Privacy-Preserving Decision Tree Mining Based on Random Substitutions[*]

Jim Dowd, Shouhuai Xu, and Weining Zhang

Department of Computer Science, University of Texas at San Antonio
{jdowd, shxu, wzhang}@cs.utsa.edu

Abstract. Privacy-preserving decision tree mining is an important problem that has yet to be thoroughly understood. In fact, the privacy-preserving decision tree mining method explored in the pioneer paper [1] was recently showed to be completely broken, because its data perturbation technique is fundamentally flawed [2]. However, since the general framework presented in [1] has some nice and useful features in practice, it is natural to ask if it is possible to rescue the framework by, say, utilizing a different data perturbation technique. In this paper, we answer this question affirmatively by presenting such a data perturbation technique based on *random substitutions*. We show that the resulting privacy-preserving decision tree mining method is immune to attacks (including the one introduced in [2]) that are seemingly relevant. Systematic experiments show that it is also effective.

Keywords: Privacy-preservation, decision tree, data mining, perturbation, matrix.

1 Introduction

Protection of privacy has become an important issue in data mining research. A fundamental requirement of privacy-preserving data mining is to protect the input data, yet still allow data miners to extract useful knowledge models. A number of privacy-preserving data mining methods have recently been proposed [3, 4, 1, 5, 6], which take either a cryptographic or a statistical approach. The cryptographic approach [7] ensures strong privacy and accuracy via a secure multi-party computation, but typically suffers from its poor performance. The statistical approach has been used to mine decision trees [1], association rules [4, 6, 8, 9], and clustering [10], and is popular mainly because of its high performance. This paper focuses on the statistical approach to privacy-preserving decision tree mining.

The notion of privacy-preserving decision tree mining was introduced in the seminal paper [1]. However, the problem of privacy-preserving decision tree mining has yet to be thoroughly understood. In fact, the privacy-preserving decision tree mining method explored in [1] was recently showed to be completely broken, meaning that an adversary can recover the original data from the perturbed (and public) one. The reason for the attack to be so powerful was that the adopted

[*] This work was supported in part by US NFS grant IIS-0524612.

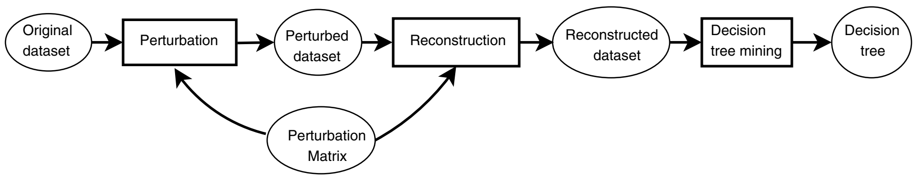


Fig. 1. A Framework of Privacy-Preserving Decision Tree Mining

data perturbation technique, called *noise-adding*, turned out to be fundamentally flawed [2].

In spite of aforementioned attack, the framework introduced in [1] (shown in Fig. 1) has a nice and useful feature: the perturbed data can be analyzed by arbitrarily many data miners using conventional mining algorithms. On one hand, an owner can protect its data by releasing only the perturbed version. On the other hand, a miner equipped with a specification of the perturbation technique can derive a decision tree that is quite accurate, compared to the one derived from the original data. We believe that this feature makes the framework a good-fit for many real-life applications. Therefore, it is natural to ask if the framework can be rescued by, say, utilizing some other data perturbation technique.

This paper makes several contributions towards privacy-preserving decision tree mining. What is perhaps the most important is that the framework introduced in [1] can be rescued by a new data perturbation technique based on *random substitutions*. This perturbation technique is similar to the randomization techniques used in the context of statistical disclosure control [11, 12], but is based on a different privacy measure called ρ_1-to-ρ_2 privacy breaching[1] [6] and a special type of perturbation matrix called the γ-diagonal matrix [4]. This utilization of both ρ_1-to-ρ_2 privacy breaching and γ-diagonal matrix seems to be new. This is because both [6] and [4] were explored in the context of privacy-preserving association rule mining. Further, it was even explicitly considered as an open problem in [4] to extend the results therein to other data mining tasks such as decision tree mining. For example, the integration of the perturbation matrix of [4] is non-trivial, because we need to make it work with continuous-valued attributes. As a consequence, we need to analyze the effect of the dimension size of the matrix with respect to the accuracy of decision trees and the performance of the system. To this end, we introduce a novel error-reduction technique for our data reconstruction, so that it not only prevents a critical problem caused by a large perturbation matrix, but also guarantees a strictly better accuracy (see Section 3.3 for a theoretic treatment and Section 5.1 for experimental results).

In addition, we show that the resulting privacy-preserving decision tree mining method has the following properties.

[1] Roughly, a perturbation technique prevents a ρ_1-to-ρ_2 privacy breaching if the adversary has an a prior probability about some private information in the original data no more than ρ_1, she can derive a posterior probability about the same private information no more than ρ_2.

