A differentially-private random decision tree is an ensemble of such differentially-private trees. To classify a test instance the classifier averages the predictions from all the trees in the ensemble. The private classifier works very well even for relatively small data sets [13].

The problem for random decision trees fundamentally lies in the scattering of the training instances into the partitions of the instance space induced by a random decision tree. One way to visualize this partitioning is by observing that if all the attributes in a data set are real valued, a random decision tree (Figure 1) partitions the space into axis parallel regions (Figure 2). The partitions correspond to the leaves of the tree. Whether or not the attributes are real valued, all training examples that arrive at a leaf of such a tree belong to the same partition, and hence have similar attribute values. If the number of partitions induced by a tree is large, each partition would likely have a small number of rows of the training data. The noise added to the summary values in each partition could overwhelm the true counts, thereby leading to poor utility. On the other hand, if the number of partitions is small, there would be more rows of the data set in each partition. However, the discriminatory power of each such partition (the "purity" of a leaf node) would likely be low because it spans a large region of the instance space. This also leads to poor utility. This situation also occurs when the dataset suffers from sparsity: That is, a small dataset in high-dimensional space or when there are insufficient examples to explore the instance space.

B. A semi-supervised learning approach

In the scenario considered in this paper the curator wishes to create a classifier using a database that has a small set of private labeled instances, and a relatively larger non-private set of unlabeled instances. The problem is analogous to the well-known semi-supervised learning model where unlabeled examples are used to improve the accuracy of a classifier that has access to only a small amount of labeled data. Our differentially private classifier problem uses unlabeled instances in two ways: (i) to control the partitioning of the instance space so that denser regions of the space are partitioned more finely than sparse regions and (ii) to "propagate" the labels from the labeled instances to larger regions of the instance space. Together, these techniques boost the utility of the differentially private classifier without lowering privacy. In the next section we detail our new learning algorithm **RDT# Classifier**.

C. RDT# Classifier

We now describe how to alter the RDT algorithm to handle unlabeled data and data sparsity. Dense regions in the instance space can be discovered if there are a large number of (unlabeled) instances. We extend the random decision tree idea to exploit the availability of (non-private) unlabeled data as follows: In each partition of the instance space induced by the random decision tree, the algorithm uses the unlabeled instances of that partition to find dense regions in the instance space. The discovery of such regions allows

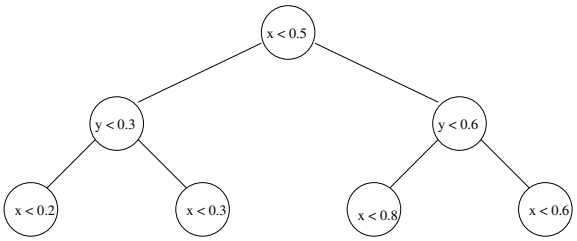

Figure 1. Example of an RDT tree

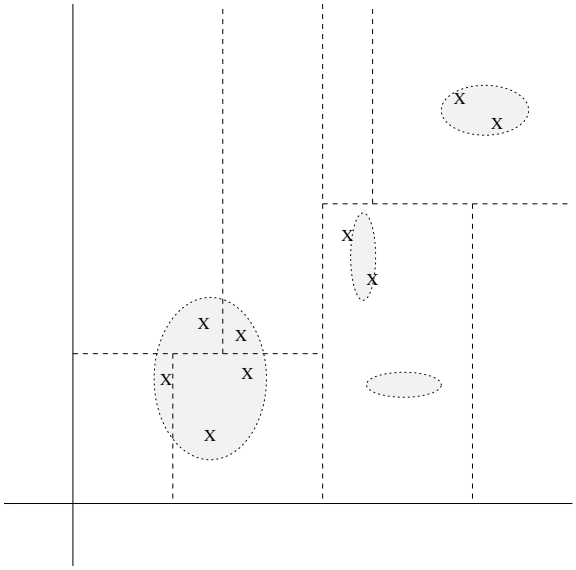

Figure 2. Using unlabeled examples

us to build decision trees with approximately the correct number of leaves. Under the assumption that the (private) labeled instances have the same distribution as the unlabeled instances, these labeled instances are not scattered either. This avoids the problem of having the summary counts in the leaves of a random decision tree being too small or the partitions being too big.

