LEARNING FROM PERTURBED DATA FOR PRIVACY-PRESERVING DATA MINING

By

JIANJIE MA

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

AUGUST 2006

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of JIANJIE MA find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor, Dr. Krishnamoorthy Sivakumar, for his invaluable guidance and consistent support during my five years graduate study in Washington State University. Dr. Sivakumar has been a wonderful advisor. His broad knowledge and enthusiasm for researches deeply impressed me.

I would like to extend my sincere appreciation to Dr. Murali Medidi and Dr. Benjamin Belzer. Their lectures helped me to build a solid research background. Special thanks to Dr. Ali Saberi whose encouragement gave me confidence. Dr. Saberi's lecture notes in several classes gave me strict training in thinking.

The work in this dissertation was partially supported by the United States National Science Foundation Grant IIS-0350533.

I also wish to thank the staff of the school of Electrical Engineering and Computer Science, especially the graduate secretary, Ms. Ruby Young, for their help and support.

Finally, I would like to thank my parents and my wife for the endless love and unconditional support they provided me.

LEARNING FROM PERTURBED DATA FOR

PRIVACY-PRESERVING DATA MINING

Abstract

by Jianjie Ma, Ph.D.
Washington State University
August 2006

Chair: Krishnamoorthy Sivakumar

In this dissertation, we concentrate on privacy-preserving data mining (PPDM) using post randomization (PRAM) techniques from distributed data. PRAM provides a general framework for randomization of categorical data. We estimate frequency counts from the randomized data by using moment estimation method or maximum likelihood estimation method. Normal approximation for the distribution of the estimator is also given. The variance of each estimator of frequency count is inversely proportional to the sample size in order.

Privacy preserved by using PRAM is quantified by γ-amplification and probabilistic K-anonymity. Randomization causes some information loss, which can be quantified by metrics like distance between two distributions. Another important aspect of information loss is independence loss, which is also discussed in this dissertation.

The proposed method is applied to Bayesian network learning. We consider both structure and parameter learning. For structure learning, we face the familiar extra-link problem since estimation errors tend to break the conditional independence among the variables. We propose modifications to score functions used for Bayesian network learning, to solve this problem.

For continuous-valued data, an MGAS (Modified Agglomerative Scheme) discretization technique based on Hierarchical Clustering is proposed in this dissertation. MGAS technique discretizes numerical variables and indirectly enhances privacy. This technique has been applied to learn linear classifiers from randomized data for privacy consideration. Linear classifier is a model based on cost optimization. Instead of using the original cost function, the expectation of the cost based on the randomized data is optimized.

Finally, the proposed technique is applied to both association rule mining and decision tree learning. The supports of the K-itemsets in association rule mining can be estimated from the randomized data. By randomizing several items simultaneously, more simultaneous privacy breaches are limited by reducing simultaneous γ-amplification. Privacy-preserving decision tree learning is accomplished by estimating frequency counts necessary for calculating information gain from randomized data.

Our experiments show that post randomization is an efficient, flexible and easy-to-use method to do Privacy-preserving data mining. Experimental results with different levels of randomization and different sample sizes show that this method produces an accurate model, even with a large level of randomization.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

Explosive growth in network, storage, and processor technologies has led to the creation of ultra large databases that record unprecedented amount of transactional information. In tandem with this dramatic increase in digital data, concerns about informational privacy have emerged globally. Privacy issues are further exacerbated now that the World Wide Web makes it easy for the new data to be automatically collected and added to databases. Data mining, with its promise to efficiently discover valuable non-obvious information from large databases, is particularly sensitive to privacy concerns. In recent years, data mining has also endeavored to become compatible with privacy. Fruitful research has been produced by different researchers on the topic of privacy-preserving data mining (PPDM). PPDM deals with the problem of learning accurate models over aggregate data, while protecting privacy at the level of individual records.

In this dissertation, we propose to use post randomization (PRAM) techniques for PPDM. PRAM was originally proposed as a statistical disclosure control technique and is closely related to randomized response techniques in collecting private or confidential data from individuals. We explore the possibility of using PRAM techniques for PPDM. The analysis of the proposed techniques and the proposed algorithm can be directly applied to analysis or mining from data collected using randomized response techniques.

Before we describe our proposed work, we first offer some background information in the following sections. In section 1.1, we briefly introduce distributed data mining, which forms the basis for some of our proposed methods. Different aspects of privacy are introduced in section 1.2. Some methods for PPDM proposed in literature are reviewed in section 1.3.

1.1 Distributed Data Mining

Distributed data mining deals with the problem of finding patterns in an environment where the data and/or computational units are distributed. A typical application domain of distributed data mining either has inherently distributed data sources or a centralized data partitioned into different blocks. The data may be distributed in the following three ways:

a. Homogeneously distributed databases

In this setting, the database is horizontally distributed into several parties. Every party has records with the same set of features. One extreme case is a "fully" homogeneously distributed database in which each individual record is held by its owner.

b. Heterogeneously distributed databases

In this setting, the database is vertically partitioned. Every party owns a vertical piece of every record in the database; i.e., it holds records for a subset of the features.

c. Arbitrarily partitioned databases

In this setting, different features of different items can be owned by any party.

The above three distributed cases are illustrated in figure 1.1, where a database is distributed into two partitions.

The problem of learning from distributed data can be summarized as follows: Given a data set D, which is distributed, a hypothesis class H, and a performance criterion C, the learning algorithm L outputs a hypothesis $h \in H$ that optimizes C. For example, in pattern classification applications, h is a classifier (e.g., a decision tree, a support vector machine, etc.). In clustering, h is a set of clusters. The data D typically consists of a set of training examples. Each training example is an ordered tuple of attribute values, where

one of the attributes corresponds to a class label (if classification is involved) and the remaining attributes represent inputs. The goal of learning is to produce a hypothesis that optimizes the performance criterion of minimizing some function of the error (on the training data) and the complexity of the hypothesis.

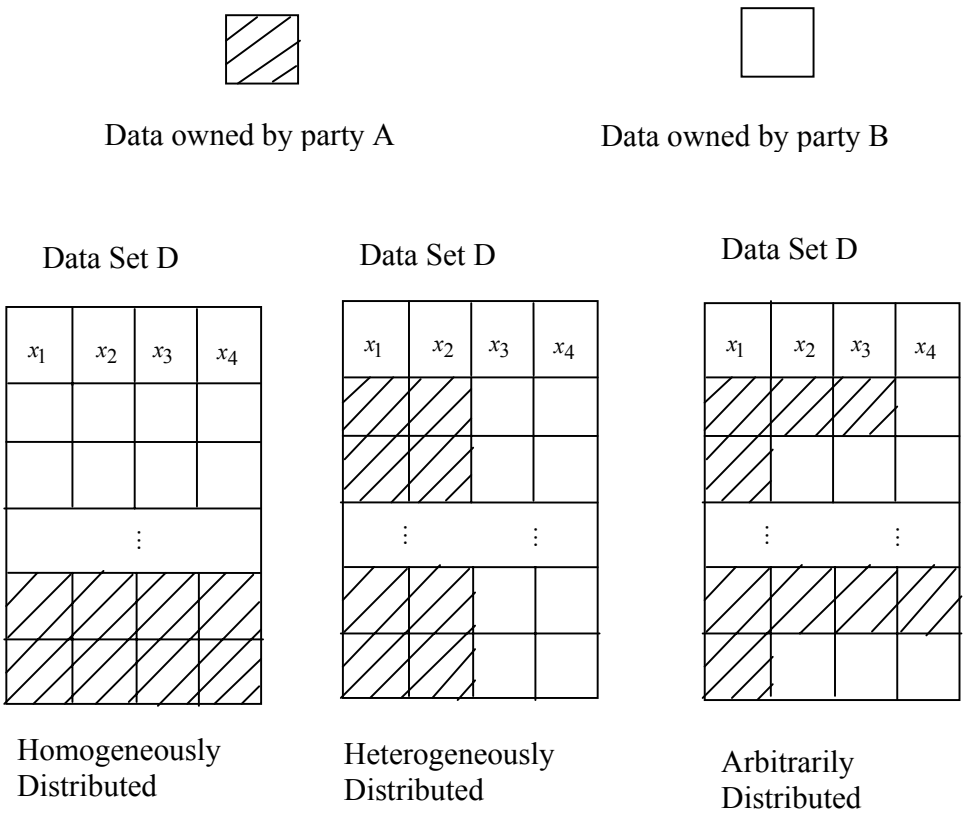


Figure 1.1: Distributed Databases

Some illustrative examples of distributed data mining applications are as follows:

Example 1: Consider an epidemiologist, studying the spread of hepatitis-C in the US. She is interested in detecting any underlying relation of the emergence of hepatitis-C in the US with weather pattern.

Example 2: Two major financial organizations want to cooperate for preventing fraudulent intrusion into their computing system. They need to share data patterns relevant to fraudulent intrusion.

Example 3: Several large corporations want to analyze customer transaction records for developing a successful business strategy for mutual benefits.

Example 4: A research center wants to find the correlation of DNA sequence and specific diseases. They have DNA sequences of a group of people while personal medical histories are stored in databases of hospitals.

Many distributed data mining techniques have been developed to save communication overhead and offer better scalability, with minimal communication of possibly private data.

Mining from heterogeneous data constitutes an important class of distributed data mining problems. This issue is discussed in [PB95] from the perspective of inductive bias. The WoRLD system [AKPB97] addressed the problem of concept learning from heterogeneous sites by developing an "activation spreading" approach that is based on a first order statistical estimation of the underlying distribution. A novel approach to learn association rules from heterogeneous tables is proposed in [CS99]. This approach exploits the foreign key relationships for the case of a star schema to develop decentralized algorithms that execute concurrently on the separate tables, and subsequently merge the results. An order statistics-based technique for combining high-variance models generated from heterogeneous sites is proposed in [TG00a]. In particular, Kargupta and his colleagues [KPHJ00] also considered the heterogeneous case and proposed the collective framework to address data analysis for heterogeneous

environments. Chen et al. [CSK03, CSK04] proposed collective Bayesian network learning from heterogeneous data.

Our work and other works in PPDM are an attempt to solve another important problem in the distributed data mining, which is the privacy issue. The algorithms discussed above in distributed data mining might be very efficient both in computation and communication, but obviously they were not designed with privacy in mind.

1.2 Privacy Issues in Data Mining

As introduced in previous section, data mining technology has been developed with the goal of providing tools for automatically and intelligently transforming large amount of data into knowledge relevant to users. The extracted knowledge, often expressed in the form of e.g., association rule, Bayesian networks, classification rules, decision tree or clustering, allows people to find interesting patterns and rules buried in the data, which can be used to help decision making process. When a complete data set is available, various statistical, machine learning and modeling techniques can be applied to analyze the data. In many contexts, data are distributed across different sites. Traditionally, the data warehousing approach has been used to mine distributed databases. It requires that data from all the participating sites are collected at a centralized warehouse. However, many data owners may be reluctant to share their data with others due to privacy and confidentiality concerns, which impedes performing mutually beneficial data mining tasks. Also in many contexts, data owners are often not data miners and data owners will often be unwilling to disclose their data to the data miner

due to privacy concerns. For examples, consider the distributed data mining examples given in section 1.1 further:

Example 1: The epidemiologist has access to a large environmental and weather database and the center for disease control (CDC) has a large hepatitis-C database. However, CDC may not be able to provide their databases to her due to privacy consideration.

Example 2: The companies may not be willing to share their data due to proprietary reasons.

Example 3: Though there are mutual benefits, those corporations will be unwilling to share their customer information with others since it involves commercial competition.

Example 4: Though the research center has DNA sequences of a group of people, personal medical histories are often stored in databases of hospitals. Hospitals cannot provide any information about individual medical history to the research center due to privacy regulations.

In many situations, individual variables are inherently sensitive and people are reluctant to divulge them to a data miner. In some cases, although the variables themselves are not sensitive, it will help identify confidential information through links between databases. Those variables that can help identify confidential information should also be paid attention to. Re-identification of confidential information through links is widely discussed in [Swe02a, Swe02b]. The following is the example directly from [Swe02a] to illustrate re-identification by linking:

The National Association of Health Data Organizations (NAHDO) reported that 37 states in the USA have legislative mandates to collect hospital level data and that 17 states have started collecting ambulatory care data from hospitals, physicians offices, clinics, and so forth. The leftmost circle in Figure 1.2 contains a subset of the fields of information, or attributes, that NAHDO recommends these states collect; these attributes include the patient's ZIP code, birth date, gender, and ethnicity. In Massachusetts, the Group Insurance Commission (GIC) is responsible for purchasing health insurance for state employees. GIC collected patients' specific data with nearly one hundred attributes per encounter along the lines of those shown in the leftmost circle of Figure 1.2 for approximately 135,000 state employees and their families. Because the data were believed to be anonymous, GIC gave a copy of the data to researchers and sold a copy to industry. For twenty dollars, the author of [Swe02a] purchased the voter registration list for Cambridge Massachusetts and received the information on two diskettes. The rightmost circle in Figure 1.2 shows that these data included the name, address, ZIP code, birth date, and gender of each voter. This information can be linked using ZIP code, birth date and gender to the medical information, thereby linking diagnosis, procedures, and medications to particular named individuals. For example, William Weld was governor of Massachusetts at that time and his medical records were in the GIC data. Governor Weld lived in Cambridge Massachusetts. According to the Cambridge Voter list, six people had his particular birth date; only three of them were men; and, he was the only one in his 5-digit ZIP code.

The example above provides a demonstration of re-identification by directly linking (or "matching") on shared attributes. Though the problem is not so dramatic in

most cases, the above example gives us an idea of how privacy breaches can happen if only those inherent sensitive variables are considered and that privacy problem is far from solved by only removing personal identification numbers of data records.