– It is immune to the relevant *data-recovery* and *repeated-perturbation* attacks. The data-recovery attack was introduced in [2] (and further explored in [13]) to recover the original data from a given perturbed dataset. Our method is immune to this attack because the random substitution technique is fundamentally different from the noise-adding one [1]. Specifically, noise-adding technique draws noises from a *single* probabilistic distribution, but the random substitution draws noises from *many different* distributions, even if it is "artificially and mechanically" viewed as a kind of noise-adding. This suffices to defeat the data-recovery attack, which crucially relies on that the noises are drawn from the *same* distribution. The repeated-perturbation attack is introduced in this paper based on the observation that an adversary may repeatedly perturb the data with the hope to recover the original data. This attack can be effective if the perturbation technique can be viewed as a function f that has the property $f(...f(f(x))) = x$, where x is an original dataset and f is known to the adversary (or a data miner). Fortunately, we are able to show rigorously that our method is also immune to this attack. While our perturbation technique may have inherited the privacy assurance guarantee of [6], the *data-recovery* and *repeated-perturbation* attacks were not accommodated in the model of [6].
– It ensures that the decision trees learned from the perturbed data are quite accurate compared with those learned from the original data. Furthermore, a small number of parameters are seemingly sufficient for capturing some important trade-offs in practice. The parameters include: (1) the privacy assurance metric γ, (2) the dimension size N of perturbation matrix, which affects the accuracy of reconstructed data as well as performance, and (3) the entropy of a perturbation matrix, which, to some extent, can encompass the effect of both γ and N. Systematic experiments show that there is a strong correlation between entropy of the perturbation matrix and accuracy of the resulting decision tree. As a result, both γ and accuracy can be used to select N. Furthermore, it is showed that by selecting an appropriate N, one can achieve the desired trade-off between accuracy, privacy, and performance.

The rest of the paper is organized as follows. In Section 2, we present random substitution perturbation algorithm and analyze its immunity to the data-recovery attack. In Section 3, we present the reconstruction algorithm with heuristic methods for reducing estimation error of original data distributions. In Section 4, we analyze the effect of perturbation matrix parameters and the immunity to the repeated-perturbation attack. In Section 5, we present results from our experiments. In Section 6, we discuss how to select perturbation matrix parameters in practice. Section 7 concludes the paper.

2 Random Substitution Perturbation and Analysis

In the following, we assume that data records have attributes $A_1, \ldots, A_q$, with discrete- or continuous-valued domains. Without loss of generality, we consider the perturbation of a single discrete- or continuous-valued attribute, because it is straightforward to extend the method to perturb a set of attributes together.

2.1 Perturb a Discrete-Valued Attribute

The basic idea is to replace the value of each data record under the attribute by another value that is chosen randomly from the attribute domain according to a probabilistic model. The general algorithm is given in Algorithm 1 and explained in the following.

Algorithm 1. Random Substitution Perturbation (RSP)

```
Input: a dataset $\mathcal{O}$ of $n$ records, an attribute $A$ with domain $\mathcal{U} = \{u_1, \ldots, u_N\}$, and a perturbation matrix $\mathbf{M}$ for $\mathcal{U}$.
Output: the perturbed dataset $\mathcal{P}$.
Method:

    $\mathcal{P} = \emptyset$;
    for each $o \in \mathcal{O}$ begin
    1.  $k = getIndex(o[A])$;
    2.  $p = o$;
    3.  obtain a random number $r$ from a uniform distribution over
        $(0, 1]$;
    4.  find an integer $1 \leq h \leq N$ such that $\sum_{i=1}^{h-1} m_{i,k} < r \leq$
        $\sum_{i=1}^{h} m_{i,k}$;
    5.  set $p[A] = getValue(h)$;
    6.  add $p$ to $\mathcal{P}$;
    return $\mathcal{P}$;
```

To perturb a set of data records $\mathcal{O} = \{o_1, \ldots, o_n\}$ on an attribute A, we create a perturbation matrix $\mathbf{M}$ for the attribute domain $\mathcal{U} = \{u_1, \ldots, u_N\}$. For each $u_k \in \mathcal{U}$, $p(k \to h) = \Pr(u_k \to u_h)$ denotes the (transition) probability that u_k is replaced by $u_h \in \mathcal{U}$. The perturbation matrix is then defined as $\mathbf{M} = [m_{h,k}]_{N \times N}$, where $m_{h,k} = p(k \to h)$. Since each value must be replaced by a value in $\mathcal{U}$, $\sum_{h=1}^{N} m_{h,k} = 1$, for $1 \leq k \leq N$. Therefore, each column k of $\mathbf{M}$ defines a probability mass function, that is, $p(h) = m_{h,k}$ for $1 \leq h \leq N$, and a cumulative probability function $F(a) = \sum_{h=1}^{a} m_{h,k}$, where $1 \leq a \leq N$. The choice of probabilities in the perturbation matrix is an important issue (in particular, it is related to the privacy assurance guarantee) that will be described in Section 4.