In keeping with the spirit of random decision trees, we use a simple strategy to find dense regions in a partition: We randomly choose a small subset of the unlabeled instances in the partition. The instances in this subset serve as the "centers" of dense regions (Figure 2). The values in the instance space for which a center c is the closest center form the *neighborhood* of c. Note that we do not explicitly seek to discover the contours of these regions or estimate how many of them exist. This scattershot approach may place multiple centers within a single dense region. This is unlikely to pose a problem since the algorithm builds an ensemble of such trees.

Having constructed such a tree, the algorithm uses the (private) labeled instances to find labels for the leaves of the tree. More precisely, we store with each center c the distribution (counts) of class labels for the labeled instances in the neighborhood of c. Adding noise to these counts by sampling from an appropriate Laplace distribution makes the tree differentially private. The main algorithm creates a *primary* ensemble with a small number of these differentially private RDT# trees. To label a new instance, the algorithm finds the label distribution for the instance from each of the trees in the ensemble, averages them and predicts the class with the highest frequency.

The algorithm uses this primary ensemble to compute (pseudo) labels for all the unlabeled examples. The labels are propagated to a larger portion of the instance space. The algorithm then creates a large *secondary* ensemble of RDT# trees (or classifiers) and populates them with these newly labeled instances. It returns as the final classifier the union of the primary and secondary ensembles.

The algorithm to build a single RDT# tree is shown in Figure 3. The algorithm as presented works only for categorical attributes, though it can easily be extended to continuous-valued attributes as in [10]. The algorithm recursively creates the structure of the tree (**CreateTree**). When a leaf node of the tree is created, it selects at random the centers for the dense regions within it. The number of centers is proportional to the fraction of the unlabeled instances that arrive at that leaf.

Next, the algorithm incorporates the labeled examples into the tree (**AddLabeled**, **AddInstance**), "filtering" each instance through the tree to its leaves. Each center c in the leaves of the tree holds T counters, $\alpha_c[1], \ldots, \alpha_c[T]$, where T is the number of possible labels for training instances. The running time of the algorithm is linear in the size of the database. The **RDT# Classifier** is an ensemble of RDT#. When a test instance needs to be classified, the posterior probability is output as the weighted sum of the the probabilities output from the individual trees (Figure 4).

The main algorithm is listed in Figure 5. It builds a primary ensemble E_1 of p trees. The **AddNoise** function makes the tree differentially private by adding noise sampled from a Laplace distribution. Then, it uses E_1 to label all the instances in the unlabeled set. Next it builds a secondary ensemble E_2 of t trees using this pseudo labeled set. The output of this algorithm is the union of E_1 and E_2. Note

that the ensemble E_2 is built using non-private data.

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

Proof: Let A denote the **RDT# Classifier** algorithm. Since the p-sized primary ensemble E_1 is the only part of the algorithm that uses the private instances, it suffices to show that each tree in E_1 is ϵ'-differentially private, where $\epsilon' = \epsilon/p$. For a RDT# tree R, we denote the vector formed by concatenating the α_c for all centers c of R by $\lambda(R)$.

Consider a RDT# structure into which no labeled instances have yet been incorporated. Let D_1 and D_2 be two databases differing in at most one element of the private set. Assume that they 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 differs 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 $e^{\epsilon'}$. Since the structure of the random decision tree is generated even before the labeled 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 $e^{\epsilon'}$, 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$(1/\epsilon')$. ∎

V. Experiments

In this section, we present our experimental results that show that non-private data can be used to improve the utility of the differentially private random decision tree classifier [13], especially when the size of the private data set is relatively small. We ran experiments to measure the accuracy of private RDT# ensembles for various values of the privacy parameter ϵ. We implemented our algorithms in Java using the Weka machine learning framework [20].

