Privacy-Preserving Decision Tree Mining Using A Random Replacement Perturbation

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 data mining has become an important topic, and many methods have been proposed for a diverse set of privacy-preserving data mining tasks. However, privacy-preserving decision tree mining pioneered by [1] still remains to be elusive. Indeed, the work of [1] was recently showed to be flawed [2], meaning that an adversary can actually recover the original data from the perturbed ones. This naturally triggers the following question: Is the data mining approach of [1] still useful despite that its specific perturbation method (called *adding noise*) is flawed? In this paper we resolve this issue by exploring a different perturbation method for privacy-preserving decision tree mining. In particular, we show that this perturbation method is immune to attacks including that of [2]. Besides, we thoroughly investigate the parameter selections that are useful in guiding privacy-preserving decision tree mining practice. Systematic experiments show that our method is effective.
Keywords: Privacy-preservation, decision tree, data mining, perturbation, matrix.

1 Introduction

Protecting privacy has become an important issue in data mining research. A fundamental requirement of privacy-preserving data mining is to protect the private 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, 5–11], which fall into either a cryptographic approach or a statistical one. The cryptographic approach [8] ensures both privacy and accuracy in a strong sense (following the well-known secure multi-party computation), but typically suffers from a low performance. The statistical approach to decision tree mining [1], association rule mining [4, 6, 7, 10], and clustering [9] is popular mainly because of its high performance. This paper falls into the category of statistical approach to privacy-preserving decision tree mining.

The first privacy-preserving decision tree mining method was pioneered by [1]. Unfortunately, this work was recently shown by [2] to be flawed. Indeed [2] gives a method whereby an adversary can actually recover the original data from the perturbed ones. Therefore, privacy-preserving decision tree mining remains to be elusive [4]. This paper was motivated by the following question: Is the data mining approach of [1] still useful despite that its specific perturbation method (called *adding noise*) is flawed? The main result of this paper is a practical and effective privacy-preserving decision tree mining method based on a *random replacement* perturbation that is immune to attacks including that of [2].

This paper makes several contributions to the topic of privacy-preserving decision tree mining. Perhaps the most important one is to show that the privacy-preserving decision tree mining framework in Figure 1, which is inspired by [1], remains to be interesting and useful. In this framework, a data owner can protect data privacy by providing the data miner only a perturbed dataset resulted from applying a perturbation method to the original dataset, and the data miner, equipped with the perturbation method, can indeed derive a decision tree that is quite accurate compared to the one derived from the original dataset. We believe that it fits well with real-life application models (e.g., it is easy to deploy in practice since existing data mining algorithms can be used without any modification).

We reaffirm the usefulness of this framework by using it with a random replacement perturbation, which is more secure than the adding noise perturbation and still ensures that a data

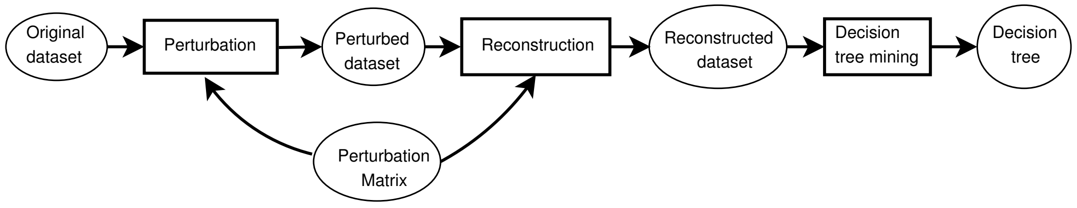


Fig. 1. A Framework of Privacy-Preserving Decision Tree Mining

miner learns accurate decision trees from the perturbed data after a reconstruction method is applied to it. In particular, we show that this perturbation method is immune to two attacks called *data-recovery* and *repeated-perturbation*, respectively.

The data-recovery attack was introduced in [2] (and further explored in [12]) to recover the original data from a given perturbed dataset. The random replacement perturbation is immune to this attack because a fundamental difference between it and the adding noise perturbation of [1]. Namely, while the adding noise draws noises from a *single* probabilistic distribution, random replacement draws noises from *many different* distributions, even if it is viewed as a special adding noise method. This suffices to show the immunity of this perturbation method against the attack of [2] because the attack relies crucially on that the noises are drawn from the same distribution.