We associate two functions with the attribute domain: function $getIndex(u)$, which returns index of value u (that is i if $u = u_i$), and function $getValue(i)$, which returns the ith value in $\mathcal{U}$ (that is u_i). Naturally, we call $\mathcal{O}$ and $\mathcal{P}$ the original and the perturbed dataset and the records in them the original and perturbed records, respectively.

We now explain steps of algorithm RSP. In step 1, we obtain the index of domain value $o[A]$. Step 2 initializes the perturbed data record. Steps 3 and

4 determine the index of the replacement (or perturbed) value using the perturbation matrix. Notice that we can always find the index h that satisfies the condition of Step 4. Step 5 replaces the original value by the value chosen in Step 4. Finally, the perturbed record is added to the perturbed dataset in Step 6. The time complexity of Algorithm RSP is $O(n \cdot N)$.

2.2 Perturb a Continuous-Valued Attribute

One way to apply RSP to a continuous-valued attribute is to discretize the attribute domain into intervals $I_1, \ldots, I_N$ for some given N, and to define the perturbation matrix over the discretized domain $\mathcal{U} = \{I_1, \ldots, I_N\}$. As a result, each element $m_{h,k}$ of the perturbation matrix is now interpreted as the probability that a value in I_k is replaced by a value in I_h. To maintain consistent semantics, each interval is represented by a value that is contained in it. This value can be either a fixed value, such as the center or an endpoint of the interval, or a randomly selected value. Thus, in this case, the function $getIndex(u)$ returns index i, such that $u \in I_i$ and function $getValue(i)$ returns the representative value of I_i. In our experiments, the representative value of an interval is its center.

Many discretization methods are known. We use a simple equi-width binning method in our experiments. Without loss of generality, let the attribute domain be an interval of real numbers with finite endpoints (for simplicity), that is $[l, u) \subset \mathbf{R}$, where $-\infty < l < u < \infty$. With a user-specified parameter $N > 1$, the discretization method partitions the domain into N intervals (also called bins) $I_i = [l_i, u_i)$, such that, $l_1 = l$, $u_N = u$, $u_i = l_{i+1}$ for $1 < i < N$, and $u_i - l_i = \frac{u-l}{N}$. The choice of N has an impact on performance and will be further explored in Sections 4 and 6.

2.3 Why Is Random Substitution Fundamentally Different from Adding Noise?

The adding noise perturbation method introduced in [1] is subject to what we call *data-recovery* attacks [2, 13], which can accurately derive the original data from the perturbed data. It is natural to ask if these attacks will also be effective against the random substitution perturbation method. In this section, we show that the answer to this question is negative. These attacks are based on, among other things, a crucial assumption that the noises are independently drawn from a single distribution and the noise variance σ^2 is known. While this assumption is certainly valid for adding noise perturbation of [1] due to its very design, we show that this assumption is no longer valid for the random substitution perturbation. Thus, the random substitution perturbation method is immune to the attacks explored in [2, 13].

The basic idea is to view the random substitution perturbation as a special adding noise perturbation and show that the added noises must be drawn from *different* probabilistic distributions that depend on the original data.

Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be a set of original data and $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a set of perturbed data that are obtained using the random substitution perturbation. The original data can be viewed as a realization of a set $O = \{O_1, \ldots, O_n\}$ of independent, identically distributed random variables, and the perturbed data as a realization of another set $P = \{P_1, \ldots, P_n\}$of random variables. By the design of the random substitution perturbation, all these random variables have the same domain, which is assumed without loss of generality to be a set $D = [a, b]$ of integers where $a < b$.

The random substitution perturbation can be viewed as a special case of adding noise perturbation: for each original data o_i, it draws a noise r randomly from the interval $[-(b-a), (b-a)]$ with a probability

$$\Pr[r \mid o_i] = \begin{cases} m_{k,o_i}, & \text{if } a \leq k = o_i + r \leq b; \\ 0, & \text{otherwise.} \end{cases}$$

and generates a perturbed data $p_i = o_i + r$. It is easy to verify that this special adding noise perturbation is indeed equivalent to the random substitution perturbation. The following theorem[2] indicates that for this special adding noise perturbation, if the perturbation matrix allows any domain value to be perturbed into a different value, the probability distribution of the noise given an original data can be different from that given another original data, therefore, the noises must not be drawn from the same distribution.

Theorem 1. *If some non-diagonal element of the perturbation matrix is positive, that is, $m_{k,h} > 0$, for $k \neq h$, then $\exists o_i, o_j \in [a, b]$, $o_i \neq o_j$ and $\exists r \in [-(b-a), +(b-a)]$, such that* $\Pr[r \mid o_i] \neq \Pr[r \mid o_j]$.