Experimental Setup: In general, many of the differential privacy-preserving mechanisms are known to yield accurate results when they are run in conjunction with large data sets [6], [4], [11]. Smaller values of the privacy parameter ϵ require larger data sets to reach a desired level of utility. This series of experiments was run with labeled sets with as few as 140 instances. The smallest value of ϵ we used was 0.5. The algorithm continues to gradually lose utility with further reductions in ϵ, which is to be expected, considering the amount of noise being added.

We now explain the two parameters of the algorithm and the values we used for them. The size of the primary ensemble (p) is the first user-specified parameter. In the non-private version, increasing the size of the RDT ensemble results in an increase in the overall accuracy of the classifier, with the improvements eventually tapering off [10]. However, the relationship is not quite as straightforward when differential privacy is involved. Since we need to use the composition theorem 2 to aggregate the predictions from p trees, the

Algorithm RDT#
Input: D_{priv}, the private labeled data set,
D_{npriv}, the non-private unlabeled data set,
X, the set of attributes, and
h, the height of the tree.
Output: A RDT# R

R = **CreateTree**(D_{npriv}, $|D_{npriv}|$, X, 0, h)
AddLabeled(R, D_{priv})
return R

Subroutine **CreateTree**(D,N,X,d,h)

if $d = h$ **then**
  Create a leaf node
  $num\_centers = \lceil N\_CELLS * |D|/N \rceil$
  Select $num\_centers$ instances of D at random as dense region centers for this leaf
  return 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
  Split D on F to give subsets $D_1, \ldots, D_m$
  for i = 1 to m **do**
    c_i = **CreateTree**(D_i,N,$X - \{F\}$,$d+1$,h)
    Add c_i as a child of r
  end for
  return r
end if

Subroutine **AddLabeled**(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 c be the region center in r closest to x
  Let $\alpha_c[t]$ = # of t-labeled rows that reach the region centered at c
  $\alpha_c[t] \leftarrow \alpha_c[t] + 1$
end if

Figure 3. RDT# Algorithm

Algorithm Classify

Input: $\{R_1, \ldots, R_N\}$, an ensemble of RDT# trees, 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

Figure 4. Computing the probability for each possible label for a test instance

Algorithm RDT# Classifier

Input: D_{priv}, the private labeled data set, D_{npriv}, the non-private unlabeled data set, and ϵ, the privacy parameter.

Output: A RDT# ensemble

Create p RDT# trees for the primary ensemble E_1 using D_{priv} and D_{npriv}.
for each R in E_1 **do**
  AddNoise(R,ϵ/p).
end for
for each instance x in D_{npriv} **do**
  $distrib$ = **Classify**(E_1,x)
  Assign x to the class with the highest probability in $distrib$.
end for
Create t RDT# trees for the secondary ensemble E_2 using D_{npriv}.
return $E_1 \cup E_2$.

Subroutine **AddNoise**(R,ϵ)
for each center c in R **do**
  for each i in $\{1, \ldots, T\}$ **do**
    $\alpha_c[i] = \alpha_c[i] + \texttt{Lap}(1/\epsilon)$
  end for
end for

Figure 5. **RDT# Classifier**.

privacy parameter for each tree reduces to ϵ/p. This implies that our algorithm needs to sample noise from a distribution whose standard deviation is proportional to p. Since the labeled dataset is fairly small, the values in the α counters of the tree can be easily overwhelmed for large values of p. In [10], Fan et al. suggest that as few as 10 trees suffice. In [13], the smallest dataset uses 5 trees. We use the same value 5 for p.

The second parameter is the total number of centers in all leaves (N_CELLS). The implementation of our algorithm ensures that each leaf of the random decision tree has at least one center. Ideally one could run a clustering algorithm on the unlabeled instances of each leaf to discover the true centers of the dense regions within it. Instead, we use a global parameter (N_CELLS) to control the number of centers. We set this parameters to 10 for all our experiments. Finding a good choice of this parameter given the data set and the privacy budget (the overall ϵ) is interesting future work.