The repeated-perturbation attack, which is newly introduced in this paper, is based on the observation that an adversary may repeatedly perturb the perturbed dataset, in hope of recovering the original data. The concern is legitimate because, if we abstract the perturbation as a function f, then it may be the case that $f(...f(f(x))) = x$, where x is an original dataset and the perturbation method f is known to the adversary (or a data miner). We rigorously show that, for any x, this attack does *not* give the adversary any extra power.

On the practical side, we thoroughly investigate the parameter selections that are important to the use of privacy-preserving decision tree mining in practice. Besides a privacy assurance metric γ, we introduce two new important metrics: one is the dimension size N of perturbation matrix, which intuitively has an impact on the accuracy of reconstructed data as well as performance, and the other is the entropy of a perturbation matrix, which can hopefully 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 serve as input parameters in selecting N. We elaborate on practical methods for selecting N, whereby 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 discuss the work related to this paper. In Section 3, we present random replacement perturbation algorithms and analyze its immunity to the data-recovery attack. In Section 4, we present the reconstruction algorithm with heuristic methods for reducing estimation error of original data distributions. In Section 5, we discuss the form of perturbation matrix and analyze the effect of perturbation matrix parameters and the immunity to the repeated-perturbation attack. In Section 6, we present results from our experiments. In Section 7, we discuss how to select perturbation matrix parameters in practice. Section 8 concludes the paper.

2 Related Work

Several works are closely related to this paper. Statistical approach to privacy-preserving decision tree mining was pioneered by [1], in which an adding noise perturbation method was explored. Unfortunately, this method was later showed to be flawed [2], because an adversary can actually recover the original data from the perturbed data. The attack of [2] (as well as its follow-up treatment [12]) is based on certain assumptions, one of which is that the added noises are drawn from the *same* probabilistic distribution. This paper was motivated by the above mentioned attacks as well as the need to explore secure privacy-preserving decision tree mining methods.

The random replacement perturbation is similar to randomization techniques in statistical disclosure control [13, 14]. However, it is based on a different privacy measure called ρ_1-to-ρ_2 privacy breaching [6]. Roughly, a perturbation technique prevents a ρ_1-to-ρ_2 privacy breaching if the adversary has an a prior probability of some private information in the original data no more than ρ_1, she can derive a posterior probability of the private information no more than ρ_2. The work of [6] was generalized and thoroughly investigated by [4] to introduce a special type of perturbation matrix called the γ-diagonal matrix, which is adapted and extended in the current paper.

Both [6] and [4] are concentrated on privacy-preserving association rule mining. It was explicitly considered as an open problem in [4] to extend the results therein to other data mining tasks such as decision tree mining. In this paper, we concentrate on privacy-preserving decision tree mining. While our perturbation method may have inherited the privacy assurance guarantee of [6], the aforementioned *data-recovery* and *repeated-perturbation* attacks studied in this paper were not accommodated in the model of [6].

While our perturbation matrix is based on [4], we introduce a novel error-reduction technique in our data reconstruction method, which is guaranteed to achieve a strictly better accuracy (see Section 4.3 for a theoretic treatment and Section 6.1 for experimental results). In addition, we investigate the effect of perturbation matrix size, which was not considered in either [6] or [4].

3 Random Replacement Perturbation and Analysis

In this section we explain the basic ideas underlying the random replacement perturbation method for both discrete and continuous attribute domains, while deferring many details (e.g., how a perturbation matrix should be selected) to the following sections. Nevertheless, we thoroughly explore *why the random replacement is fundamentally different from the adding noise* [1], which implies that the perturbation method employed in this paper is immune to the attacks explored in [2]. In the following, we assume that data records have attributes $A_1, \dots, A_q$, 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.