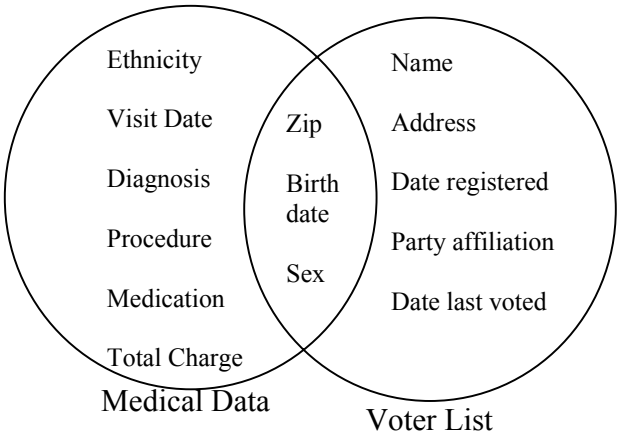


Figure 1.2 : Example of re-identification by links

Due to privacy and confidentiality concerns discussed above, application of data mining technologies is largely constrained. PPDM has emerged to address the privacy issues in data mining. Embedding privacy into data mining has been an active and fruitful research area. Several data mining techniques, incorporating privacy protection mechanisms, have been proposed based on different approaches.

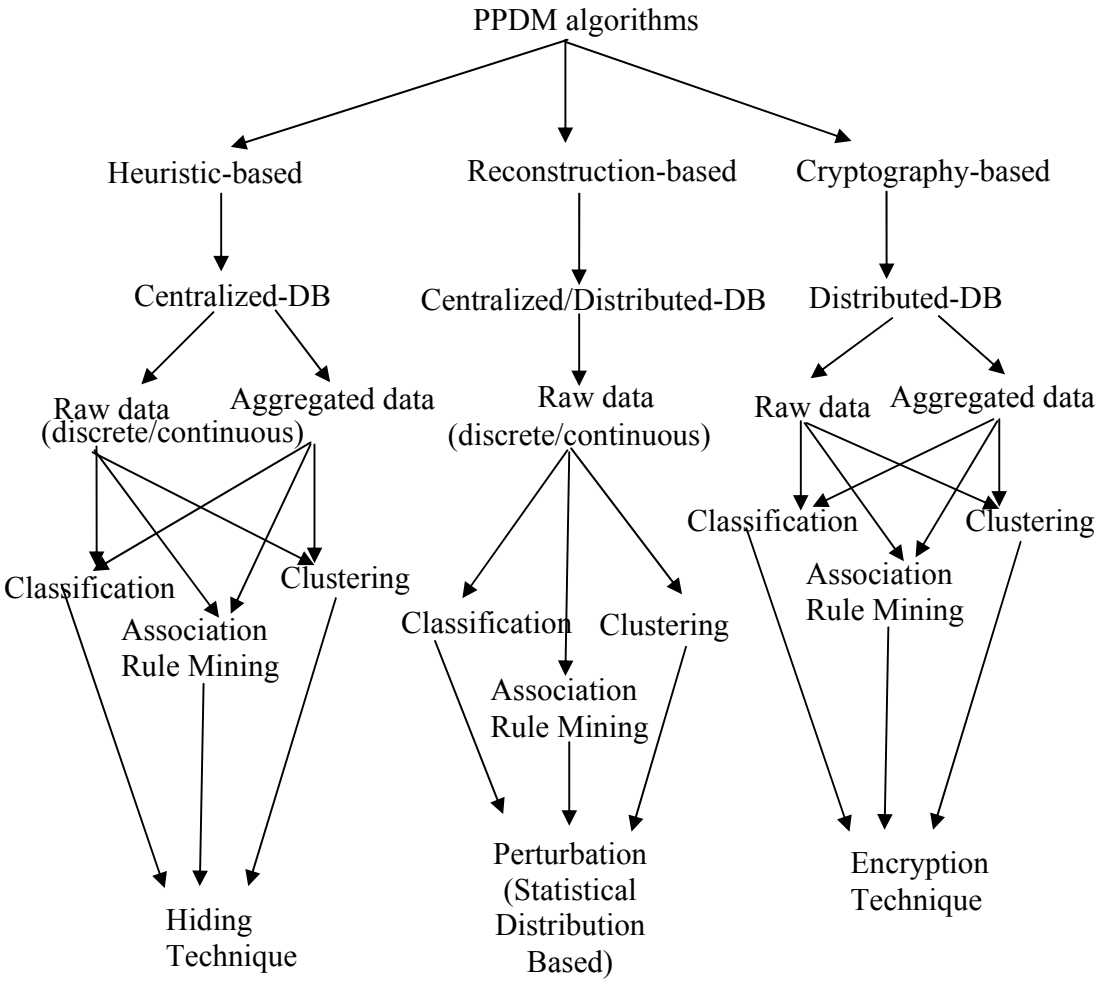
Another privacy issue concerning data mining is from the release of databases. Recent advances in data mining techniques and related applications have, on the other hand, increased the security risks that one may incur when releasing data. The elicitation of knowledge that can be obtained using such techniques has been the focus of the knowledge discovery in databases researcher's effort for years and by now, it is a well understood problem. However the impact on information confidentiality originated by these techniques has not been considered until recently. The process of uncovering

hidden patterns from large databases has been already indicated as a threat to database security. One aspect of PPDM has been focused on preventing data mining techniques from discovering sensitive knowledge which is not even known to the database owners before they are released. This aspect of PPDM has also been called rule hiding.

1.3 PPDM

Recent research in the area of PPDM has devoted much effort to determine a trade-off between privacy and the need for knowledge discovery, which is crucial in order to improve decision-making processes and other human activities. In this section, we first present a taxonomy of the PPDM algorithms that have been proposed based on a classification presented in Verykios et al. [VBFP04, BN05]. We then present a brief review of the major work in this area. Verykios et al. [VBFP04] analyze the state-of-the-art in the area of PPDM, classifying the proposed privacy preservation techniques according to five different dimensions: (i) data distribution (centralized or distributed); (ii) the modification applied to the data (encryption, perturbation, generalization, and so on) in order to sanitize them; (iii) the data mining algorithm which the privacy preservation technique is designed for; (iv) the data type (single data items or complex data correlations) that needs to be protected from disclosure; (v) the approach adopted for preserving privacy (heuristic, reconstruction or cryptography-based approaches). Figure 1.3 shows a taxonomy of the existing PPDM algorithms according to those dimensions [BN05]. Obviously, it represents a first organization in this new area and does not cover all the possible PPDM algorithms. However, it gives one overview of the algorithms that

have been proposed so far, focusing on their main features. While heuristics are mainly



Hiding Technique
= {perturbation, blocking, swapping, aggregation, generalization, sampling}

Figure 1.3 Taxonomy of PPDM methods

conceived for centralized datasets and cryptography based algorithms are designed for protecting privacy in a distributed scenario by using encryption techniques, reconstruction-based algorithms can be applied to both centralized and distributed datasets depending on specific application environments. In the following, reconstruction-based methods and cryptography-based methods for PPDM are briefly

introduced in section 1.3.1 and 1.3.2 respectively. In section 1.3.3, we introduce the possible privacy breaches that have been discussed in the literature using reconstruction-based and cryptography-based algorithms.

1.3.1 Reconstruction-based Methods

A number of recently proposed techniques address the issue of privacy preservation by perturbing the data and reconstructing the distributions at an aggregate level in order to perform the mining. This approach was first introduced by Agrawal and Srikant [AS00]. The work presented in [AS00] addresses the problem of building a decision tree classifier from training data in which the values of individual records have been perturbed. While it is not possible to accurately estimate original values in individual data records, the authors propose a reconstruction procedure to accurately estimate the distribution of original data values. By using the reconstructed distributions, they are able to build classifiers whose accuracy is comparable to the accuracy of classifiers built with the original data. For the distortion of values, the authors have considered a discretization approach and a value distortion approach. For reconstructing the original distribution, they have considered a Bayesian approach and they proposed three algorithms for building accurate decision trees that rely on reconstructed distributions. The work presented in [AA01] proposes an improvement over the Bayesian-based reconstruction procedure by using an Expectation Maximization (EM) algorithm for distribution reconstruction. More specifically, the authors prove that the EM algorithm converges to the maximum likelihood estimate of the original distribution based on the perturbed data. They also show that when a large amount of data is available,

the EM algorithm provides robust estimates of the original distribution. It is also shown, that the privacy estimates of [AS00] had to be lowered when additional knowledge that the miner obtains from the reconstructed aggregate distribution was included in the problem formulation. Reconstruction-based techniques for binary and categorical data include works presented in [ESAG02], [RH02] and [DZ03]. [ESAG02] and [RH02] deal with binary and categorical data in the context of association rule mining. Both papers consider randomization techniques that offer privacy while they maintain high utility for the data set. Evfiemievski et. al. [ESAG02] proposed a select-a-size randomization technique for privacy preserving mining of association rules. Du et. al. [DZ03] suggested using randomized response techniques for PPDM and constructed decision trees from randomized data. Other Reconstruction-Based works include [LKR06, MS05, MS06a, MS06b, MS06C, Wu03].

1.3.2 Cryptography-based Methods

A number of cryptography-based approaches have also been developed in the context of PPDM algorithms, to solve problems of the following nature. Two or more parties want to conduct a computation based on their private inputs, but neither party is willing to disclose its own input to anybody else. The issue here is how to conduct such a computation while preserving the privacy of the inputs. This problem is referred to as the secure multiparty computation (SMC) problem. Secure computation was first introduced by Yao [Yao86]. Secure computation works in this way: the function to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Secure multiparty computations are closely related to the

problem of secret sharing, and more specifically verifiable secret sharing; every multi-party computation protocol uses verifiable secret sharing. This approach was first introduced into PPDM by Lindell and Pinkas [LP00]. Du et al. [DHC04] proposed a transformation framework that allows systematic transformation of normal computations to secure multiparty computations. Among other information items, a discussion on transformation of various data mining problems to a secure multiparty computation is demonstrated. The data mining applications which are described in this framework include data classification, data clustering, association rule mining, data generalization, data summarization, and data characterization. Clifton et al. [CKV03] present four secure multiparty computation based methods that support PPDM. The methods described include secure sum, secure set union, secure size of set intersection, and scalar product.

In the following, we present secure sum as a simple example of secure multiparty computation. Assume that the value $u = \sum_{l=1}^{s} u_l$ is to be computed and is known to lie in the range $[0, n]$. One site is designated as the master site and is given the identity 1. The remaining sites are numbered $2, \cdots, s$. Site 1 generates a random number *R*, uniformly chosen from $[0, n]$. Site1 adds this number to its local value u_1 and sends the sum $R + u_1$ mod *n* to site 2. Since the value of *R* is chosen uniformly from $[0, n]$, the number $R + u_1$ mod *n* is also distributed uniformly in this range and site 2 learns nothing about the actual value of u_1. For the remaining sites $l = 2, \cdots, s-1$, the algorithm is as follows. Site l receives $V = R + \sum_{j=1}^{l-1} u_j$ mod n. Since this value is uniformly distributed in [0, n], l learns nothing. Site l then computes $R + \sum_{j=1}^{l} u_j$ and passes it to site $l+1$. Site *s* performs the

above step, and sends the result to site 1. Site 1, by knowing *R*, can subtract *R* to get the actual result. Below we present the approaches which have been developed by using the solution framework for secure multiparty computation. Because of the nature of this solution methodology, the data in all of the cases for which this solution is adopted is distributed among two or more sites.

Though secure multiparty computation is appealing in its generality and simplicity, specific and efficient protocol have to be developed for data mining purpose since it is apparently inefficient for data mining applications. Clifton et al. [CKV03, JW05, VC02, VC03] have developed efficient secure computation techniques for association rules, decision tree with vertically partitioned data, clustering, and k-nearest neighbor classification. Wright et al. [WY04, YW05] discuss privacy-preserving Bayesian network structure computation and parameter learning on distributed heterogeneous data while Meng et al. [MSK04] have considered the privacy-sensitive Bayesian network parameter learning problem. The underlying method used in both works is to convert the computations required for Bayesian network learning into a series of inner product computations and then to use a secure inner product computation method proposed elsewhere. All these techniques are based on a semi-honest model, where each party is curious about other parties' data but follows the protocol exactly.

Though SMC method is good at preserving privacy, it still has the following two drawbacks: (1) It assumes semi-honest models, which is often unrealistic in the real world. (2) It requires large volumes of synchronized computations among participating parties. Most of the computations are the overheads due to privacy requirement.

1.3.3 Privacy Breaches

Agrawal and Srikant [AS00] proposed using additive noise to solve PPDM problem. In their randomization scheme, a random number is added to the value of a sensitive attribute. For example, if x_i is the value of a sensitive attribute, $x_i + r$ rather than x_i will appear in the perturbed database, where r is a random value drawn from some distribution. It is shown that given the distribution of random noises, recovering the distribution of the original data is possible. Kargupta et al. [KDWS03] challenged the additive noise schemes, and pointed out that additive noise might not be secure. They proposed a random matrix-based spectral filtering technique to recover the original data from the perturbed data. Their results have shown that the recovered data can be reasonably close to the original data. The results indicate that for certain types of data, additive noise might not preserve privacy as much as we require for PPDM. Motivated by Kargupta et al's work, Huang and Du [HDC05] further suggested that the key factor for those kind of privacy breaches is the correlation among attributes. They propose two data reconstruction methods that are based on data correlations. One method uses the principal component analysis (PCA) and the other method uses Bayes estimate technique. They conducted theoretical and experimental analysis on the relationship between data correlations and the amount of private information that can be disclosed based on the data reconstruction method based on PCA and Bayes estimation technique. Their studies have shown that when the correlation between the attributes is high, the original data can be reconstructed more accurately, i.e., more privacy breaches will happen.