3 Generating Reconstructed Dataset

3.1 A Reconstruction Algorithm

The purpose of creating a reconstructed dataset is to allow the data miner to learn decision trees using existing decision tree mining algorithms. We emphasize that while it can be used to learn accurate decision trees, a reconstructed dataset is not the original dataset and will not violate the privacy guarantee.

Algorithm 2 is the matrix-based reconstruction (MR) algorithm that we use to create the reconstructed dataset, which is based on a heuristic method of [1]. Using function $estimate(\mathcal{P}, \mathbf{M})$ (whose detail is given in Section3.2 below), it first estimates the data distribution of the attribute that we want to reconstruct, and sort the perturbed records on that attribute. Next, it heuristically assigns the attribute values to perturbed data records according to the estimated data distribution. For example, if the estimated distribution predicts that for $1 \leq i \leq N$, $Dist[i]$ original records have value $getValue(i)$ in attribute A, we assign $getValue(1)$ to the first $Dist[1]$ perturbed records, $getValue(2)$ to the next $Dist[2]$ perturbed records, and so on. If multiple attributes need to be reconstructed, we apply MR to one attribute at a time.

[2] Due to space limitation, proofs of all results of this paper have been omitted. The readers are referred to [14] for the proofs.

Algorithm 2. Matrix-based Reconstruction (MR)

```
Input: a perturbed dataset $\mathcal{P}$ , an attributes $A$ and an $N \times N$ perturbation
matrix $\mathbf{M}$ of $A$.
Output: a reconstructed dataset $\mathcal{R}$.
Method:

    $\mathcal{R} = \emptyset$;
    $Dist = estimate(\mathcal{P}, \mathbf{M})$;
    sort $\mathcal{P}$ on $A$ in ascending order;
    $next = 1$;
    for $i = 1$ to $N$ do begin
      for $j = 1$ to $Dist[i]$ do begin
        $r = p_{next}$;
        $next = next + 1$;
        $r[A] = getValue(i)$;
      end
    end
    return $\mathcal{R}$;
```

3.2 Estimating Original Data Distributions

We now briefly describe a simple method for estimating original data distribution [4]. Let $\mathcal{U}$ be the domain of a discrete-valued attribute containing N values. Recall that $\mathcal{O}$ and $\mathcal{P}$ are the original and perturbed datasets of n records, respectively. Let $\mathbf{M}$ be the perturbation matrix defined for $\mathcal{U}$. For each value $u_i \in \mathcal{U}$, let Y_i be the count (that is, total number) of u_i in a perturbed dataset generated from a given original dataset and let X_i be the count of u_i in the original dataset. Since from the data miner's point of view, $\mathcal{O}$ is unknown and many $\mathcal{P}$ can be randomly generated from a given $\mathcal{O}$, both X_i and Y_i, for $1 \leq i \leq N$, are random variables. Let $X = [X_1, \ldots, X_N]^T$and $Y = [Y_1, \ldots, Y_N]^T$ be the (column) vector of counts of the original and the perturbed datasets, respectively. For a given $\mathcal{O}$, we have

$$E(Y) = [E(Y_1), \ldots, E(Y_N)]^T = \mathbf{M}X$$

where $E(Y_h) = \sum_{k=1}^{N} m_{h,k} X_k = \sum_{k=1}^{N} p(k \to h) X_k$ is the expected number of u_h in any $\mathcal{P}$ perturbed from $\mathcal{O}$ and $p(k \to h)X_k$ is the expected number of u_h due to the perturbation of u_k. If $\mathbf{M}$ is invertible and $E(Y)$ is known, we can obtain X by solving the following equation

$$X = \mathbf{M}^{-1} E(Y)$$

However, since $E(Y)$ is unknown, we estimate X by $\hat{X}$ with the following formula

$$\hat{X} = [\hat{X}_1, \ldots, \hat{X}_N]^T = \mathbf{M}^{-1}\mathbf{y} \quad (1)$$

where $\mathbf{y} = [y_1, \ldots, y_N]$ is the number of u_k in the observed perturbed dataset $\mathcal{P}$. Notice that this method can be applied directly to estimate interval distribution of a discretized domain of an attribute.

3.3 Reducing Estimation Errors

A problem of the distribution estimation method is that $\hat{X}$ often contains an estimation error. For small N, such as that considered in [4], the error can be undetected, but for larger N, the error may cause some $\hat{X}_i$ to become negative, which can in turn cause the failure of the reconstruction. To resolve this problem, let us explore the estimation error.

Let $\hat{X} = X + \Delta$, where $\Delta = [\delta_1, \ldots, \delta_N]^T$ is a vector of errors. It is obvious that if the 1-norm $||\Delta||_1 = \sum_{i=1}^{N} |\delta_i| = 0$, the estimate is accurate. Since neither X nor Δ is known, in general, we may not even know whether $\hat{X}$ contains an error. Fortunately, if the estimate $\hat{X}$ is accurate, it must satisfy the following constraints **C1** and **C2**.