3.1 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 Replacement Perturbation (RRP)

```
Input: a dataset $\mathcal{O}$ of $n$ records, an attribute $A$ with domain $\mathcal{U} = \{u_1, \dots, u_N\}$, 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 $1 \le h \le N$ such that $\sum_{i=1}^{h-1} m_{i,k} < r \le \sum_{i=1}^{h} m_{i,k}$;
    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 5.

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 RRP. 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 RRP is $O(n \cdot N)$. This time complexity can be improved if, instead of using $\mathbf{M}$, we use a cumulative probability matrix $\mathbf{M}'$ with general element $m'_{h,k} = \sum_{j=1}^{h} m_{h,k} = F_k(h)$ to generate perturbed values. In this case, the random number r is generated as before, but we now use a binary search in column k of $\mathbf{M}'$ to find the row index that corresponds to r. When the binary search completes, we will find some $1 \leq h \leq N$, such that $m'_{h-1,k} < r \leq m'_{h,k}$, and therefore choose u_h as the perturbed value of u_k. The time complexity of this modified algorithm will be $O(n \log_2 N)$. However, both the time and the space complexities of generating $\mathbf{M}'$ from $\mathbf{M}$ are $O(N^2)$.

3.2 Perturb a Continuous-Valued Attribute

One way to apply RRP 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 5.2 and 7.

3.3 Why Is Random Replacement Fundamentally Different From Adding Noise?

The adding noise perturbation method introduced in [1] is subject to what we call *data-recovery* attacks [2, 12], 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 replacement 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 replacement perturbation. Thus, the random replacement perturbation method is immune to the attacks explored in [2, 12].

The basic idea here is to view the random replacement 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 replacement 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 design of the random replacement 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 replacement 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 replacement perturbation. The following result 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* $\Pr[r \mid o_i] \neq \Pr[r \mid o_j]$.

Proof. Without loss of generality, assume $h < k$. (It is easy to generalize the proof for $h > k$.) Notice that $a \leq h < k \leq b$. Let $o_i = h$ and $r = k - o_i = k - h$. Clearly, $o_i \in [a, b]$, $r > 0$, $k = o_i + r$ and $a \leq k \leq b$. Thus, $\Pr[r \mid o_i] = m_{k,h} > 0$. Now let $o_j = b$, which is in $[a, b]$. Since $r > 0$, we have $o_j + r = b + r > b$. As a result, $0 = \Pr[r \mid o_j] \neq \Pr[r \mid o_i] > 0$. □

In fact, Theorem 1 can be shown to be also true for any o_j satisfying $b - |k - h| < o_j \leq b$ (or $a \leq o_j < a + |k - h|$, if $h > k$). Intuitively when perturbing a dataset, the noise distribution used to perturb a data generally depends on the data itself, and this distribution will be different from that used for another data, if both data can be replaced by a value different from themselves. It follows that if every domain value can be perturbed into every other values, the noise distribution for one original data will be different from that for any other original data. Furthermore, without knowing the original data, the adversary could not know exactly which distribution a noise is drawn from.

4 Generating Reconstructed Dataset

4.1 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 a 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 Section4.2 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

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 $\mathcal{R}$;
```

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 on these attributes one at a time.

4.2 Estimating Original Data Distributions

A simple method for estimating original data distribution is given in [4], which is briefly described in the following. 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

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

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.

A problem of this method is that since $\mathbf{y}$ is an estimate of $E(Y)$, $\hat{X}$ often has 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 cause the failure of the reconstruction. To resolve this problem, we describe a correction method in the next section.

4.3 Reducing Estimation Errors

To 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 $\hat{X} = X$, $||\hat{X}||_1 = n$.*

Proof. Since by definition, $||\hat{X}||_1 = \sum_{i=1}^{N} |\hat{X}_i| \geq \sum_{i=1}^{N} \hat{X}_i$, all we need to show is that for any set of n perturbed data, $\sum_{i=1}^{N} \hat{X}_i = ||X||_1 = ||Y||_1 = n$.

Recall that $\mathcal{P}$ is the set of perturbed data, $\mathcal{O}$ is the corresponding set of original data, and X and Y are the sets of counts of domain values in the original and the perturbed datasets, respectively. Since each original data is perturbed into one perturbed data, and by definition all counts are non-negative, we have $||X||_1 = ||Y||_1 = n$.