Possible privacy breaches have been also discussed by [AA00, EGS03]. [AA00] suggests by an example that the method in [AS00] does not take into account the

distribution of original data, which is reconstructed during PPDM. This kind of reconstruction of the distribution also provides a certain level of knowledge of sensitive attributes, which can be used to guess the value of a sensitive attribute to a high level of accuracy. Another possible privacy breach using randomization scheme is proposed by [EGS03]. It is not enough to simply concentrate on randomization and recovery of the model. We must also ensure that the randomization is sufficient for preserving privacy, as we randomized in the first place to achieve privacy. The following two examples by [EGS03] explain this kind of privacy breach.

Example 1: Suppose we randomize age x_i by adding a random number r_i drawn uniformly from an interval say $[-50, 50]$. Assuming the server receives age 120 from a user, privacy is somewhat compromised, as the server can conclude that the age of the user cannot be less than 70 (otherwise 120 will not result). Thus the server has learned a potentially valuable piece of information about the user.

Example 2: Suppose we randomize a small set of items (a transaction) by replacing each item by a random item with probability 80%. If the transaction contains a subset A of three items that has a support of 1%, it has $(0.2)^3 = 0.008 = 0.8\%$ chance to retain the same set of three items after the randomization. Thus whenever the server sees A in the randomized transaction, it learns with high probability of the presence of A in the original transaction as well. Indeed there are $1\% * 0.8 = 0.008\%$ randomized transactions that have *A* both before and after randomization, while the probability that *A* occurs in 10 randomly inserted items (out of, say, 10,000 possible items) is less than $10^{-7}\%$.

1.3.4 PPDM using PRAM

Gouweleeuw et al. [GKW98] introduced PRAM for statistical databases for information disclosure control. Disclosure control in statistical database is to preserve the privacy of individual records when the database is released for simple public use. PPDM is a more challenging problem than disclosure control in statistical database. The PPDM technique has proved to be effective in disclosure control. We explored the possibility of using PRAM in PPDM. PRAM is so called because the randomization happens after the data has been collected. The proposed algorithm applies both to uniform and non-uniform randomization in order to meet the different privacy requirement of different variables. We compute estimators for frequency counts for data used in data mining and estimators of their covariance, based on the randomized data. PRAM is an efficient, flexible and easy-to-use method to learn from privacy sensitive data. Using PRAM in PPDM overcomes the inherent drawbacks of SMC method and provides reasonable privacy and accuracy. Using PRAM for PPDM provides a general framework for randomization of variables in PPDM. We can also apply PRAM to numerical variables by proper discretization and then reconstruct their probability distribution during data mining. The main problem is to design the PRAM intelligently according to different applications. Select-a-size randomization can be considered as a special and intelligently designed PRAM. MASK in [RH02] is a PRAM implemented to binary variables. We can even view the randomization of numerical data using additive noise [AS00] as a special design of PRAM since the numerical values are also discretized into intervals and the distribution of the variables are also recovered using the frequency counts of the records in the interval. One difference between our schemes of implementing PRAM to

discretized numerical variable and implementing additive noise is that our framework provides a more flexible way to control those privacy breaches discussed in the previous section by controlling γ-amplification, a privacy measure for limiting those kind of privacy breach, which will be introduced in detail in chapter 3. This important difference will be also further discussed in chapter 3. Works in [ESAG02, EGS03, DZ03] randomized the data by records, which actually implements simultaneous randomization to all variables. Simultaneous randomization to all variables introduces unnecessary randomization. However, simultaneous randomization can reduce simultaneous γ-amplification. PRAM provides a more flexible way to handle variables with different privacy requirement by using non-uniform randomization; for example, a group of variables can be randomized simultaneously and others can be randomized independently.

Our framework for PPDM using PRAM can be applied for two different cases in distributed data mining environments discussed in section 1.1.

a. There is a data miner in addition to the parties that own the database. Every party who owns a part of the data simply sends the data miner a randomized version of their data, which are randomized according to their privacy requirements. The probability transition matrices (used to perform the randomization) are also sent to the data miner. The data miner does all the learning using randomized data and probability transition matrices from those parties.

b. No data miner exists. All parties who own the database have to cooperate and take part in mining. This setting is quite similar to the multi-party model of SMC, where every party takes part in the computation and has to have some ability to perform computations.

In homogeneously distributed data environment, PPDM is relatively easy for learning models that only depend on sufficient statistics of the data set, for example, the frequency counts of each feature. Each party can share sufficient statistics from their part of the database without compromising privacy of individual records. However, special care still has to be given to the fully homogeneously distributed data set since the sufficient statistics for each customer might be the individual record itself. Our framework for PPDM using PRAM can be applied in the fully homogeneously distributed environment. In fully homogeneously distributed environment, we consider the above case (a) only, since case (b) will require every customer holding a single record take part in the learning, which is quite impractical. Figure 1.4 graphically illustrates PPDM using PRAM in fully homogeneously distributed environment. Every customer in the fully homogeneously distributed environment sends a randomized version of its record using pre-defined randomization schemes known to the data miner.

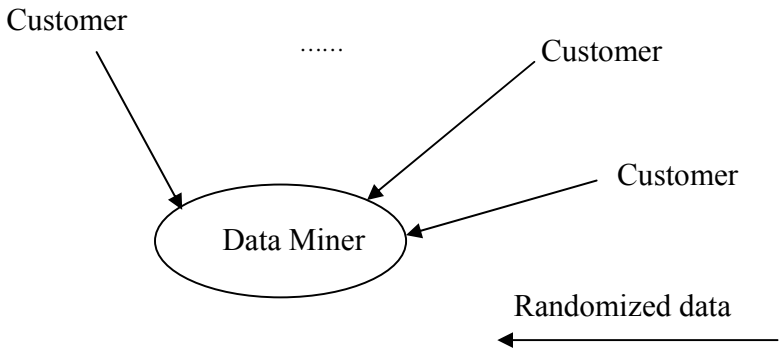


Figure 1.4 PPDM in fully homogeneously distributed environment

A challenging task for PPDM in heterogeneously distributed data set is to preserve the correlation or joint distribution of all features without accessing the original data. Figure 1.5(a) and 1.5(b) graphically illustrate PPDM using PRAM techniques in heterogeneously distributed environment and under the above case (a) and (b)

respectively. In case (b), every party sends to every other party their randomized data when they are necessary for that party's computation. In case (a), every party simply sends their randomized data to the data miner. In general, there is less randomized data in case (a) than case (b) since variables owned by the party who is doing computation do not need to be randomized for his own computation. Simulation results of case (a) are in general better than case (b). While the learning and analysis from randomized data is almost the same in these two cases, case (b) needs cooperation and synchronization between the participating parties. Cooperation and synchronization requirements are similar to those in PPDM using SMC method. However, SMC method relies on the semi-honest model. It is quite easy for a malicious party in SMC to get other party's private information by not obeying the protocols while those parties can only get randomized information from other parties if they do not obey the protocols deliberately.

When there is a data miner, PPDM algorithm used in the above case (a) in heterogeneously distributed environment can be directly used in the arbitrarily distributed environment since every party in arbitrarily distributed environment also simply sends their randomized data to the data miner using pre-defined randomization schemes and thus they are exactly the same from the data miner's perspective. If there is no data miner, case (b) can also be used in the arbitrarily distributed environment with a protocol depending on application and how the data set is distributed.

This dissertation for the first time proposed to use PRAM for PPDM in research of PPDM. The privacy preserved by PRAM schemes are measured by a metric called γ-amplification. The concept of γ-amplification was proposed by [EGS03]. We proposed originally to use γ-amplification as a privacy metric for PRAM. Privacy preserved by

different PRAM schemes and techniques is originally analyzed by using γ-amplification.

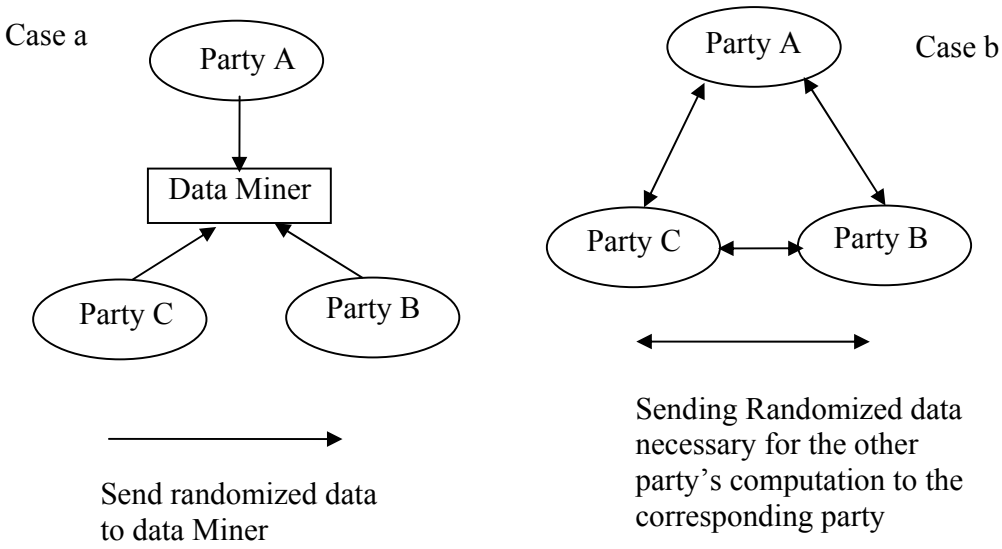


Figure 1.5: PPDM in heterogeneously distributed environment

CHAPTER TWO

PRAM TECHNIQUES

In this chapter, PRAM techniques are introduced. PRAM is introduced in section 2.1. Problems in applying PRAM in practice are discussed in section 2.2. PRAM is initially implemented for randomizing categorical variables. In section 2.3, we discuss implementing PRAM to discretized numerical variables. Frequency counts estimation methods are discussed in section 2.4. Finally, experimental results are provided in section 2.5.

2.1 PRAM

Consider a data set D with a set U of variables $X_1, X_2, \ldots, X_n$, where X_i $(i = 1, \cdots, n)$ takes discrete values from a set S_i whose cardinality is K_i. $D = \{y_1; y_2; \cdots; y_N\}$, where $y_i = \{y_{i1}, y_{i2}, \cdots, y_{in}\}$ is a $1 \times n$ vector such that $y_{i1} \in S_1, y_{i2} \in S_2, \cdots, y_{in} \in S_n$. The PRAM for variable X_i is a (random) mapping $R_i : S_i \to S_i$, based on a set of probabilities $p_{lm}^i = p(\tilde{X}_i = k_m \mid X_i = k_l)$, where $k_m, k_l \in S_i$ and $\tilde{X}_i$ denotes the (randomized) variable value corresponding to variable X_i. The transition probability p_{lm}^i is the probability that a variable with original value k_l is randomized to the value k_m. Let $P^i = \{p_{lm}^i\}$ denote the $K_i \times K_i$ dimensional matrix that has p_{lm}^i as its (l, m) th entry. The randomized data set is $\tilde{D} = \{\tilde{y}_1; \tilde{y}_2 \cdots, \tilde{y}_N\}$, where $\tilde{y}_i$ is a randomized vector corresponding to y_i. Theoretically, we can choose any randomization scheme. However, the condition that P^i is nonsingular has to be imposed if we want to

estimate the frequency distribution of variable X_i from the randomized data set $\tilde{D}$. Note that PRAM is also applied independently to each record in the dataset.

PRAM is applied to each variable independently and a transition matrix is released per variable. PRAM can also be applied on several variables simultaneously. Consider the case when we want to apply PRAM simultaneously to two variables X_i and X_j who take discrete values from set S_i and S_j with cardinalities K_i and K_j respectively. Simultaneous PRAM for variables X_i and X_j is a (random) mapping $R_{ij} : S_i \times S_j \to S_i \times S_j$, based on a set of transition probabilities $p^{ij}_{l_1 l_2 k_1 k_2} = p(\tilde{X}_i = k_{m_1}, \tilde{X}_j = k_{m_2} \mid X_i = k_{l_1}, X_j = k_{l_2})$. Let the corresponding probability transition matrix $P^{ij} = \{p^{ij}_{l_1 l_2 m_1 m_2}\}$, which is now a $(K_i \times K_j) \times (K_i \times K_j)$ matrix. Simultaneous PRAM is also applied independently to each record in the data set. We can see that there is no essential difference between applying PRAM to variables independently and applying PRAM simultaneously to two (or more) variables. If we had applied the PRAM independently to the two variables X_i and X_j , the probability transition matrix would be $P^{ij}_{l_1 l_2 m_1 m_2} = P^i \otimes P^j$, where $\otimes$ denotes the Kronecker product. Kronecker product of a $K_1 \times K_1$ matrix P and a $K_2 \times K_2$ matrix Q, is a $K_1 K_2 \times K_1 K_2$ matrix such that $P \otimes Q = \begin{bmatrix} p_{11}Q & \cdots & p_{1n}Q \\ \vdots & \ddots & \vdots \\ p_{n1}Q & \cdots & p_{nn}Q \end{bmatrix}$. Application of PRAM simultaneously to more than two variables is straightforward. The extreme case is applying PRAM simultaneously to all the variables in the data set. Applying PRAM independently to each variable or applying PRAM simultaneously to some of the variables is a matter of design

choice. We can apply the same randomization schemes independently to all of the variables: uniform randomization to the data set. Alternatively, we can use a non-uniform randomization where different PRAM schemes are applied to different variables, independently. The non-uniform randomization is effective when different variables have different sensitivity levels. For example, we can choose different randomization parameters p_1 and p_2 to different binary variables for non-uniform randomization if the privacy requirement of the two variables are different. The non-uniform randomization includes the special case when there is no privacy requirement for some of the variables. In the following, we provide some simple but effective PRAM schemes on which most of our experiments are based. If variable X_i takes binary values, we can use binary randomization as shown in Figure 2.1(a). Implementing different randomization with $p_1 \neq p_2$ to different values of a binary variable can be used in the case when one value of the binary variable is more sensitive than the other. If the variable X_i is ternary, a ternary symmetric randomization as shown in Figure 2.1(b) can be used.