C1: The 1-norm of $\hat{X}$ should be equal to n. This can be shown by the following Lemma.

Lemma 1. *For any set of n perturbed data, $||\hat{X}||_1 \geq n$, and furthermore, if $\hat{X} = X$, $||\hat{X}||_1 = n$.*

C2:$\hat{X}$ contains no negative element. This is because it is the estimated counts and if it contains a negative count, it must has an estimation error.

While **C1** can be used to detect an estimation error, the following proposition indicates that **C2** can also be used to reduce an estimation error in $\hat{X}$, and therefore lead to a useful heuristic (adopted in our experiments).

Proposition 1. *If $\hat{X}$ contains any negative element, setting the negative element to zero strictly reduces the estimation error.*

4 Perturbation Matrix and Analysis

Given that the privacy is measured by the (ρ_1-to-ρ_2) privacy requirement (as defined in [6]) and the accuracy is measured by the condition number, [4] showed that for a given $\gamma \leq \frac{\rho_2(1-\rho_1)}{\rho_1(1-\rho_2)}$, the optimal perturbation matrix is the gamma diagonal matrix $\mathbf{M} = x\mathbf{G}$, where $x = \frac{1}{\gamma+N-1}$, and

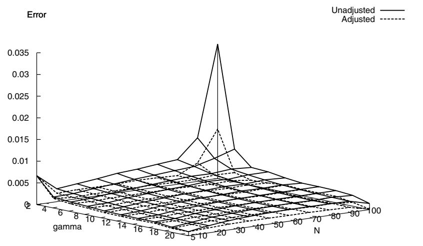
$$\mathbf{G} = \begin{bmatrix} \gamma & 1 & 1 & \cdots \\ 1 & \gamma & 1 & \cdots \\ 1 & 1 & \gamma & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

which guarantees γ and has a minimum condition number.

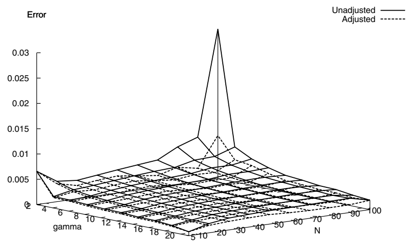
Now we investigate some important aspects of perturbation matrix that are relevant to privacy-preserving decision tree mining.

Diagonal vs Off-diagonal Elements. We observe that both *privacy* and *accuracy* are affected by the ratio of diagonal and off-diagonal elements in the perturbation matrix. As a starting point, consider the gamma diagonal matrix and let $\gamma = \infty$. In this case, $\mathbf{M}$ essentially becomes the identity matrix $\mathbf{I}$. It provides the maximum accuracy, since each original value is perturbed into itself, therefore, $E(Y) = X = Y$. But it also provides the minimum privacy guarantee, since $\gamma = \infty$ implies $\rho_1 = 0$ and $\rho_2 = 1$, that is, the perturbed data discloses the private information completely.

As γ reduces, diagonal elements $\frac{\gamma}{\gamma+N-1}$ will be smaller and off-diagonal elements $\frac{1}{\gamma+N-1}$ larger. As a result, each domain value is more likely to be perturbed into other values during the perturbation. Thus, the privacy guarantee is improved due to reduced γ and the estimation accuracy is reduced because the increased randomness in the perturbation matrix makes accurate estimation more difficult. As γ approaches 1, both diagonal and off-diagonal elements will converge to $\frac{1}{N}$, that is, the probability distribution of each column approaches the uniform distribution. This is the case of the maximum privacy guarantee and the minimum estimation accuracy. However, for practical reason, $\gamma = 1$ is not allowed. Otherwise, $\mathbf{M}$ becomes singular, and the estimation method is invalid since $\mathbf{M}^{-1}$ no longer exists.



(a) Plateau data



(b) Triangle data

Fig. 2. Estimation Errors of Two Original Data Distributions

The Dimension of Perturbation Matrix. In the previous analysis , we assume that the dimension N of $\mathbf{M}$ is fixed. What if N can also vary? Previous work has not considered the effect of N. In [6], the amplification factor concerns only the ratios between transition probabilities and does not care how many such probabilities there are. In [4], N is treated as a constant. In our work, when a continuous-valued (and maybe also some discrete-valued) attribute domain is discretized into N intervals, we need to decide what the N should be. Ideally, we should choose N to improve privacy guarantee and estimation accuracy.