By definition of $\hat{X}$, we have $Y = \mathbf{M}\hat{X}$ and therefore, $Y_i = \sum_{j=1}^{N} m_{ij}\hat{X}_j$. This leads to

$$\begin{aligned} n &= \sum_{i=i}^{N} Y_i = \sum_{i=1}^{N} |Y_i| = ||Y||_1 = \sum_{i=1}^{N}\sum_{j=1}^{N} m_{ij}\hat{X}_j \\ &= \sum_{j=1}^{N}(\sum_{i=1}^{N} m_{ij})\hat{X}_j = \sum_{j=1}^{N} \hat{X}_j \leq \sum_{j=1}^{N} |\hat{X}_j| \end{aligned}$$

where the last equality is because that the sum of transition probabilities of each column of the perturbation matrix is 1. □

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.

The following Proposition indicates that constraint **C2** can be used not only to identify but also to reduce an estimation error in $\hat{X}$. It leads to a useful heuristic that has been adopted in our experiments.

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

Proof. Assume some $\hat{X}_i < 0$. By definition, this means that $X_i + \delta_i < 0$. Since $X_i \geq 0$, it must be true that $\delta_i < 0$ and $|\delta_i| > X_i$. By setting $\hat{X}_i$ to zero, we have $\hat{X}'_i = 0 = X_i + \delta'_i$, which implies $X_i = -\delta'_i$, that is, $|\delta'_i| = X_i < |\delta_i|$. Thus the error is reduced. □

5 Perturbation Matrix and Analysis

5.1 Perturbation Matrix: Our Starting Point

An ideal perturbation matrix should provide a privacy guarantee that is acceptable to the data owner and be optimized for the best quality of data mining results, in terms of classification accuracy. In this section, we describe the γ-diagonal matrix, which was adopted from [4] for our purpose.

Privacy Requirement The basic idea behind the privacy measure of [6] is to assume that the adversary knows the prior probability that a property of private information holds on an original data and is able to derive the posterior probability that the property holds on the original data given the perturbed data. Therefore, the leak of private information can be measured by the difference between the prior and the posterior probabilities. A perturbation technique provides a (ρ_1, ρ_2) privacy guarantee [6, 4] if for any property of private information that has a prior probability less than ρ_1 the posterior probability is less than ρ_2. To support a (ρ_1, ρ_2) privacy guarantee, the perturbation matrix must satisfy the following condition [4]: for $\forall k_1, k_2, h \in \{1, \ldots, N\}$,

$$\frac{m_{h,k_1}}{m_{h,k_2}} \leq \gamma \leq \frac{\rho_2(1-\rho_1)}{\rho_1(1-\rho_2)}$$

where $\gamma = \max_{1 \leq r \leq N}(M_r/m_r)$ is called *amplification factor* [6] of the perturbation matrix, M_r and m_r are respectively the largest and the smallest elements in row r of the perturbation matrix.

Accuracy Requirement Since the estimation of original data distribution needs to compute Eq. (1), the estimation error is used to measure the quality of the perturbation matrix in [4]. There are two sources of estimation errors in Eq. (1): the sensitivity of the problem as measured by the *condition number* of the perturbation matrix and the deviation of $\mathbf{y}$ from $E(Y)$. One way to reduce the estimation error is to minimize the condition number of the perturbation matrix.

Gamma Diagonal Perturbation Matrix According to the aforementioned privacy and accuracy requirements, [4] showed that for a given γ the optimal perturbation matrix is the gamma diagonal matrix $\mathbf{M} = x\mathbf{G}$, where $x = \frac{1}{\gamma+N-1}$, and

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

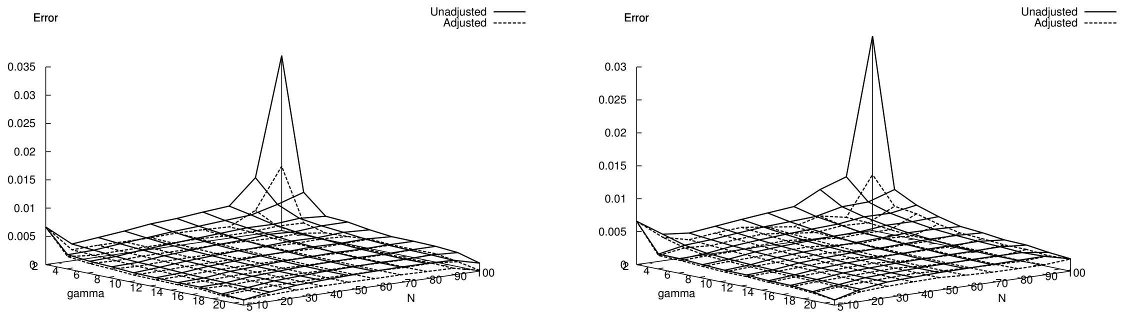
which guarantees γ and has a minimum condition number.