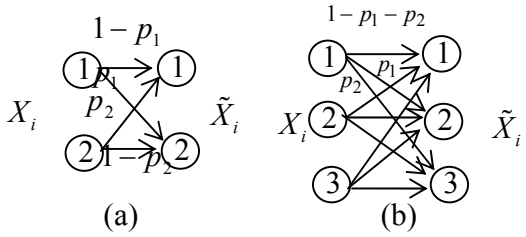


Figure 2.1: Randomization Schemes:

Another PRAM scheme we frequently used for multi-category variable X_i is as follows:

$$p_{lm}^{i} = p(\tilde{X}_i = k_m \mid X_i = k_l) = \begin{cases} 1-p & if \quad m = l \\ \dfrac{p}{K_i - 1} & if \quad m \neq l \end{cases}$$

In the sequel, we will call this randomization scheme as multi-category randomization.

2.2 Problems when Applying PRAM in Practice

The implementation of PRAM to data in practice involves several aspects. We discuss those aspects in this section.

First of all, it has to be decided which variables will be perturbed. If PRAM is applied to (some of the) identifying variables in the data, then as a result it becomes more difficult for an intruder to recognize a record corresponding to some individual in the population. PRAM can also be applied to sensitive variables in the data. In that case, an intruder may recognize the record of an individual, but he or she can not be sure as to whether the sensitive information obtained from the data is correct. From a statistical viewpoint, it does not matter whether PRAM is applied to identifying or sensitive variables.

There are usually many variables in a data set, which are candidates for applying PRAM. A choice has to be made whether it is preferable to perturb only a few variables a lot or to mildly perturb many variables. For the same privacy requirement, it is preferable to choose the one which will produce more accurate frequency counts estimation by choosing smaller variance of the estimator (we will discuss estimators in section 2.4). The other thing is to decide whether applying randomization to all the variables independently or apply randomization to some of the variables simultaneously. Simultaneous randomization will usually prevent the inconsistency of the randomized data, which is often a problem with independent randomization. Also, simultaneous

randomization can cause unnecessary randomization to some of the variables but it will usually have lower simultaneous γ-amplification, a privacy measure defined in chapter 3.

After it has been decided to which variables PRAM should be applied, the next step is to decide which category can be replaced by which category and what randomization schemes should be used. From the above, we can see that if variable X_i takes K_i values (or categories), the dimension of P^i will be $K_i \times K_i$. With larger K_i, more randomization is introduced into variable X_i in general. This is good from a privacy point of view. However, the variances of the estimator of frequency counts will also be larger for a given size of training data. One solution to this problem is to partition the K_i categories of variable X_i into several groups such that a value in one group can only be randomized to a value in the same group. If grouping of the categories is used, how to group those categories should also be decided. The grouping structure determines the structure of the probability transition matrix P. For categories $1,\cdots,K$ grouped into G groups, the matrix can be written as $\begin{bmatrix} P_1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & P_G \end{bmatrix}$. If the grouping of the categories is used, a privacy measure we define in chapter 4, probabilistic K-anonymity K_P, will be lower, which means worse privacy (using this measure). However, grouping of the categories will usually produce lower $\gamma - amplification$, another privacy measure defined in chapter 3, which means better privacy (using this measure).

After all of the above, the remaining item is to decide the values of p_{lm}^i. The choice of the values of p_{lm}^i is not a trivial task in practice. It has to be decided according to the trade-off between accuracy and privacy. Different choices of randomization

schemes will affect the specific probability transition matrix used in learning but will not affect the learning algorithm used for learning from randomized data.

2.3 Implementing PRAM to Discretized Quantitative Data

In many cases, actual data are numerical. PRAM was originally developed for categorical data. Some techniques have been proposed in the literature to reduce the number of values for a given continuous attribute, by dividing the range of the attribute into intervals. All those techniques are introduced in the framework for data preprocessing. However, we propose discretization in the framework for PPDM. Though we can directly borrow many existing ideas, many algorithms have to be modified to suit privacy. Reducing the number of values for a given continuous attribute is quite beneficial for many models to be learned since over-fitting can be greatly reduced though details are lost. From a privacy point of view, it provides some ambiguity (privacy) to the actual data by certain intervals. Many techniques can provide a hierarchical or multi-resolution partitioning of features known as a concept hierarchy. Data owner can disclose appropriate level of the hierarchy to the data miner or other parties according to their privacy requirements.

Privacy requirements usually have two aspects.

a. Minimum granularity requirement *g*: Larger *g* provides more ambiguity (privacy) but loses more details. For univariate case, the minimum granularity is equivalent to the minimum interval.

b. The minimum number of records *K* in the transformed categories. This *K* provides a level of *K*-anonymity to possible identification.

The existing methods for discretization of numerical variables usually emphasize one of the above but ignore the other. A clustering algorithm can be applied to partition data into clusters or groups. Using a clustering algorithm to transform the numerical data into categories is suitable for both univariate and multivariate cases. However, any clustering algorithm has to be modified since the object of a clustering algorithm is to maximize the in-group similarity and between-group dissimilarity without any constraints on group size and granularity. Privacy requirements, however, do not specifically require optimization of in-group similarity and between-group dissimilarity as an objective. The hierarchical clustering algorithm produces a hierarchy of clusterings and is easier to modify to meet the two privacy requirements. We propose a Modified Agglomerative Scheme (MGAS) to discretize the numerical variables in favor of the two privacy requirements. MGAS is similar to the heuristic method proposed for data-oriented microaggregation for statistical disclosure control in [DM02]. However, the method proposed in [DM02] constrains both the minimal and maximal size of groups while MGAS constrains the minimal number of records in a cluster and the minimal granularity. Constraining both minimal and maximal is appropriate for the scenario of re-identification disclosure. However, imposing this kind of constraint could still result in possibly disclosing the individual value of a data since the numerical differences between some groups are very likely to be negligible. In the framework for privacy of individual value of the data, controlling the minimal granularity as done by MGAS is more reasonable.

The following is the Modified Agglomerative Scheme (MGAS). The similarity and dissimilarity we consider here are Euclidean Distances.

(a) Initialization

i. Choose $R_0 = \{C_i = \{x_i\}, i = 1 \cdots N\}$ as the initial clusterings.

ii. t = 0;

(b) Repeat

i. $t = t + 1$

ii. Among all possible pairs of clusters (C_r, C_s) in R_{t-1}, find the one, say (C_i, C_j) such that $g(C_i, C_j) = \min_{r,s} g(C_r, C_s)$ if g is a dissimilarity function and $g(C_i, C_j) = \max_{r,s} g(C_r, C_s)$ if g is a similarity function.

iii. Define $C_q = C_i \cup C_j$ and produce the new clustering

$R_t = \{R_{t-1} - \{C_i, C_j\}\} \cup \{C_q\}$

(c) Checking the Granularity and the number of records in each cluster. Let R_K be a subset of R_t and every cluster in R_K has met the minimum requirement of Granularity and minimum requirement number of records. $R_f = R_f \cup R_k$ and $R_t = R_t - R_k$

(d) Until all clusters meet the two requirements. The final set of clusters is R_f

After the numerical data is grouped using the MGAS algorithm, we can do the following according to the different models to be learned.

a. If the data models to be learned require the numerical properties of the data, e.g. generalized linear regressions, linear classifier and models based on optimizing empirical risk minimization, the original values of variables are replaced by group means they belong to.

b. If the data models to be learned require only the summaries or sufficient statistics of the data, e.g. association rules, decision tree, Bayesian network, we can replace

individual values of the variables by general concepts or simply the assigned orders of the groups they belong to. PRAM can now be applied to the transformed numerical variables which are now discretized. PRAM provides another level of privacy for the data besides the anonymity introduced by discretization using MGAS. The implementation of PRAM to discretized numerical variables is the same as that for categorical variables.

γ-amplification can be controlled more effectively by using discretization first followed by PRAM as compared with additive noise method, where discretization happens after the noise is added and during the estimation of frequency counts of intervals. The additive noise method usually has a very large γ-amplification, which is usually near infinite. We will discuss this issue further in Chapter 3.

2.4 Frequency Counts Estimation

Frequency counts estimation is the key part of PPDM using PRAM. PPDM using PRAM is made possible because we can reasonably estimate frequency counts. The estimated frequency counts are used in the subsequent learning algorithm. Two kinds of estimation methods can be used in estimating the frequency counts: moment estimation and maximum likelihood estimation. Moment estimation is introduced in section 2.4.1 and the maximum likelihood estimation is introduced in section 2.4.2.

2. 4.1 Moment Estimation

We consider a general case, where n variables $X_1,\ldots,X_n$ of a database are randomized independently of each other. Each record is also randomized independently.

Suppose the probability transition matrices of $X_1,\cdots,X_n$ are $P^1,\ldots,P^n$ respectively. The probability transition matrix for non-randomized variable is an identity matrix. Suppose $X_1,\ldots,X_n$ take values from sets $S_1,\ldots,S_n$ respectively and the cardinalities of sets $S_1,\ldots,S_n$ are $K_1,\ldots,K_n$ respectively. The problem here is to estimate a $J=\prod_{i=1}^{n}K_i$ dimensional vector of frequency counts N_X from the randomized data. Each element of N_X , say N_j for $j=1,\cdots,\prod_{i=1}^{n}K_i$, is the number of records such that $X_1=k_{j_1},\ldots,X_n=k_{j_n}$, where $k_{j_1}\in S_1,\ldots,k_{j_n}\in S_n$. The order of arrangement of elements N_j in the vector N_X is such that the change of variable X_1 is the fastest and X_n is the slowest. These variables can be arbitrary vertically partitioned. The variables can also be combined variables. The randomization can also be done by grouping the categories of the variables into groups. How partitioning, combining and grouping are done does not affect our discussion below since they only affect the structure and value of specific probability transition matrix entries. The following are some notations used in the sequel: ~ denote the variables or vectors after randomization and $\wedge$ denote the estimate of the corresponding vectors or variables. Given a training data set *D* with *N* records for *n* variables $X_1,\ldots,X_n$ and a randomization scheme characterized by probability transition matrices $P_1,\ldots,P_n$, we have the following theorems.

Theorem 1

a. $E[\tilde{N}_X\mid D]=P^tN_X$, where $P=P_1\otimes\ldots,\otimes P_n$, $\otimes$ denotes Kronecker matrix product and t denotes transpose.

b. Let Y_{ml} denote a binomial random variable that gives the number of records such that $\{\tilde{X}_1,\cdots,\tilde{X}_n\}$ takes value corresponds to N_l, lth element of N_X, for $l=1,\cdots J$, while $\{X_1,\cdots,X_n\}$ takes value corresponds to $N_m, m=1,\cdots,J$, i.e., the number of records such that they take values corresponds to N_m th configuration and are randomized to the value corresponds to N_l th configuration. We have $Y_{ml}^i \sim B(N_m,\pi)$, where $\pi = P(m,l)$, the (m,l) th element of the probability matrix P defined in Theorem 1(a) and *B* denotes Binomial distribution. Moreover, $Cov\{Y_{ml_1}^i, Y_{nl_2}^i\} =$

$$\begin{cases} var\{Y_{ml}^i\} = N_m P(m,l_1)(1-P(m,l_1)) & \text{if } n=m, l_1=l_2 \\ -N_m P(m,l_1)P(m,l_2) & \text{if } n=m, l_1\neq l_2 \\ 0 & \text{if } n\neq m \end{cases}$$

c. For $l=1,2\cdots,J$, $\tilde{N}_l = \sum_{m=1}^{J} Y_{ml}$. Moreover, $Cov\{N_X \mid D\} = \sum_{l=1}^{J} N_l V_l$, where V_l is a $J\times J$ covariance matrix such that its (l_1,l_2) th element

$$V_l(l_1,l_2) = \begin{cases} P(l,l_1)(1-P(l,l_1)) & \text{if } l_1=l_2, \\ -P(l,l_1)P(l,l_2) & \text{if } l_1\neq l_2. \end{cases}$$

Proof: All the probabilities in the proof are conditional on the training data *D*. We skip the notation of $"\mid D"$ in the proof for simplicity.