Let us consider the ratio of the diagonal element to the sum of off-diagonal elements in a single row, which is the likelihood that a domain value is perturbed into itself. This is given by $\frac{\gamma}{N-1}$. Let us assume a fixed γ and a varying N. If $N = \infty$ or $N = 1$, we have a singular matrix, and both perturbation and distribution estimation are impossible. So, assume $1 < N < \infty$. As N increases, the likelihood decreases that a domain value will be perturbed into itself, since the diagonal elements are reduced. But, at the same time, the probability is also reduced that the domain value is perturbed into any other specific domain value, since non-diagonal elements also become much smaller. Thus it is more likely for the estimated counts of domain values to be negative, which will increase the estimation error.

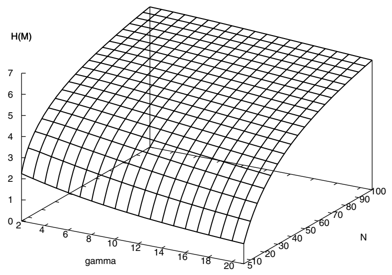


Fig. 3. Entropy of some perturbation matrices

The Entropy of Perturbation Matrix. So far, we analyzed how γ and N individually affect privacy and accuracy. But, it is also important to know how γ and N collectively affect these measures, particularly if we need to determine these parameters for a given dataset. Ideally, a single "metric" that can "abstract" both γ and N could simplify this task. To study this problem, we introduce a new measure of the perturbation matrix, namely, the entropy of perturbation matrix $\mathbf{M}$, which is defined as

$$H(\mathbf{M}) = \sum_{j=1}^{N} P_j H(j)$$

where P_j is the probability of value u_j in the original dataset, which captures the prior knowledge of the adversary, and $H(j) = -\sum_{i=1}^{N} m_{i,j} \log_2 m_{i,j}$ is the entropy of column j of the perturbation matrix. Since we do not have any prior knowledge about the original dataset, we assume that $P_j = \frac{1}{N}$, and therefore, $H(j) = -\frac{\gamma}{\gamma+N-1} \log_2 \frac{\gamma}{\gamma+N-1} - \sum_{i=1}^{N-1} \frac{1}{\gamma+N-1} \log_2 \frac{1}{\gamma+N-1}$, for any column j, and $H(\mathbf{M}) = -\frac{\gamma}{\gamma+N-1} \log_2 \frac{\gamma}{\gamma+N-1} - \frac{N-1}{\gamma+N-1} \log_2 \frac{1}{\gamma+N-1}$. Figure 3 shows the graph of $H(\mathbf{M})$ over a range of $2 \leq \gamma \leq 21$ and $5 \leq N \leq 100$. It is easy to see that for a given γ, the entropy increases as N increases, which indicates a decrease of estimation accuracy, and for a given N, the entropy increases as γ decreases, which indicates a increase of privacy guarantee. In Section 6, we will show that the entropy is a very useful instrument to give an insight of how γ and N affect the accuracy of decision tree.

Security Against the Repeated-Perturbation Attack. The random substitution perturbation described in Section 2 can be viewed as a two-step Markov chain with a specific N-state Markov matrix, namely, the gamma diagonal matrix.

This Markov chain is irreducible and ergodic since all the states communicate with each other and have period 1. An interesting question about this perturbation method is whether an adversary can gain any additional information by repeatedly perturbing the original dataset using the given perturbation matrix. We call this attack *repeated-perturbation.* In the following, we show that the effect of such a repeated perturbation will converge to the effect of a single perturbation using a perturbation matrix with the maximum entropy. Therefore, the adversary can *not* gain any additional information by repeatedly perturbing the perturbed dataset.

Assume that we apply the perturbation t times using the given perturbation matrix $\mathbf{M}$, the process is a Markov chain of $t+1$ steps. The t-step transition probability that u_j is replaced by u_i after the tth step is $m^t_{i,j} = \Pr[u_j \xrightarrow{t} u_i]$, which is the (i,j)th element of the t-step transition matrix $\mathbf{M}^t = \prod_{i=1}^{t} \mathbf{M}$. The following theorem says that the transition probabilities in $\mathbf{M}^t$ strictly converges to a uniform distribution as t approaches ∞, which implies that $\mathbf{M}^t$ has the maximum entropy (for the given N).

Theorem 2. *For any integer $t > 1$, $m^t_{i,i} > m^{t+1}_{i,i}$ and $m^t_{i,j} < m^{t+1}_{i,j}$, and $\lim_{t\to\infty} m^t_{i,j} = \frac{1}{N}$ for $1 \le i,j \le N$.*

5 Experiments

We performed extensive experiments to study the impact of perturbation matrix on the reconstruction of original data distribution and on the accuracy of decision trees. These experiments were run on a Pentium 4 PC with all algorithms implemented in Java.