5.2 An Analysis of Perturbation Matrix

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, which means $E(Y) = X = Y$. 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 reduced and off-diagonal elements $\frac{1}{\gamma+N-1}$ will be increased. 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 in to 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.

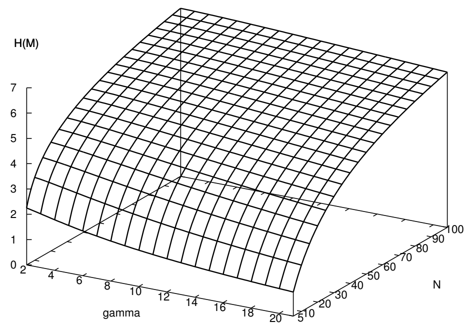


Fig. 3. Entropy of some perturbation matrices

The Entropy of Perturbation Matrix So far, we analyzed how each of γ and N individually affects privacy and accuracy. But, how does the combination of γ and N affect these measures? This is important particularly when 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 7, 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 replacement perturbation described in Section 3 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.

Dataset	#TrainRec	#TestRec	Ori. Accu	#Attr	#classes
Synth1	100K	5K	100%	9	2
Synth2	100K	5K	89.36%	9	2
Synth3	100K	5K	100%	9	2
Synth4	100K	5K	99.22%	9	2
Synth5	100K	5K	98.48%	9	2
Pendigits	7494	3498	92.0526%	16	10
Spam	3101	1500	91.7333%	57	2

Table 1. Datasets Used to Test Decision Tree Accuracy

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 $m_{i,j}^t = \Pr[u_j \xrightarrow{t} u_i]$, 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 $t > 1$, $m_{i,i}^t > m_{i,i}^{t+1}$ and $m_{i,j}^t < m_{i,j}^{t+1}$, and $\lim_{t\to\infty} m_{i,j}^t = \frac{1}{N}$ for $1 \leq i,j \leq N$.*

The proof is fairly involved, and thus is deferred to the Appendix.

6 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.

6.1 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, … , 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 Figure 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 *set-scale* heuristic is able to reduce estimation error by 50%.

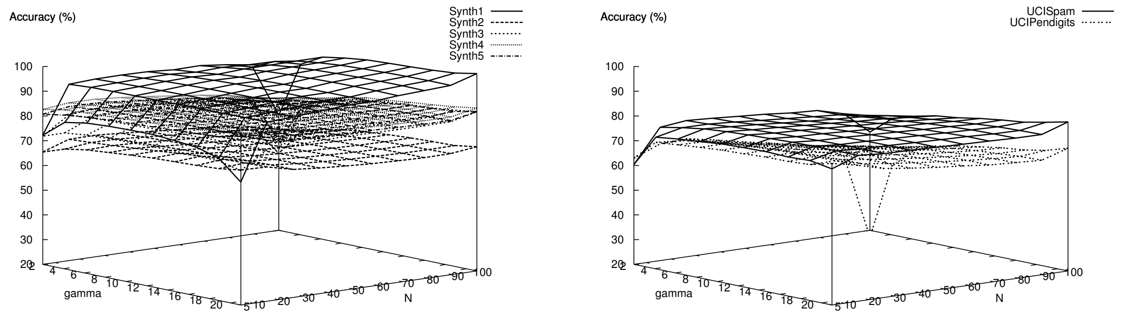
6.2 Decision Tree Accuracy

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.

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. Table 1 shows some properties of these datasets. The decision tree mining algorithm used is a version of C4.5 [16] implemented in Java. For reference, Table 1 also lists the accuracy of decision trees learned directly from the original training set.

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 listed in Table 1 under the corresponding (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).