(a) $E\{\tilde{N}_j\} = Np(\tilde{X}_1 = k_{j_1}, \tilde{X}_2 = k_{j_2}\cdots\tilde{X}_n = k_{j_n})$

$$= N\sum_{l_1=1}^{K_1}\sum_{l_2=1}^{K_2}\cdots\sum_{l_n=1}^{K_n} p(X_1=k_{l_1}, X_2=k_{l_2}\cdots X_n=k_{l_n}, \tilde{X}_1=k_{j_1}, \tilde{X}_2=k_{j_2}\cdots\tilde{X}_n=k_{j_n})$$

$$= \sum_{l_1=1}^{K_1}\sum_{l_2=1}^{K_2}\cdots\sum_{l_n=1}^{K_n} Np(X_1=k_{l_1}, X_2=k_{l_2}\cdots X_n=k_{l_n})\, p(\tilde{X}_1=k_{j_1}, \tilde{X}_2=k_{j_2}\cdots\tilde{X}_n=k_{j_n} \mid X_1=k_{l_1}, X_2=k_{l_2}\cdots X_n=k_{l_n})$$

$$= \sum_{l=1}^{J} N_l p(\tilde{X}_1=k_{j_1} \mid X_1=k_{l_1})\, p(\tilde{X}_2=k_{l_2} \mid X_2=k_{j_2})\cdots p(\tilde{X}_n=k_{j_n} \mid X_n=k_{l_n})$$

Total probability, Bayes theorem, and independent randomization of each variable are used in the above proof. We also used in the above proof the fact that $NP(X_1 = k_{l_1}, X_2 = k_{l_2} \cdots X_n = k_{l_n}) = N_l$ (the prior probabilities for the dataset D). Therefore, $E\{\tilde{N}_j\} = \sum_{l=1}^{J} N_l p(\tilde{X}_1 = k_{j_1} \mid X_1 = k_{l_1}) p(\tilde{X}_2 = k_{l_2} \mid X_2 = k_{j_2}) \cdots p(\tilde{X}_n = k_{j_n} \mid X_n = k_{l_n})$. This can be written in matrix form as $E[\tilde{N}_X \mid D] = P^t N_X$, where $P = P^i \otimes P^{Pa_i}$ and $P = P_1 \otimes P_2 \otimes \ldots \otimes P_n$.

(b) Data records for which $\{X_1, X_2, \cdots, X_n\}$ are in the m th configuration (we have N_m such records in the dataset) are randomized independently of each other. They are randomized to the configuration such that $\{X_1, X_2, \cdots, X_n\}$ is in l th configuration with probability $P(m,l)$ and to other configurations with probability $1 - P(m,l)$. It is obvious that it is a binomial variable with distribution $Y_{ml}^i \sim B(N_i(m), P(m,l))$, since there are totally $N_i(m)$ records with $\{X_1, X_2, \cdots, X_n\}$ in the m th configuration for a given data set D. Hence $E[Y_{ml}^i] = N_i(m)P(m,l)$ and $Var[Y_{ml}^i] = N_i(m)P(m,l)(1 - P(m,l))$. Because the randomization is applied independently to each record, the randomization of record with $\{X_1, X_2 \cdots X_n\}$ in m th configuration is independent of the randomization of record that $\{X_1, X_2 \cdots X_n\}$ in n th configuration when $n \neq m$. Hence, $Y_{ml_1}^i$ is independent of $Y_{nl_2}^i$ when $m \neq n$, i.e. $Cov\{Y_{ml_1}^i, Y_{nl_2}^i\} = 0$. When $m = n$ and $l_1 = l_2$, $Cov\{Y_{ml}^i, Y_{nl_2}\} = Var\{Y_{ml_1}\} = N_i(m)P(m,l_1)(1 - P(m,l_1))$. When $m = n$ and $l_1 \neq l_2$, $Cov\{Y_{ml_1}^i, Y_{nl_2}^i\} = E\{Y_{ml_1}^i Y_{ml_2}^i\} - E\{Y_{ml_1}^i\} E\{Y_{ml_2}^i\}$. It is not difficult to show that

$Cov\{Y^i_{ml_1}, Y^i_{nl_2}\} = -N_i(m)P(m,l_1)P(m,l_2)$ by using two indicator functions. When $m = n$ and $l_1 \neq l_2$, $Cov\{Y^i_{ml_1}, Y^i_{nl_2}\} = E\{Y^i_{ml_1}Y^i_{ml_2}\} - E\{Y^i_{ml_1}\}E\{Y^i_{ml_2}\}$.

For each data instance that has *m*th configuration, we assign it two indicators:

$$I_k^{l_1} = \begin{cases} 1 & \textit{if randomized to } l_1\textit{th configuration} \\ 0 & \textit{otherwise} \end{cases}$$

and $I_k^{l_2} = \begin{cases} 1 & \textit{if randomized to } l_2\textit{th configuration} \\ 0 & \textit{otherwise.} \end{cases}$.

Clearly, $Y^i_{ml_1} = \sum_{k=1}^{N_i(m)} I_k^{l_1}$ and $Y^i_{ml_2} = \sum_{k=1}^{N_i(m)} I_k^{l_2}$. We have $E\{I_{k_1}^{l_1} I_{k_2}^{l_2}\} = 0$ for $k_1 = k_2$ and $l_1 \neq l_2$ since one record cannot be randomized into two different configurations.

Therefore, $E\{Y^i_{ml_1}Y^i_{ml_2}\} = E\{\sum_{k1=1}^{N_i(m)} I_{k1}^{l_1} \sum_{k2=1}^{N_i(m)} I_{k2}^{l_2}) = \sum_{k1=1}^{N_i(m)} \sum_{k2=1, k2\neq k1}^{N_i(m)} E\{I_{k1}^{l_1} I_{k2}^{l_2}\}$ when $l_1 \neq l_2$.

$E\{Y^i_{ml_1}Y^i_{ml_2}\} = \sum_{k1=1}^{N_i(m)} \sum_{k2=1, k_2\neq k_1}^{N_i(m)} E\{I_{k1}^{l_1}\}E\{I_{k2}^{l_2}\}$, since $I_{k1}^{l_1}$, and $I_{k_2}^{l_2}$ are independent when $k_1 \neq k_2$.

Therefore,

$Cov\{Y^i_{ml_1}, Y^i_{nl_2}\} = E\{Y^i_{ml_1}Y^i_{ml_2}\} - E\{Y^i_{ml_1}\}E\{Y^i_{ml_2}\} = -\sum_{k1=1}^{N_i(m)} E\{I_{k1}^{l_1}\}E\{I_{k1}^{l_2}\}$. It is clear that

$E\{I_{k1}^{l_1}\} = P(m,l_1)$ and $E\{I_{k1}^{l_2}\} = P(m,l_2)$

Hence $Cov\{Y^i_{ml_1}, Y^i_{nl_2}\} = -N_i(m)P(m,l_1)P(m,l_2)$ when $m = n$ and $l_1 \neq l_2$.

(c) From part (b), $\tilde{N}_l = \sum_{m=1}^{J_iK_i} Y^i_{ml}$, where $Y^i_{ml} \sim B(N_i(m), P(m,l))$. $Y^i_{m_1 l}$ and $Y^i_{m_2 l}$ are independent of each other when $m_1 \neq m_2$. Therefore,

$Var\{\tilde{N}_l\} = \sum_{m=1}^{J_iK_i} Var\{Y^i_{ml}\} = \sum_{m=1}^{J_iK_i} N_i(m)P(m,l)(1-P(m,l))$ and

$Cov\{\tilde{N}_{l_1}, \tilde{N}_{l_2})\} = Cov\{\sum_{m=1}^{J_iK_i} Y^i_{ml_1}, \sum_{n=1}^{J_iK_i} Y^i_{nl_2}\} = \sum_{m=1}^{J_iK_i} Cov\{Y^i_{ml_1}, Y^i_{ml_2}\}$ because $Cov\{Y^i_{ml_1}, Y^i_{nl_2}\} = 0$ for $m \neq n$. So, $Cov\{\tilde{N}_i(l_1), \tilde{N}_i(l_2)\} = \sum_{m=1}^{J_iK_i} -N_m P(m,l_1)P(m,l_2)$ by (b) when $l_1 \neq l_2$. This shows (c).

The following theorem establishes the bias and variance of the estimator $\hat{N}_i = (P^t)^{-1}\tilde{N}_i$. Its proof is straight-forward and is omitted.

Theorem 2

$\hat{N}_X = (P^t)^{-1}\tilde{N}_X$ is an unbiased estimator for N_X and $Cov\{\hat{N}_X \mid D\} = (P^{-1})^t Cov\{\tilde{N}_X \mid D\}P^{-1}$, where P and $Cov\{\tilde{N}_X \mid D\}$ are given in Theorem 1.

In order to illustrate the above theorems, we consider an example, where two variables X_1, and X_2 have been randomized. Variable X_1 has K_1 categories and randomization scheme used to randomize X_1 has probability transition matrix P_1. Variable X_2 has K_2 categories and randomization scheme used to randomize X_2 has probability transition matrix P_2. $N_X = [N_1, N_2 \cdots N_J]^t$ and $\tilde{N}_X = [\tilde{N}_1, \tilde{N}_2 \cdots \tilde{N}_J]^t$, where $J = K_1K_2$. By definition, N_1 is the number of records such that $X_1 = 1$ and $X_2 = 1$. N_J is the number of records such that $X_1 = K_1$ and $X_2 = K_2$. The order of elements of N_X is such that values of X_1 changes slower than that of X_2. By theorem 1(a), we have $E[\tilde{N}_X \mid D] = P^t N_X$. P is a $K_1K_2 \times K_1K_2$ transition matrix as

follows: $P = P_1 \otimes P_2 = \begin{bmatrix} p^1_{11}P_2 & p^1_{12}P_2 & \cdots & p^1_{1K_1}P_2 \\ \vdots & \ddots & \ddots & \vdots \\ p^1_{K_1 1}P_2 & \cdots & \cdots & p^1_{K_1 K_1}P_2 \end{bmatrix}$, where p^1_{ij} is the $(i,j)th$ element of matrix P_1 and each $p^1_{ij}P_2$ for $i, j \in \{1, \cdots K_1\}$ is a $K_2 \times K_2$ matrix. By theorem 2, $\hat{N}_X = (P^t)^{-1}\tilde{N}_X$ is an unbiased estimator of N_X.

In order to simplify the distribution of the estimator, we can use the normal approximation of the distribution of estimators. The DeMoivre-Laplace Theorem tells us that a Binomial distribution $B(N, p)$ can be approximated by a normal distribution $Normal(Np, Np(1-p))$, when N is large. Since in data mining we usually have a relatively large sample size, the distribution of $\tilde{N}_X$ can be well approximated by a summation of J normally distributed random variables. The distribution of $\tilde{N}_l$ can also be approximated by a normal distribution. $\tilde{N}_X$ and $\hat{N}_X$ can be approximated by a J-dimensional joint normal random variable since $\hat{N}_X$ is a linear transformation of $\tilde{N}_X$. In particular, $\hat{N}_X \sim Normal(N_X, Cov\{\hat{N}_X \mid D\})$, i.e. $\hat{N}_X$ is a J-dimensional normal variable with covariance matrix $Cov(\hat{N}_X \mid D)$ where $Cov\{\hat{N}_X \mid D\} = (P^{-1})^t[\sum_{l=1}^{J} N_l V_l](P^{-1})$ and V_l is a $J \times J$ covariance matrix such that $V_l(l_1, l_2) = \begin{cases} P(l,l_1)(1-P(l,l_1)) & if \quad l_1 = l_2 \\ -P(l,l_1)P(l,l_2) & if \quad l_1 \neq l_2 \end{cases}$, where $P(l, l_1)$ is the $(l, l_1)th$ entry of the matrix P.

One observation we can make for the moment estimate $\widehat{\theta} = \frac{\hat{N}_X}{N}$ for $\theta = \frac{N_X}{N}$ is that it will provide a zero-error estimate if the sample size N of the data set D is very

large since it is unbiased and $Cov\{\hat{\theta} \mid D\} = \frac{1}{N^2} Cov\{\hat{N}_X \mid D\} = \frac{1}{N^2} (P^{-1})^t [\sum_{l=1}^{J} N_l V_l](P^{-1})$ is inversely proportional to the sample size N in order.

2.4.2. Maximum Likelihood Estimation

One problem with the above moment estimator is that it can yield estimates outside of the parameter space, for example, negative frequency counts, which is awkward. In order to avoid such kind of negative frequency, we can use the maximum likelihood estimator, which is not unbiased but has many other desirable properties. Negative frequency counts occur more often when the number of records in some categories are small, which is usually true when PRAM is applied to discretized continuous variables since large interval discretization will lead to poor model accuracy.

As in moment estimation, we assume the data is fixed and ignore the sampling error. Let $\theta = \{\theta_1, \ldots, \theta_j, \cdots \theta_J\} = \frac{N_X}{N}$, where $j = 1, \ldots, J$ and θ_j is equivalent to the probability such that variable X is in the j th configuration since we ignore sampling errors. The loglikelihood of the randomized data $\tilde{D}$ is $LL(\tilde{D} \mid \theta) \propto \tilde{N}_1 Log(\tilde{\theta}_1) + \tilde{N}_2 Log(\tilde{\theta}_2) + \ldots + \tilde{N}_j Log(\tilde{\theta}_J)$, where $\tilde{\theta} = \{\tilde{\theta}_1, \ldots, \tilde{\theta}_J\} = P\theta$. Since matrix P is known, $LL(\tilde{D} \mid \tilde{\theta})$ can be denoted as $LL(\tilde{D} \mid \theta)$. The objective is to find θ that maximizes $LL(\tilde{D} \mid \theta)$. Analytical solution exists for simple cases, however iterative algorithm, eg., Expectation Maximization algorithm (EM), is more convenient in complex cases when the dimension J is very large.

EM algorithm proceeds as if a more comprehensive data, say $DC = d$ is observable and maximizes $LL(DC \mid \theta)$ over all values of θ (M-Step). Since $DC = d$ is not available,

$LL(DC|\theta)$ is replaced by its conditional expected value given $\tilde{D}$ and the current estimate of θ (E-Step). The complete data DC can be chosen to make the E-step easy to compute. We can choose data D as the the complete data. Define a Q function as follows: $Q(\theta,\theta^t) = E[LL(D|\theta)|\tilde{D};\theta^t]$, the expected value of $LL(D)$ with respect to $f(D|\tilde{D},\theta^t)$. The essence of the EM algorithm is that maximizing $Q(\theta,\theta^t)$ leads to an increase in the logliklihood $LL(\tilde{D}|\theta)$ of the observed data. After initialization of θ to a initial value θ^0, the EM algorithm will iterate over the following two steps:

- E-Step Compute $Q(\theta,\theta^t)$
- M-Step Update $\theta^{t+1} = Argmax_\theta Q(\theta,\theta^t)$.