Since the secondary ensemble is built using the differentially private primary classifier (and does not directly use private data), the composition theorem 2 does not apply here. Larger secondary ensembles do increase the accuracy of the classifier. The curator, after having created the primary ensemble may add a secondary ensemble as large as their computational limitations permit. Our results were obtained by setting the size of the secondary ensemble t to 200 trees. We observed that there is little additional utility to be obtained by using even larger ensembles. Following the same lines of reasoning as in [13], we set the height of each tree to $h = \min(\lfloor k/2 \rfloor, (\lfloor \log_b n \rfloor 1))$, 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 private database.

We performed our experiments on three data sets from the UCI Machine Learning Repository [1], namely the **Nursery**, **Mushroom** and **Congressional Voting Records** data sets and on one synthetic data set that we generated. We removed from the **Mushroom** database the attribute that has missing entries. In the **Congressional Voting Records** database, we handled missing data by replacing them with the majority vote for that bill. The synthetic data set was generated from a handmade Boolean decision tree. We added noise to the data set by flipping each class label with a probability of 0.05 to make these synthetic data set more realistic. See Table I for data characteristics.

We split each data set randomly into the labeled and unlabeled data sets for each run of the algorithm. We did not use a fixed number or percentage of the original data set for the labeled subset. Instead, after fixing ϵ at 0.5, for each data set we repeatedly ran the supervised algorithm in [13] with increasingly larger subsets until the error rate was no worse than 0.3. We used this value as the size of the labeled subset. For the **Mushroom** data set the size of the labeled set was 2% of the overall data, for the **Nursery** data set it amounted to about 20% of the overall data set, and for the **Congressional Voting Records** data set we had to use 30% of the original data set as the size of the labeled subset. This part of the experiment is for simulation purposes only. In practice the curator will have a well defined labeled and unlabeled set. We created 10 ensembles (each with its own primary and secondary ensembles) for each data set. The algorithm was run 10 times for each ensemble with privacy parameter ϵ being varied from 0.5 to 1 in steps of 0.1. For comparison, we also ran the algorithm from [13] with the

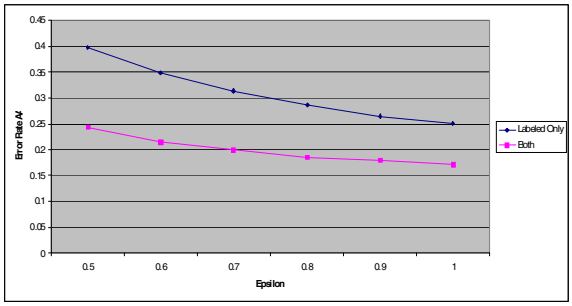

Figure 6. Error on the Nursery data set from the UCI Repository.

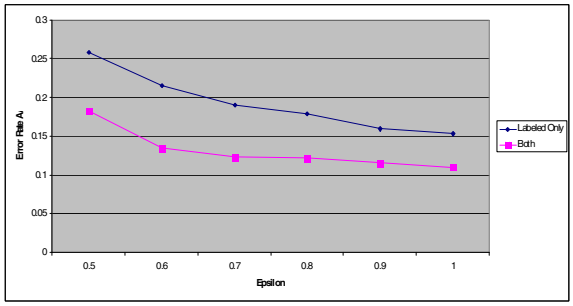

Figure 7. Error on the Mushroom data set from the UCI Repository.

labeled data alone. We used the stratified cross-validation technique available in Weka.

Results: The Figures 6, 7 and 8 show the improvement in error rate of the **RDT# Classifier** over the differentially private RDT classifier that uses only the labeled set in each case. In Figure 9 we show the improvement in error rates for the synthetic data set we used. The standard deviation for error rates is in Table II.