Fig. 5 shows the relationships between entropy and the accuracy for the datasets we considered. Here, the accuracy for each dataset is plotted as a line in which dots correspond different



(a) Synthetic Datasets (b) UCI Datasets

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

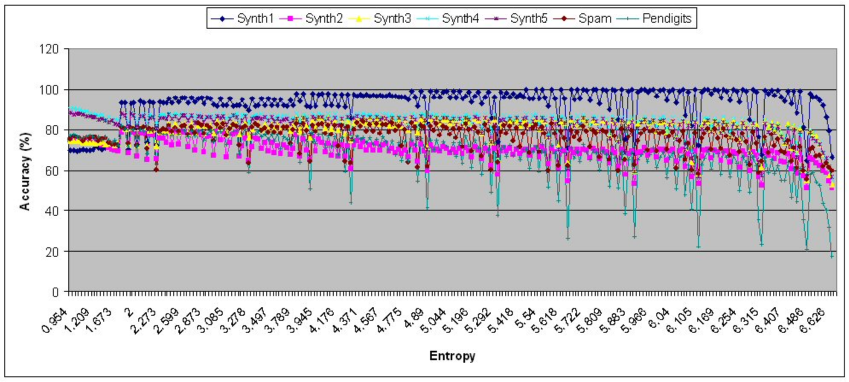


Fig. 5. Entropy vs. Accuracy (all datasets)

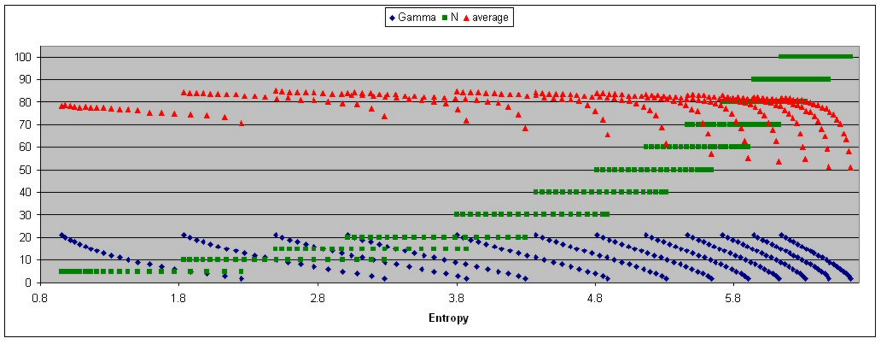


Fig. 6. Entropy vs. γ, N, and Average Accuracy . Notice that the vertical axis has **three** meanings: for γ (represented by diamonds), it is integer between 2 and 21 (i.e., on the bottom of the chart); for N (represented by squares), it is integer between 5 and 100 (i.e., scattered from bottom left 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.

combination of γ and N of perturbation matrices. The height of the lines may be affected by the nature of the corresponding datasets (since the accuracies for original datasets also vary among datasets). Despite some variation of shapes and a lack of detail due to the clutter of lines, Fig. 5 still shows a strikingly similar pattern. To better illustrate this pattern, we take the average of the 7 accuracies at 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. 6, 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. 6, 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.

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. 6 are lower than those in some other intervals of entropy.

7 A Guide to Select Parameters in Practice: Putting Pieces Together

The results described in Section 6.2 clearly indicate that both N and γ are important parameters. In this section, we address the question "How should a data owner determine γ and N, so that both desirable accuracy and privacy can be achieved while leading to as efficient as possible solutions?"

It would be nice if there is a closed formula that expresses decision tree accuracy as a function of the entropy of perturbation matrix, which can be used by data owners to choose optimal N and γ. Unfortunately, due to the nature of decision tree mining, it is seemingly very difficult, if not impossible, to obtain such a formula. As an alternative, we suggest to use Fig. 6 (or any such figure obtained from representative domain datasets) as a useful guide for data owners to determine the parameters of a perturbation matrix. We elaborate on two methods of using this figure to select a good design.

For a given 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 larger γ provides a poor protection of privacy and small γ reduces the accuracy, we suggest that a reasonable range of γ should be $5 \leq \gamma \leq 12$.