The above provides the general framework for EM algorithms; the actual details of the E-steps and M-steps require a derivation which is problem specific. The E-step and M-step of maximum liklihood estimates of frequency counts from randomized data are as the following:

- E-Step $LL(D|\theta) = \sum_{j=1}^{J} N_j log(\theta_j)$

$$Q(\theta,\theta^t) = E[LL(D|\theta)|\tilde{D};\theta^t] = \sum_{j=1}^{J} E[N_j|\tilde{D};\theta^t]\log(\theta_j) = \sum_{j=1}^{J} E[N_j|\tilde{N}_1,\tilde{N}_2,\cdots,\tilde{N}_J;\theta^t]\log(\theta_j)$$

$$E[N_j|\tilde{N}_1,\tilde{N}_2,\cdots,\tilde{N}_J;\theta^t] = \sum_{k=1}^{J} \frac{\theta_j^t P_{jk}}{\sum_{m=1}^{J}\theta_m^t P_{mk}}\log(\theta_j)$$

$$\text{so, } Q(\theta,\theta^t) = \sum_{j=1}^{J}\sum_{k=1}^{J} \frac{\theta_j^t P_{jk}}{\sum_{m=1}^{J}\theta_m^t P_{mk}} log(\theta_j)$$

• M-step Let $\frac{\partial Q(\theta;\theta^t)}{\partial \theta_j} = 0$ for $j = 1,\ldots,J$. Together with $\sum_{j=1}^{J}\theta_j = 1$ and

$\sum_{j=1}^{J}\sum_{k=1}^{J}\frac{\theta_j^t P_{jk}}{\sum_{m=1}^{J}\theta_m^t P_{mk}} = N$. We have $\theta_j = \frac{1}{N}\sum_{k=1}^{J}\frac{\theta_j^t P_{jk}}{\sum_{m=1}^{J}\theta_m^t P_{mk}}\tilde{T}_k$.

Specific EM Algorithm is as the following:

- Initialize $\theta^0 = \frac{\tilde{N}_X}{N}$

- Update θ as $\theta_j^{t+1} = \frac{1}{N}\sum_{k=1}^{J}\frac{\theta_j^t P_{jk}}{\sum_{m=1}^{J}\theta_m^t P_{mk}}\tilde{N}_k$ for $j = 1,\cdots,J$

- $t = t+1$

- If termination condition has not been met, return to the update step.

The termination criterion for the EM algorithm is based on how much θ^t has changed since the last iteration. The termination criterion used in our simulation is $\left\|\theta^t - \theta^{t-1}\right\| \le \varepsilon$, where ε is an application-dependent threshold.

The EM algorithm proposed here is similar to the EM reconstruction algorithm in [AA01]. However, their data is perturbed by additive noise while data in our framework is discretized and then post randomized. The convergence property of the above EM algorithm can be proved in a similar fashion [AA01].

It has been proved in [HH02, HH04], that the log-likelihood $LL(\tilde{D}\,|\,\theta)$ is from a regular exponential model. The log likelihood is therefore strictly concave; so finding the maximum should not pose any difficulties when the starting point is chosen in the interior of the parameter space and the maximum is also achieved in the interior.

The Maximum likelihood estimation produces estimates that are asymptotically Gaussian with covariance matrix $\frac{1}{N}M^{-1}$, i.e., $\hat{\theta} \sim N(\theta, \frac{1}{N}M^{-1})$ in which M is the Fisher information matrix $-E\{\frac{\partial^2}{\partial\theta} LL(\tilde{D}\,|\,\theta)\}$. The Maximum likelihood estimator is also consistent. From the asymptotic normality and consistency of maximum likelihood estimator, we can also conclude that maximum likelihood estimator of $\theta = \frac{N_X}{N}$ provide zero-error estimate if the sample size N of the data set D is large enough.

The moment estimation and the maximum likelihood estimation are related. Using the property that the log likelihood is from a regular exponential family, the following relation between these two estimates hold:

a. The maximum of $LL(\tilde{D}\,|\,\theta)$ is unique when it is in the interior of the parameter space.

b. The maximum likelihood estimate and moment estimate are equal when both are found in the interior of the parameter space.

The proofs are provided in [HH02, Lucy74]. [HH02] proved the above relationship between maximum likelihood estimator and moment estimator in the framework for analyzing misclassified data while [Lucy74] proved it in framework for measurement error with known error distribution.

2. 5 Experimental Results

The following two simulations are performed and the results are shown in figures

2.2 and 2.3 respectively.

(1) Generate 5000 samples from two categorical random variables with joint probability as shown in table 2.1. Both of the categorical variables can take 5 different categories. Each categorical value is randomized to every other categorical value with probability $\frac{p}{4}$, where $p = 0.5$ is used. The randomization is done independently and uniformly to the two variables. The frequency counts are estimated using both moment estimation and maximum liklihood estimation. Number 1 to 25 on horizontal axis of figure 2.2 correspond to the 25 possible joint categories while the numbers on the vertical axis are frequency counts of the categories.

X	1	2	3	4	5
1	$\frac{1}{40}$	$\frac{3}{10}$	0	$\frac{1}{60}$	0
2	$\frac{3}{40}$	0	$\frac{1}{30}$	$\frac{1}{20}$	$\frac{1}{20}$
3	0	$\frac{1}{10}$	$\frac{1}{30}$	0	0
4	$\frac{1}{20}$	0	0	$\frac{1}{60}$	$\frac{1}{10}$
5	$\frac{1}{20}$	0	$\frac{1}{30}$	$\frac{1}{60}$	0

Table 2.1: Joint PMF of $X = \{X_1, X_2\}$

(2) Generate 100,000 random samples from a two-triangle distribution. MGAS algorithm is used to discretize the variable with minimum number of records 1,500 and minimum granularity 0.25. 39 groups result from the MGAS algorithm. Every sample is replaced by the mean of the group it belongs to. Each sample is then randomized to other categories (mean of other groups) with probability $\frac{1-p}{38}$, where $p = 0.3$ is used. Moment estimator and maximum likelihood estimator are used to estimate the number of records in each group, i.e. a set of frequency counts. The distribution after MGAS and

distribution reconstructed using maximum likelihood estimation are shown in figure 2.3.

From the above two simulation, we can see that both moment estimator and maximum liklihood estimator accurately estimates the probability mass function and probability distribution function. Moment estimator may produce negative frequency counts estimate. We can use the estimated frequency counts to learn models without accesing the original data.

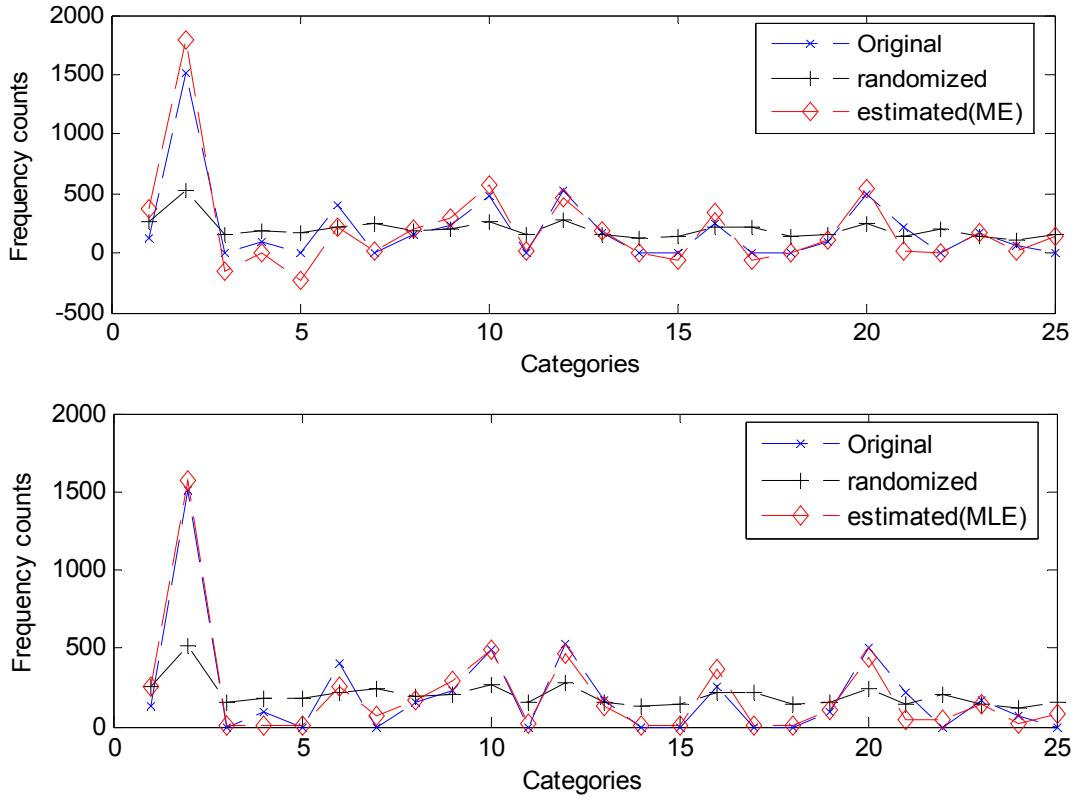


Figure 2.2 Frquency counts estimation

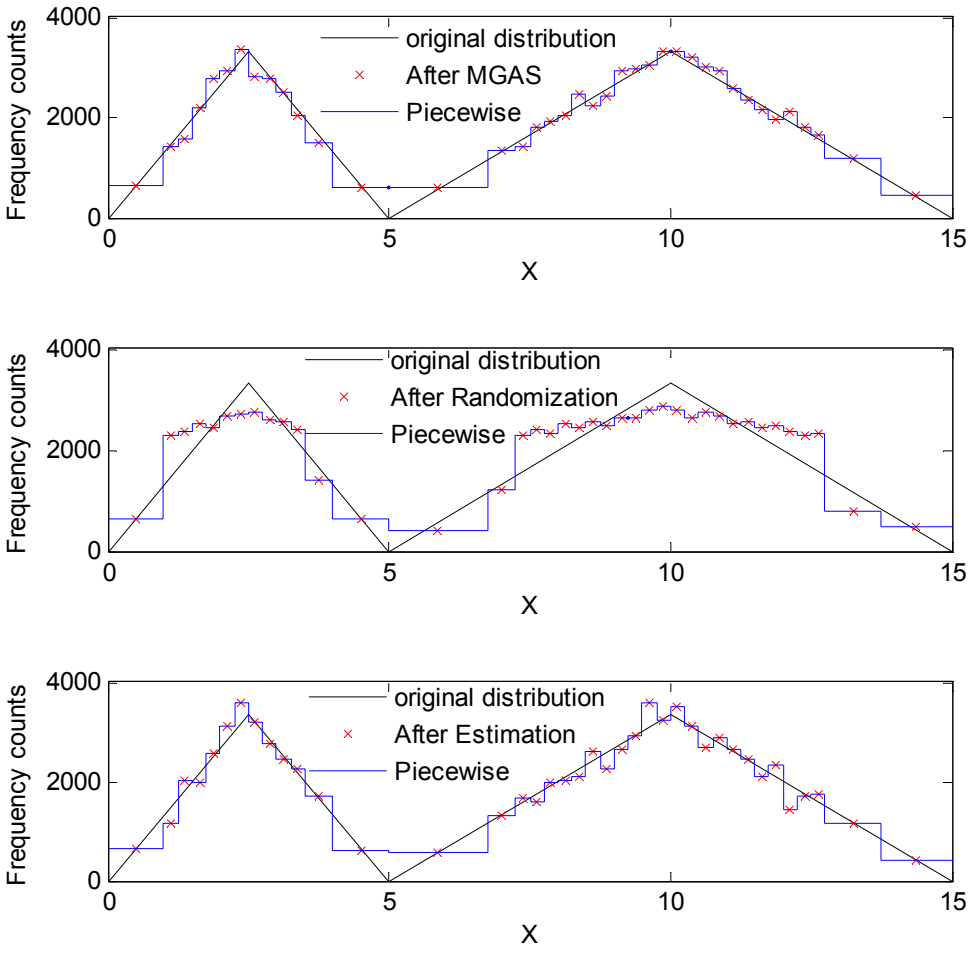


Figure 2.3 Distributions after MGAS discretization, randomization and reconstruction

CHAPTER THREE

QUANTIFICATION OF PRIVACY AND ACCURACY

In this chapter, quantification of privacy and accuracy is discussed. Quantification of privacy of PRAM is discussed in section 3.1 and quantification of accuracy is discussed in section 3.2. There exists trade off between privacy and accuracy in PPDM using PRAM. Simulation results are provided in section 3.3 to illustrate the trade off between privacy and accuracy.