As can be seen, the use of public unlabeled data can indeed provide a noticeable decrease in error. Clearly, not all data sets benefit equally from the availability of unlabeled data. This non-uniformity is not a surprise and similar observations have been made in the semi-supervised setting in machine learning [19]. Although not seen directly in these figures, we observed that classifiers in which the primary ensemble had higher accuracy, unsurprisingly, had higher accuracy overall. However, the relationship between the accuracy of the primary ensemble and that of the overall classifier is not simple. When accuracy of the primary ensemble is close to the frequency of the most frequent class label (see Table I), unlabeled examples do not help very much. At the other end of the spectrum, when the accuracy of the primary ensemble is very high, unlabeled examples again provide only marginal help. Overall, as each of these figures show, the **RDT# Classifier** algorithm has good accuracy, even for relatively small data sets. It is to be expected that the error rate decreases as the amount of labeled data increases. Figure 10 shows the decrease in error rate as the number of labeled instances increases in the **Mushroom** data set. Figure 11 shows that unlabeled instances are important as well; it shows the reduction in the error rate as the size of the unlabeled set increases.

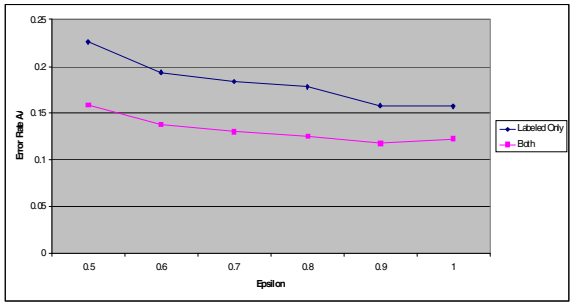

Figure 8. Error on the Congressional Voting Records data set from the UCI Repository.

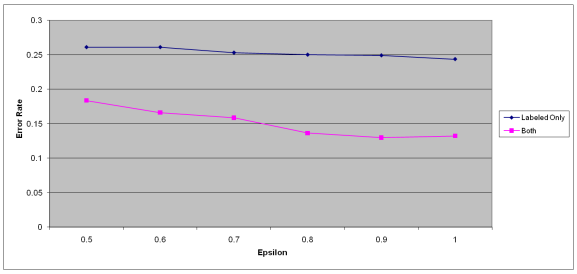

Figure 9. Error on a synthetic data set. It is based on a function defined on 9 attributes.

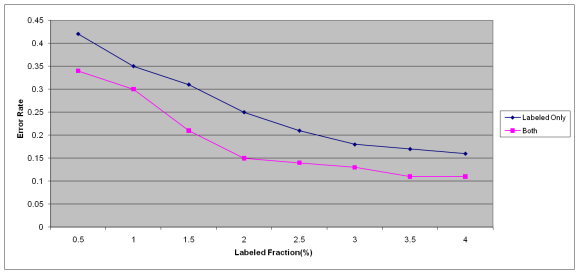

Figure 10. The error reduces as the size of the labeled set increases. The experiments were run on the **Mushroom** data set.

REFERENCES

[1] A. ASUNCION AND D.J. NEWMAN, *UCI Machine Learning Repository*, 2007.

[2] A. BLUM, C. DWORK, F. MCSHERRY, AND K. NISSIM, *Practical privacy: The SuLQ framework*, in PODS '05, 2005, pp. 128–138.

Data set	# attribs	# rows	# Class labels	Class Distribution
Nursery	8	12960	3	[33.3%,0.0%,2.5%, 32.9%,31.2%]
Mushroom	22	8124	2	[51.8%, 48.2%]
Cong. Votes	16	435	2	[45.2%, 54.8%]
Synth. Data	9	2000	2	[49.3%, 50.7%]

Table I
EXPERIMENTAL DATA CHARACTERISTICS

Data Set/ϵ =	0.5	0.6	0.7	0.8	0.9	1.0
Mushroom	0.019	0.019	0.0123	0.015	0.02	0.015
Nursery	0.042	0.03	0.031	0.026	0.028	0.025
Votes	0.030	0.023	0.018	0.013	0.014	0.018
Synth.	0.026	0.029	0.023	0.034	0.029	0.032

Table II
STANDARD DEVIATIONS FOR ALL DATA SETS FOR **RDT# Classifier**.