7.1 Find N For Given γ

For a given set of γ (say $5 \leq \gamma \leq 12$), the data owner can find the highest accuracy from Fig. 6 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 chosen as the best design, 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. This also means that there is no need to conduct discretization.

7.2 Find N For Given Accuracy And γ

If in addition to a desired γ, the data owner also wants to guarantee a certain level of accuracy of decision trees that can be learned from her dataset, she can first identify average accuracies in Fig. 6 that are equal to or more than the desired accuracy (if no such accuracy exists, she has to instead settle with the highest accuracy in the figure). For these identified accuracies, she will find the corresponding combinations of γ and N. The combination whose γ is no greater than the desired γ and whose N is the smallest among all combinations will be selected. If more than one combination qualify, she will choose the combination with the smallest γ, so that, a stronger privacy assurance guarantee can be achieved without downgrading accuracy or increasing computational complexity.

8 Conclusion

Inspired by the fact that the pioneering privacy-preserving decision tree mining method of [1] was flawed [2], we explored a random replacement perturbation method for privacy-preserving decision tree mining. This perturbation method is showed to be immune to attacks including that of [2]. Besides, 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. 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.
8. 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.
9. Srujana Merugu and Joydeep Ghosh. Privacy-preserving distributed clustering using generative models. In *IEEE International Conference on Data Mining*, 2003.
10. Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *International Conference on Very Large Data Bases*, 2002.
11. V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 33(1):50–57, 2004.
12. 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.
13. S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of American Statistical Association*, 57:622–627, 1965.
14. Leon Willenborg and Ton de Waal. *Elements of Statistical Disclosure Control*. Springer, 2001.
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.

Appendix

To prove Theorem 2, we need the following Lemma.

Lemma 2. *Let* $\mathbf{O}$ *be an* $N \times N$ *matrix of* 1s*, that is, every element is* 1*, and* $\lambda = \gamma - 1$*. Then,* $\mathbf{G}^t = \lambda^{t-1}\mathbf{G} + (\lambda+N)^{(t-1)} \sum_{i=0}^{t-2} (\frac{\lambda}{\lambda+N})^i \cdot \mathbf{O}$.

Proof. We prove it by induction.

(***Basis***). It is easy to verify in the case of $\mathbf{G}^2 = \left(g_{i,j}^2\right)$, it holds that

$$g_{i,j}^2 = \begin{cases} \gamma^2 + N - 1, \text{ if } i = j; \\ 2\gamma + N - 2, \textit{ otherwise}. \end{cases}$$

With some elementary algebraic manipulation, we have $g_{i,i}^2 = \lambda\gamma + x^{-1}$, and $g_{i,j}^2 = \lambda + x^{-1}$. Substitute these values in $\mathbf{G}^2$ and decompose the matrix according to linear algebra, we have $\mathbf{G}^2 = \lambda \cdot \mathbf{G} + x^{-1} \cdot \mathbf{O} = \lambda\mathbf{G} + (\lambda+N)^1 (\frac{\lambda}{\lambda+N})^0 \cdot \mathbf{O}$.

(***Hypothesis***). Assume for $2 \le t \le T$-1, $\mathbf{G}^t = \lambda^{t-1}\mathbf{G} + (\lambda+N)^{(t-1)} \sum_{i=0}^{t-2} (\frac{\lambda}{\lambda+N})^i \cdot \mathbf{O}$.

(***Induction***). Let $t = T$.