3.1 Quantification of Privacy

Recently researchers have found that it is difficult to use a single measure to quantify the different aspects of privacy. We have to use several measures to quantify privacy. Good privacy in one measure may not be as good under a different measure. For example, individual privacy breaches will happen even if the privacy is good under an entropy-based measure. Different privacy quantification methods are proposed in the literature. In [AS00], the privacy provided by a reconstruction-based technique is measured by evaluating how closely the original values of a modified attribute can be determined. If one can estimate with c% confidence that a value lies in an interval, then the width of such interval defines the amount of privacy with a c% confidence level. Privacy metrics proposed by [AA01] take into account the fact that the perturbed individual record, the reconstructed distribution and the perturbing distribution are available to the user. The privacy metric they proposed is based on mutual information between original and perturbed records. The average conditional privacy of an attribute A

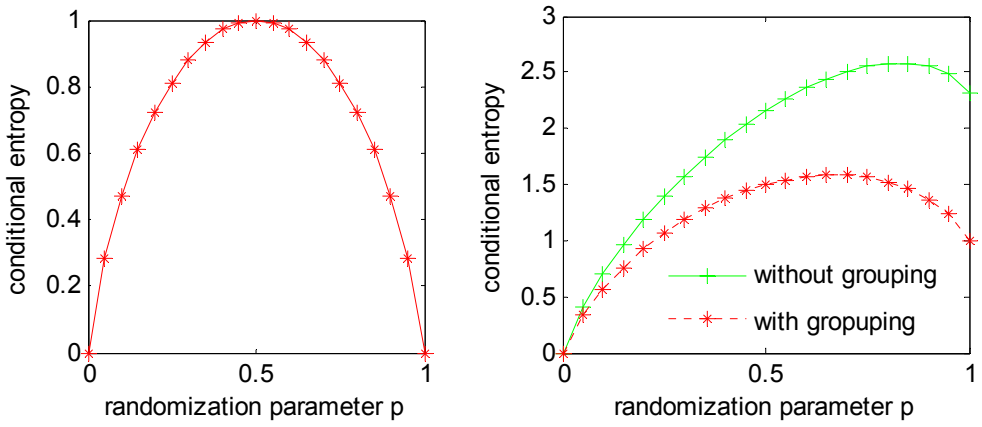
given some other information, modeled by a random variable B, is defined as $2^{H(A|B)}$, where $H(A|B)$ is the conditional entropy of A given B. Evfimievski et al.[EGS03] provide a formal definition of privacy breaches and a concept of $\gamma-$amplification to limit the privacy braches. We discuss in this chapter privacy quantification of PRAM using entropy-based metrics, $\gamma-$amplification, and probabilistic K-anonymity.

3. 1.1 Conditional Entropy-based Metrics

Conditional entropy provides a measure of privacy preserved on average using PRAM; more specifically it gives a measure of how much information we have about the original variable X_i given the knowledge of the randomized variable $\tilde{X}_i$ and the probability transition matrix P. The conditional entropy-based privacy measure of PRAM schemes is defined as

$H(X_i \mid \tilde{X}_i) = -\sum_{k=1}^{K_i} \sum_{k'=1}^{K_i} p(X_i = k')P(k',k) log_2 \frac{p(X_i=k')P(k',k)}{\sum_{k'=1}^{K_i} p(X_i=k')P(k',k)}$, where $p(X_i = k')$ is the prior probability that X_i takes k' th category and $P(k',k)$ is the transition probability that k' th category is randomized to k th category. The larger the conditional entropy-based privacy measurement, better the privacy is preserved on average. The metric of privacy using conditional entropy depends on the prior distribution of the variable of X_i, i.e. $p(X_i = k')$. Figure 3.1(a) gives the conditional entropy-based privacy measure $H(X_i \mid \tilde{X}_i)$ versus the randomization parameter p for a binary variable X_i (assuming uniform prior distribution), which is randomized to $\tilde{X}_i$ by using binary symmetric randomization scheme with parameter p. Figure 3.1(b) gives the conditional entropy-

based privacy measure $H(X \mid \tilde{X})$ versus the randomization parameter p for a variable X_i with cardinality 6 (assuming uniform prior distribution), where X_i is randomized to $\tilde{X}_i$ by using multi-category randomization scheme with parameter p. Figure 3.1(b) also shows the entropy-based privacy measure $H(X \mid \tilde{X})$ when 6 categories of X_i are grouped into two groups with 3 categories each and each group is randomized using multi-category randomization scheme with parameter p. From Figure 3.1(b), we can see less privacy is preserved on average by PRAM with grouping of categories than by PRAM without grouping the categories.



(a) (b)

Figure 3.1: Conditional-entropy based privacy measurement versus randomization parameter p

3.1.2 γ - Amplification

We consider the notion of privacy introduced by [EGS03] in terms of an amplification factor γ. The γ-amplification in [EGS03] is proposed in the framework that every data record should be randomized with a factor less than γ, to limit the privacy breach, before the data are sent to the data miner. However, we use the

amplification γ purely as a worst-case quantification of privacy for a designed PRAM. A PRAM operator for variable X_i with transition probability P^i is at most γ-amplifying for $\tilde{X}_i = k$ if $\forall k_1, k_2 \frac{P^i(k_1,k)}{P^i(k_2,k)} \le \gamma$, where $k, k_1, k_2 \in S_i$ and $|S_i| = K_i$. A PRAM operator is at most γ-amplifying for variable X_i if it is at most γ-amplifying for any $k \in S_i$. An upward ρ_1 -to- ρ_2 privacy breach occurs when the posterior belief $p(X_i = k' \mid \tilde{X}_i = k) \ge \rho_2$, while the prior belief $p(X_i = k') \le \rho_1$. A downward ρ_2-to-ρ_1 privacy breach occurs when the posterior belief $p(X_i \ne k' \mid \tilde{X}_i = k) \ge \rho_1$, while the prior belief $p(X_i \ne k') \le 1 - \rho_2$. If the randomization operator is at most γ-amplifying for X_i, revealing $\tilde{X}_i$ will cause neither an upward ρ_1-to-ρ_2 privacy breach nor a downward ρ_2-to-ρ_1 privacy breach if $\frac{\rho_2(1-\rho_1)}{\rho_1(1-\rho_2)} > \gamma$. For details of γ-amplification, readers can refer to [EGS03]. Clearly, the smaller the value of γ, the better the worst case privacy. Ideally we would like to have $\gamma = 1$ since by definition $\gamma \ge 1$. For binary symmetric randomization introduced in Chapter 2, if $0 \le p_1 = p_2 < 0.5$, then it is easy to see that it is at most γ-amplifying for $\gamma = \frac{1-p}{p}$. For the ternary symmetric randomization introduced in Chapter 2, if $1 - p_1 - p_2 \ge p_1 \ge p_2$, then amplification is at most $\gamma = \frac{1-p_1-p_2}{p_2}$. For the multi-category randomization scheme, the amplification is at most $\gamma = \frac{1-p}{p/(|X|-1)}$ if $1 - p \ge \frac{p}{|X|-1}$. For the variable with 6 category we take for example in figure 3.1, the randomization scheme without grouping has $\gamma = \frac{5(1-p)}{p}$ if $1 - p \ge \frac{p}{5}$ and the

randomization scheme with grouping has $\gamma = \frac{2(1-p)}{p}$ if $1-p \geq \frac{p}{2}$. We can see that the randomization with grouping have smaller γ (better privacy in worst-case sense) than that without grouping though the conditional entropy is also smaller (worse privacy on average).

The γ-amplification provides a worst case quantification of privacy which is a privacy measure that does not depend on the prior distribution of the data. γ-amplification privacy metric does not provide any information of privacy preserved on average. For a given γ and prior belief ρ_1, we can get a ρ_2^* such that $\frac{\rho_2^*(1-\rho_1)}{\rho_1(1-\rho_2^*)} = \gamma$ and we will not have a privacy breach with posterior belief $\rho_2 > \rho_2^*$. Figure 3.2 below gives the (ρ_2^*, ρ_1) pair for several values of γ.

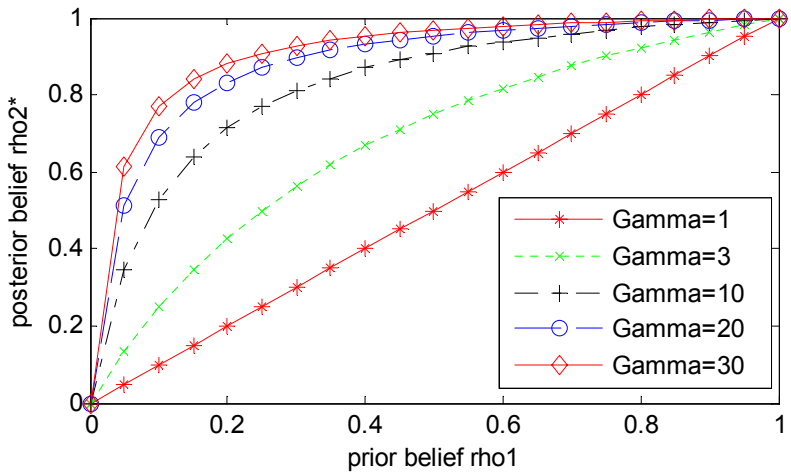


Figure 3.2: (ρ_1, ρ_2^*) pair

We mentioned in chapter 1 that the randomization scheme using discretization first followed by PRAM to numerical variables usually has smaller γ-amplification than randomization using additive noise. Discretization and then PRAM also provides a

flexible way to control γ-amplification. Value distortion using additive noise adds a random value r drawn from some distribution to the original value x_i. In [AS00], uniformly distributed additive noise and Gaussian additive noise are considered. The data values $x_1 \cdots x_N$ can be viewed as realizations of N independent identically distributed random variables X. We denote the randomized variable as $\tilde{X} = X + R$. We denote M_1 as the number of intervals the support (domain) of variable $\tilde{X}$ is divided into during the subsequent discretization in reconstruction of the distribution and M_2 as the number of intervals that are also support (domain) of the original variable X among M_1 intervals. Obviously $M_1 > M_2$. We can calculate a $M_1 \times M_1$ probability transition matrix from the distribution of R. The γ-amplification for those intervals in the margin of the support of variables $\tilde{X}$ will inevitably be very large and usually be infinite. For example, the γ-amplification for the last interval, which corresponds to the interval with the largest values of $\tilde{X}$, will be usually be infinite, since it usually can be got only by randomizing the value in the largest intervals of X and adding the largest possible values of the random variable R. This actually is the reason why privacy breaches discussed in section 1.3.3 occurs. If discretization and then PRAM is used instead, we can discretize variable X into M_2 intervals and then use PRAM with a $M_2 \times M_2$ probability transition matrix. In this way, the above problem regarding additive noise will not exist. The reconstruction method is similar to that for additive noise randomization method. Hence, we can say that discretization followed by PRAM has a smaller γ-amplification and can limit more privacy breaches for the same reconstruction accuracy. Furthermore, PRAM provides a way to control the γ-amplification and thus control possible privacy breaches by design

of the $M_2 \times M_2$ probability transition matrix, for example, implementation of PRAM by grouping the M_2 intervals into several groups.

3.1.3. Probabilistic K-anonymity

The γ-amplification does not provide any information of privacy preserved in general for a PRAM scheme and is independent of the prior distribution. However, in heterogeneously distributed data sets, it is not difficult for the party who owns the vertical part of the data corresponding to a variable to have some idea about the prior distribution of the variable. It turns out that γ-amplification provides a very conservative measure of privacy if the knowledge of prior distribution is available. Entropy-based metric only provides an average measure of privacy and does not pay attention to privacy breaches to individual values of a variable.

Besides γ-amplification, we can use a probabilistic K-anonymity pair $(K_p, Dist_K)$ to quantify privacy preserved by PRAM schemes. The definition of the probabilistic K-anonymity pair is as follows:

Let X be a (possibly combined) variable with *J* categories, *X* is randomized to $\tilde{X}$ by using a PRAM scheme. We say privacy preserved by the PRAM scheme is at least $(K_p, Dist_K)$ if $K_p = \min_{k'} \#\{k \mid p(X = k \mid \tilde{X} = k') > 0\}$ and $Dist_K = \min_{k'} KL(p(X = k \mid \tilde{X} = k'), Unif(X))$, where $\#\{k \mid p(X = k \mid \tilde{X} = k') > 0\}$ is the number of source categories k such that $p(X = k \mid \tilde{X} = k') > 0$, i.e. the posterior probability of $X = k$ given $\tilde{X} = k'$ is greater than 0 for a designed PRAM and $KL(p(X = k \mid \tilde{X} = k'), Unif(X))$ is the Kullback-Leibler Distance between posterior

distribution $p(X = k \mid \tilde{X} = k')$ and a discrete uniform distribution over K_p categories. Definition of K_p is similar to the K defined in K-anonymity in [Swe02a], however, our definition of K_p extends the definition of K-anonymity into probabilistic cases. $Dist_K$ here gives a measure of the degree of probabilistic K-anonymity.

3.2 Information Loss Analysis

There is always some information loss introduced due to the randomization. Information loss has two aspects. One is related to dependence between variables, which can be measured on average by distance between estimated and original distributions. The other information loss is related to the independence between variables. Independence between variables is also important in data mining.

3.2.1 Distance between Two Sets of Counts

Given the post randomized data, it is usually not possible to estimate frequency counts with an arbitrary precision. The more randomization introduced, the lower is the precision of the estimation of frequency counts. The frequency counts and their estimates are naturally two distributions of the corresponding variables. We can use the Kullback-Leibler distance as a measure of information loss. This information loss measure does not include the information loss caused by discretization. If we want to include the information loss due to discretization, we can use the $\int_{\Omega_X} \left| f_X(x) - \hat{f}_X(x) \right|$ to quantify the information loss, where $f_X(x)$ is the original density function of X and

$\hat{f}_X(x) = \frac{1}{N}\sum_{i=1}^{J}\hat{T}_X(i)I_{\Omega_i}(x)$, where $I_{\Omega_i} = \begin{cases} 1 & x \in ith \quad interval \\ 0 & otherwise \end{cases}$.