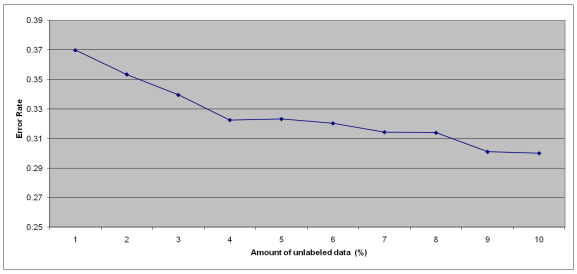

Figure 11. The error reduces as the size of the unlabeled set increases. The experiments were run on the **Nursery** data set.

[3] A. BLUM, K. LIGETT, AND A. ROTH, *A learning theory approach to non-interactive database privacy*, in STOC '08, 2008, pp. 609–618.

[4] L. CHAO AND M. GEROME, *An adaptive mechanism for accurate query answering under differential privacy*, Proc. VLDB Endow., 5 (2012), pp. 514–525.

[5] O. CHAPELLE, B. SCHÖLKOPF, AND A. ZIEN, eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.

[6] C. DWORK, *Differential privacy: A survey of results*, in TAMC : Theory and Applications of Models of Computation, 5th International Conference, 2008, pp. 1–19.

[7] ———, *A firm foundation for private data analysis*, Commun. ACM, 54 (2011), pp. 86–95.

[8] CYNTHIA DWORK, FRANK MCSHERRY, KOBBI NISSIM, AND ADAM SMITH, *Calibrating noise to sensitivity in private data analysis*, in TCC, 2006, pp. 265–284.

[9] CYNTHIA DWORK, GUY N. ROTHBLUM, AND SALIL P. VADHAN, *Boosting and differential privacy*, in FOCS, 2010, pp. 51–60.

[10] W. FAN, H. WANG, P.S. YU, AND S. MA, *Is random model better? On its accuracy and efficiency*, in ICDM '03, 2003, p. 51.

[11] A. FRIEDMAN AND A. SCHUSTER, *Data mining with differential privacy*, in KDD, 2010, pp. 493–502.

[12] K. GEORGIOS AND P. STAVROS, *Practical differential privacy via grouping and smoothing*, in Proc. 39th Int. Conf. of VLDB, 2013, pp. 301–312.

[13] G. JAGANNATHAN, K. PILLAIPAKKAMNATT, AND R. N. WRIGHT, *A practical differentially private random decision tree classifier*, Trans. Data Privacy, 5 (2012), pp. 273–295.

[14] P. JAIN AND A. THAKURTA, *Differentially private learning with kernels*, ICML, (2013), pp. 118–126.

[15] V. KARWA AND A. B. SLAVKOVIC, *Differentially private graphical degree sequences and synthetic graphs*, in Privacy in Statistical Databases, 2012, pp. 273–285.

[16] J. LEE AND C. CLIFTON, *Differential identifiability*, in Proc. 18th ACM SIGKDD, 2012, pp. 1041–1049.

[17] F. MCSHERRY AND K. TALWAR, *Mechanism design via differential privacy*, in FOCS '07, 2007, pp. 94–103.

[18] NOMAN MOHAMMED, RUI CHEN, BENJAMIN C. M. FUNG, AND PHILIP S. YU, *Differentially private data release for data mining*, in KDD, 2011, pp. 493–501.

[19] AARTI SINGH, ROBERT D. NOWAK, AND XIAOJIN ZHU, *Unlabeled data: Now it helps, now it doesn't*, in NIPS, 2008, pp. 1513–1520.

[20] I. H. WITTEN AND E. FRANK, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.