5.1 Estimation Error of Data Distribution

In this experiment, we study how perturbation parameters affect the estimation error of original data distribution. We consider two (single-attribute) numerical datasets similar to those studied in [1]. The domain of these datasets is the integers between 1 and 200. We consider perturbation matrices with integer γ that varies from 2 to 21 and N that takes values 5, 10, 15, 20, 30, 40, $\ldots$, and 100. These give 240 combinations of γ and N (or different perturbation matrices). For each dataset and each combination of γ and N, we first discretize the domain into N equi-width intervals and create the perturbation matrix. We then repeat the following steps five times: perturbing the dataset, reconstruct the data distribution, measure the estimation error using

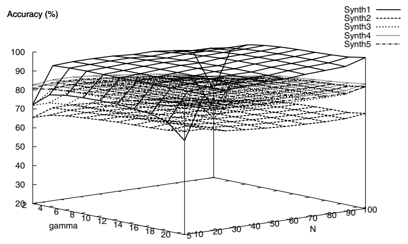
$$E = \frac{\sum_{i=1}^{N} |\hat{X}_i - X_i|}{\sum_{i=1}^{N} X_i}.$$

To reduce the randomness of the result, we report the average error over the five runs (see Fig. 2). To show the effects of the heuristic error reduction technique, we included errors both with and without applying the heuristic error reduction.

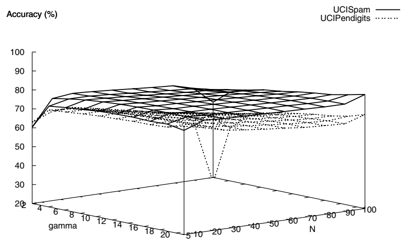
As shown in Fig. 2, the error surfaces of the two different data distributions are almost identical. This indicates that the estimation procedure is independent of data distributions and only depends on the perturbation parameters γ and N. Also, the error surfaces of the heuristically adjusted estimation are under that of unadjusted estimation. Thus, the heuristic error adjustment is effective. In fact, for most combinations of γ and N, the heuristic is able to reduce estimation error by 50%.

5.2 Decision Tree Accuracy

In this experiment, we study how perturbation matrix parameters affect decision tree accuracy (measured against testing sets). We considered 5 synthetic datasets that were studied in [1] and 2 datasets from the UCI Machine Learning Repository [15]. Again, we consider perturbation matrices based on the same 240 combinations of γ and N.



(a) Synthetic Datasets



(b) UCI Datasets

Fig. 4. Accuracy of decision trees learned from reconstructed data

Each dataset consists of a training set and a testing set. For each training set and each combination of γ and N, we discretize the domain and create the perturbation matrix as explained before. We then perturb the dataset, generate the reconstructed dataset, mine a decision tree, and measure the classification accuracy using the corresponding testing set. This process is repeated five times and the decision tree accuracy averaged over the 5 runs is reported here. The decision tree mining algorithm used is a version of C4.5 [16] implemented in Java.

Figure 4 shows the accuracy of decision trees for various combinations of γ and N. Notice that the accuracy surfaces for all datasets are lower than the accuracies of decision trees learned from original datasets. This confirms our intuition that decision trees learned from reconstructed datasets are less accurate than those learned from the original datasets. However, the overall accuracy is still reasonably high (about 80-85% for synthetic datasets and 75-80% for UCI datasets).

To illustrate the relationships among γ, N, and decision tree accuracies, we take the average of accuracies of decision trees that are learned from the 7 datasets on each combination of γ and N, and plot this average accuracy together with γ and N against the entropy of perturbation matrix. This is given in Fig. 5, which shows incredibly clear an insight into how γ and N affect the decision tree accuracy, therefore illustrates the usefulness of the entropy measure.

As shown in Fig. 5, accuracies in various intervals of entropy form bands of points that have dropping tails. As the entropy increases, the tails of bands drop increasingly earlier and deeper. It clearly shows that each band corresponds to a single N and include a point for every value of γ. In general, the entropy increases as N gets larger. Within a band, the accuracy decreases as γ decreases and the degree of the decrease depends on the corresponding N. When N is small, a small decrease of γ causes a small decrease of accuracy. But when N is large, a small decrease of γ can cause a large decrease of accuracy. This is because that for a given N, γ determines the probability that a data is perturbed into itself (that is, element $m_{i,i}$ of the perturbation matrix). That is the higher the γ is, the more likely the data is perturbed into itself. As γ decreases, the probability for a data to be perturbed into other values increases and that for it to be perturbed into itself decreases. This not only increases privacy but also increases the estimation error of data distribution, and therefore, reduces the accuracy of decision trees. This effect is compounded when N gets larger. This is because that the number of data records having a given value for a large N is lower than that for a small N. Thus, the estimation error of data distribution is larger for larger N than for small N. Thus, the effect of N is to intensify the effect of γ.