$$\begin{aligned}
\mathbf{G}^T &= \mathbf{G}^{T-1} \cdot \mathbf{G} \\
&= \left(\lambda^{T-2}\mathbf{G} + (\lambda+N)^{(T-2)} \sum_{i=0}^{T-3} (\frac{\lambda}{\lambda+N})^i \mathbf{O} \right) \mathbf{G} \\
&= \lambda^{T-2}\mathbf{G}^2 + (\lambda+N)^{(T-2)} \sum_{i=0}^{T-3} (\frac{\lambda}{\lambda+N})^i \mathbf{O}\mathbf{G} \\
&= \lambda^{T-2}\lambda\mathbf{G} + \lambda^{T-2}(\lambda+N)\mathbf{O} + \\
&\quad (\lambda+N)^{(T-2)} \sum_{i=0}^{T-3} (\frac{\lambda}{\lambda+N})^i \mathbf{O}\mathbf{G} \\
&= \lambda^{T-1}\mathbf{G} + \lambda^{T-2}(\lambda+N)\mathbf{O} + \qquad (2) \\
&\quad (\lambda+N)^{(T-2)} \sum_{i=0}^{T-3} (\frac{\lambda}{\lambda+N})^i \cdot (\lambda+N)\mathbf{O} \\
&= \lambda^{T-1}\mathbf{G} + (\lambda+N)^{T-1} (\frac{\lambda}{\lambda+N})^{T-2}\mathbf{O} + \\
&\quad (\lambda+N)^{(T-1)} \sum_{i=0}^{T-3} (\frac{\lambda}{\lambda+N})^i \mathbf{O} \\
&= \lambda^{T-1}\mathbf{G} + (\lambda+N)^{(T-1)} \sum_{i=0}^{T-2} (\frac{\lambda}{\lambda+N})^i \mathbf{O}
\end{aligned}$$

Where Step (2) is obtained because $\mathbf{O} \cdot \mathbf{G} = (\lambda+N) \cdot \mathbf{O}$. □

We now can prove Theorem 2.

Proof. (Theorem 2) By definition, $\mathbf{M} = x \cdot \mathbf{G} = (\lambda+N)^{-1} \cdot \mathbf{G}$ and $\mathbf{M}^t = (\lambda+N)^{-t} \cdot \mathbf{G}^t$. By Lemma 2,

$$\mathbf{M}^t = (\lambda+N)^{-t} \left(\lambda^{t-1}\mathbf{G} + (\lambda+N)^{(t-1)} \sum_{i=0}^{t-2} (\frac{\lambda}{\lambda+N})^i \cdot \mathbf{O} \right)$$

Accordingly, the diagonal elements of $\mathbf{M}^t$ are

$$\begin{aligned}
m_{i,i}^t &= (\lambda+N)^{-t}\lambda^{t-1}\gamma + (\lambda+N)^{-1}\sum_{i=0}^{t-2}(\frac{\lambda}{\lambda+N})^i \\
&= (\lambda+N)^{-t}\lambda^{t-1}(\lambda+1) + (\lambda+N)^{-1}\sum_{i=0}^{t-2}(\frac{\lambda}{\lambda+N})^i \\
&= (\frac{\lambda}{\lambda+N})^t + (\lambda+N)^{-1}(\frac{\lambda}{\lambda+N})^{t-1} + \\
&\quad (\lambda+N)^{-1}\sum_{i=0}^{t-2}(\frac{\lambda}{\lambda+N})^i \\
&= (\frac{\lambda}{\lambda+N})^t + (\lambda+N)^{-1}\sum_{i=0}^{t-1}(\frac{\lambda}{\lambda+N})^i
\end{aligned}$$

and similarly, the non-diagonal elements are $m_{i,j}^t = (\lambda+N)^{-1}\sum_{i=0}^{t-1}(\frac{\lambda}{\lambda+N})^i$. Since $m_{i,i}^{t+1} - m_{i,i}^t = \frac{-(N-1)}{\lambda+N}\cdot(\frac{\lambda}{\lambda+N})^t < 0$, we have $m_{i,i}^{t+1} < m_{i,i}^t$. Similarly, since $m_{i,j}^{t+1} - m_{i,j}^t = \frac{\lambda^t}{(\lambda+N)^{t+1}} > 0$, $m_{i,j}^{t+1} > m_{i,j}^t$, for $i \neq j$. Let $w = \frac{\lambda+N}{\lambda}$. Since $\lim_{t\to\infty}(\frac{\lambda}{\lambda+N})^t = 0$, it is clear that

$$\begin{aligned}
\lim_{t\to\infty} m_{i,j}^t &= \frac{1}{\lambda+N}\sum_{i=0}^{\infty}(\frac{\lambda}{\lambda+N})^i = \frac{1}{\lambda+N}\sum_{i=0}^{\infty}(\frac{1}{w})^i \\
&= \frac{1}{\lambda+N}\cdot\frac{w}{w-1} = \frac{\lambda+N}{(\lambda+N)N} = \frac{1}{N}
\end{aligned}$$