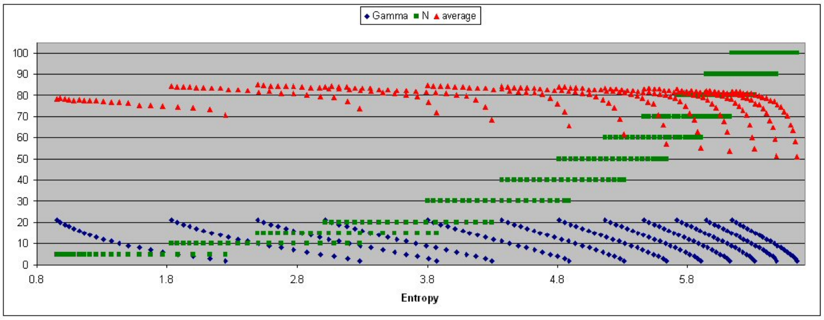


Fig. 5. Entropy vs. γ, N, and Average Accuracy . Notice that the vertical axis has **three** meanings: for γ (represented by diamonds), it is integer between 2 and 21 (i.e., on the bottom of the chart); for N (represented by squares), it is integer between 5 and 100 (i.e., scattered from bottom left to top right); for average accuracy (represented by triangles), it is percentage between 0% and 100% (the actual values are always above 50% and typically above 70%). This further shows the usefulness of the newly introduced notion of *entropy*, because it provides a "platform" to unify accuracy, privacy and performance.

On the other hand, when N is very small, each interval (resulted from discretization) will contain many distinct values of the domain. The reconstructed data distribution becomes very different from that of the original dataset, which explains why the accuracies at the low end of the entropy (for example, $N = 5$) in Fig. 5 are lower than those in some other intervals of entropy.

6 A Guide to Select Parameters in Practice: Putting the Pieces Together

Figure 5 (or any such figure obtained from representative domain datasets) can be used as a useful guide for data owners to determine the parameters of a perturbation matrix.

For example, given a dataset, the data owner can first determine the maximum of tolerable γ based on the ρ_1-to-ρ_2 privacy breaching measure. In practice, since large γ provides a poor protection of privacy and small γ reduces the accuracy, we suggest that a reasonable range of γ should be $5 \leq \gamma \leq 12$.

For a given set of γ (say $5 \leq \gamma \leq 12$), the data owner can find the highest accuracy from Fig. 5 for each γ in the set, and determine the N that corresponds to this accuracy. This will result in a set of combinations of γ and N among which the combination with the smallest N will be the best choice, since it will result in the smallest perturbation matrix and therefore reduces the computational as well as storage complexity of perturbation and distribution reconstruction. We notice that if the domain of an attribute has only $d \leq N$ distinct values, it is wise to choose $N = d$ for the perturbation matrix of that attribute. In this case, there is no need for discretization. Further discussion can be found in [14].

7 Conclusion

Inspired by the fact that the pioneering privacy-preserving decision tree mining method of [1] was flawed [2], we explored a random substitution perturbation technique for privacy-preserving decision tree mining methods. The resulting method is showed to be immune to two relevant attacks (including that of [2]). In addition, we thoroughly investigated the parameter selections that are important in guiding privacy-preserving decision tree mining practice. Systematic experiments show that our method is effective.

References

1. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM SIGMOD International Conference on Management of Data*, pages 439–450. ACM, 2000.
2. Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *IEEE International Conference on Data Mining*, 2003.
3. Dakshi Agrawal and Charu C. Aggrawal. On the design and quantification of privacy preserving data mining algorithms. In *ACM Symposium on Principles of Database Systems*, 2001.
4. Shipra Agrawal and Jayant R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *IEEE International Conference on Data Engineering*, 2005.
5. Cynthia Dwork and Kobbi Nissim. Privacy–preserving datamining on vertically partitioned databases. Microsoft Research, 2004.
6. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaching in privacy preserving data mining. In *ACM Symposium on Principles of Database Systems*, pages 211–222. ACM, 2003.

7. Y. Lindell and B. Pinkas. Privacy preserving data mining. In M. Bellare, editor, *Advances in Cryptology – Crypto 2000*, pages 36–54. Springer, 2000. Lecture Notes in Computer Science No. 1880.
8. A. Evmievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *International Conference on Knowledge Discovery and Data Mining*, 2002.
9. Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *International Conference on Very Large Data Bases*, 2002.
10. Srujana Merugu and Joydeep Ghosh. Privacy-preserving distributed clustering using generative models. In *IEEE International Conference on Data Mining*, 2003.
11. S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of American Statistical Association*, 57:622–627, 1965.
12. Leon Willenborg and Ton de Waal. *Elements of Statistical Disclosure Control.* Springer, 2001.
13. Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private informaiton from randomized data. In *ACM SIGMOD International Conference on Management of Data*, pages 37–47, 2005.
14. Jim Dowd, Shouhuai Xu, and Weining Zhang. Privacy-preserving decision tree mining based on random substitutions. Technical report, Department of Computer Science, University of Texas at San Antonio, 2005.
15. The UCI machine learning repository. http://www.ics.uci.edu/ mlearn/databases/.
16. J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.