3.2.2 Independence Loss Analysis

One important information loss due to PRAM is the independence loss of two originally independent variables. We use two independent binary categorical variables X_1 and X_2 to explain this kind of loss. Suppose X_1 and X_2 takes value in $\{1,2\}$, the independence of the two variables implies that $P(X_2 = 1 | X_1 = 1) = P(X_2 = 1 | X_1 = 2)$ and $P(X_2 = 2 | X_1 = 1) = P(X_2 = 2 | X_1 = 2)$. One traditional measure of this kind of association (independence) is odds ratio r, which is the ratio between $\frac{P(X_2 = 1; X_1 = 2)}{P(X_2 = 1; X_1 = 1)}$ and $\frac{P(X_2 = 2; X_1 = 2)}{P(X_2 = 2; X_1 = 1)}$. A sample odds ratio is $r = \frac{n_{11}n_{22}}{n_{12}n_{21}}$, where n_{ij} is the number of samples such that $X_1 = i$ and $X_2 = j$ for $i, j = 1,2$. If the two variables are independent, the odds ratio should be 1. So, the sample odds ratio should be close to 1 when there are enough samples. If the data are randomized and the frequency counts are estimated from the randomized data, the sample odds ratio using those estimated frequency counts becomes $\hat{r} = \frac{\hat{n}_{11}\hat{n}_{22}}{\hat{n}_{12}\hat{n}_{21}}$, where $\hat{n}_{ij}$ is the estimate of n_{ij}.

We used Monte Carlo simulation to see the variance of estimated odds ratio. 10,000 samples of two independent binary variables are generated. We applied PRAM to the samples using binary symmetric randomization with parameter $p = 0.4$. We calculated the sample odds ratio r and the sample odds ratio $\hat{r}$ using estimated frequency counts. The above randomization and calculation is repeated 2,000 times independently. Figure

3.3 (a) and (b) shows the histogram of r and $\hat{r}$ obtained from the 2,000 independent runs respectively.

From the simulation, we can see that the sample odds ratio using the estimated frequency counts is more widely spread. This suggests that the original independent random varibles will be judged as dependent variables with high probability if the same independence testing is used. In other words, independence relationship is sensitive to randomization while the dependence relationship is more robust to the randomization. Special care has to be given to this kind of independence loss. This kind of independence loss is further discussed in Chapter 5, where independence loss causes many extra links in learning Bayesian network using estimated sufficient statistics.

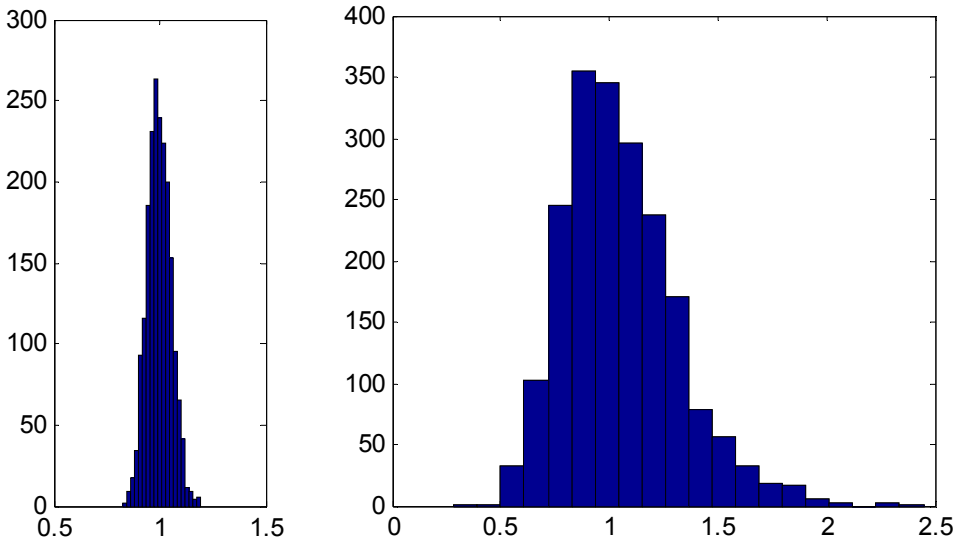


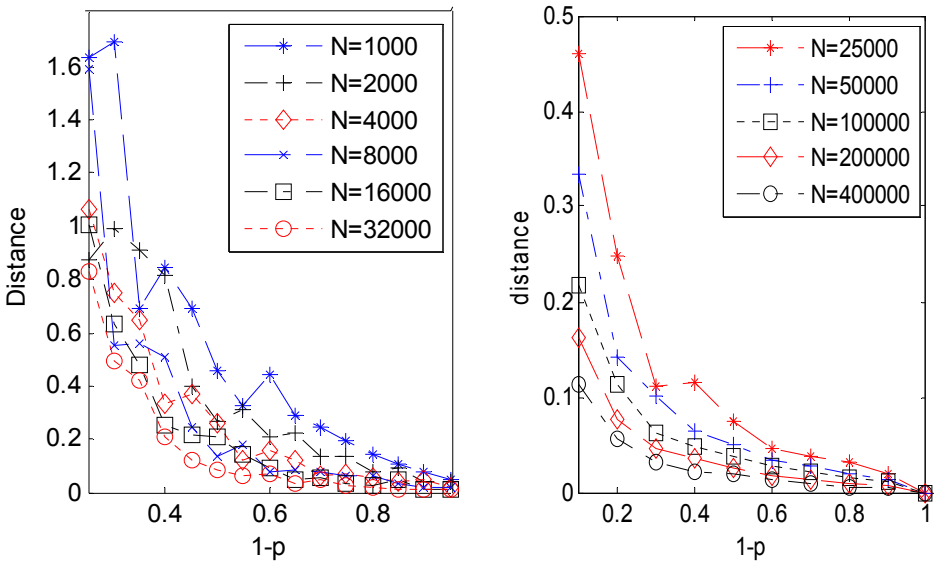
Figure 3.3 Histogram of Odds Ratio

3.3. Trade off between Accuracy and Privacy

There is a trade off between accuracy and privacy for a given size of sample

dataset. Training sample size is key in the trade off between accuracy and privacy. The goal of PPDM using PRAM is to achieve reasonable privacy with reasonable accuracy. Simulations are done to the two joint categorical variables discussed in Chapter 2 using *M*-category randomization with parameter $1-p$ from 0.25 to 0.95 (Each category is randomized to other categories with probability $\frac{p}{4}$) and sample size N from 1000 to 32000. The K-L distance between the estimated joint PMF using estimated frequency counts (maximum liklihood estimation using EM algorithm) and the original PMF is shown in figure 3.4 (a). Simulations are also done to the two triangle distribution in section 2.5 using *M*-category randomization with parameter $1-p$ from 0.1 to 1 (Each category is randomized to other categories with probability $\frac{p}{|X|-1}$, where $|X|$ is the number of intervals the variable is discretized by MGAS algorithm) and sample size N from 25000 to 400000. The distance $\int_{\Omega_X} \left| f_X - \hat{f}_X \right|$, the distance between the estimated distribution (reconstructed using MLE of frequency counts which are got by using EM algorithm) and the original distribution is used. The simulation results are shown in Figure 3.4 (b).

From the above simulation, we can see that sample size is key in the trade off between accuracy and privacy. If sample size is large enough, both accuracy and privacy can be achieved.



(a) (b)

Figure 3.4: Trade Off between Accuracy and Privacy

CHAPTER FOUR

FRAMEWORK FOR PPDM USING PRAM TECHNIQUE

In this Chapter, the framework for PPDM using PRAM is introduced. This framework can be applied in learning different kinds of models from randomized data for PPDM. We introduce the framework for learning models that depends on summary of data in section 4.1. Learning models depending on optimization of a cost function is introduced in section 4.2. Specific data mining models/applications of PPDM using PRAM are introduced in subsequent chapters.

4.1Framework for Learning Models Based on Summary

For many models, the information required from data is often a summary of the data, which is usually a function of the frequency counts, say $g(N_X)$. When the data are post randomized, the data miner can use $g(\hat{N}_X)$ instead of $g(N_X)$ in the learning process. If $\hat{N}_X$ is a maximum likelihood estimator of N_X, $g(\hat{N}_X)$ is also a maximum likelihood estimation of $g(N_X)$. In Bayesian network learning, the score function in structure learning depends on a set of frequency counts for each candidate structure and the parameter learning depends on the set of frequency counts of the learned structure. Privacy-preserving Bayesian network learning using PRAM is discussed in chapter 5. The main task in building a decision tree is to identify an attribute for the splitting point based on information gain. The information gain is usually computed using entropy. The computations of entropy are based on the frequency counts of split data sets. The key part of association rule mining is to find the frequent item sets. Finding frequent item set

depends solely on computation of frequency counts. Privacy-preserving decision tree learning and association rule mining are discussed in chapter 7.

4.2 Framework for Learning Models Based on Cost Minimization

In many other learning problems, the objective is to find some parameter w that minimizes a specified cost function or maximizes some likelihood function. Suppose $Y_1 = y_1 = \{y_{11}, y_{12}, \ldots, y_{1m}\}, \ldots, Y_N = y_n = \{y_{N1}, y_{N2}, \ldots, y_{Nm}\}$ are N samples of $X = \{X_1, X_2, \ldots, X_m\}$. The joint distribution of X is denoted as $P_X(x)$ and $\tilde{Y}_1 = \tilde{y}_1 = \{\tilde{y}_{11}, \tilde{y}_{12}, \ldots, \tilde{y}_{1m}\}, \ldots, \tilde{Y}_n = \tilde{y}_n = \{\tilde{y}_{N1}, \tilde{y}_{N,2}, \ldots, \tilde{y}_{Nm}\}$ are post randomized values of $Y_1 = y_1 = \{y_{11}, y_{12}, \ldots, y_{1m}\}, \ldots, Y_N = y_n = \{y_{N1}, y_{N2}, \ldots, y_{Nm}\}$. $\tilde{X}$ denotes the corresponding randomized variable of X . The cost-based learning problem is to find parameters w such that $E[J(X, w)]$ is minimized. The sample version of the problem is to find w which minimize $\sum_{i=1}^{N} J(y_i, w)$. Because $E[J(X, w)] = E\{E[J(X, w) \mid \tilde{X}]\}$, it is reasonable to use $\sum_{i=1}^{N} E[J(Y_i, w) \mid \tilde{y}_i]$, which is a sample version of $E\{E[J(X, w) \mid \tilde{X}]\}$, instead of $\sum_{i=1}^{N} J(y_i, w)$ as an objective function since $y_1, \ldots, y_i$ is not available to the data miner but $\tilde{y}_1, \ldots, \tilde{y}_N$ is. Suppose the variable X has been discretized into J intervals and post randomized by the data owner. We have $\sum_{i=1}^{N} E[J(Y_i, w) \mid \tilde{y}_i] = \sum_{i=1}^{N} \{\sum_{j=1}^{J} [J(m_j, w) P(Y_i = m_j \mid \tilde{y}_i = m_i)]\}$, where m_j is the mean value of each interval and $P(Y_i = m_j \mid \tilde{y}_i = m_i)$ is the posterior probability that $Y_i = m_j$ given the observed data is $\tilde{y}_i = m_i$. $P(Y_i = m_j \mid \tilde{y}_i = m_i)$ can be estimated using the

estimated frequency counts, i.e. $P(Y_i = m_j \mid \tilde{y}_i = m_i) = \dfrac{\hat{T}_j P(j,i)}{\sum_{m=1}^{J} \hat{T}_m P(m,i)}$ where and $\hat{T}_j$ is the estimator of the frequency counts of interval j and $P(j,i)$ is the $(j,i)th$ element of probablity transition matrix P used to randomize the discretized variable. The problem now is to get w such that $\sum_{i=1}^{N}\{\sum_{j=1}^{J}[J(m_j, w)\sum_{m=1}^{J}\dfrac{\hat{T}_j P(j,i)}{\hat{T}_m P(m,i)}]\}$ is minimized. It may be easy to get analytical solution for simple cases, for example , trivial cost functions, low dimensions of variables, and when variables take values from a small set. However, it is not a trivial task in most cases. Iterative method can be used to obtain a solution. The iterative method can be implemented simultaneously with a maximum likelihood estimator of frequency counts. We can also implement the estimation of frequency counts and optimization of the cost function sequentially. In chapter 6, this learning framewrok is applied to learn linear classifier from randomized data.

CHAPTER FIVE

PRIVACY-PRESERVING BAYESIAN NETWORK LEARNING USING PRAM

In this chapter, we discuss privacy-preserving Bayesian network learning from heterogeneously distributed database using PRAM. Both the case when there is a data miner and the case when there is not a data miner is discussed.

5.1 Introduction to Bayesian Network Learning

A Bayesian network is an annotated directed acyclic graph that encodes a joint probability distribution of a set of random variables $U = \{X_1, X_2 \cdots X_n\}$. Formally, a Bayesian Network for U is a pair $B =< G, \theta >$, where G is a directed acyclic graph and θ represents the set of conditional probabilities of the variables. Vertices of graph G correspond to variables and edges of G represent direct dependencies between the variables $X_1,\ldots, X_n$. We consider discrete variables, that is, each variable X_i takes values from a finite set S_i. The problem of Bayesian network learning is to find a network G that best matches the given training data set $D = \{y_1, y_2, \cdots, y_N\}$, where each record y_i is an instance of variables $\{X_1, X_2, \cdots, X_n\}$. We consider a heterogeneously distributed database, where each observation is distributed among two or more parties, with each party observing a subset of the variables $\{X_1, X_2, \cdots, X_n\}$ (vertically partitioned database). Learning Bayesian network has two parts: Bayesian network structure learning

and Bayesian network parameter learning. For parameter learning, the structure of the network is assumed known. We discuss both parameter and structure learning. Parameter learning problem can be stated as follows: Given a data set D, compute the posterior distribution $p(\theta \mid G, D)$, where G is the known structure. Two widely used algorithms to estimate the parameters from data D (and prior knowledge if available) are maximum likelihood (ML) and maximum a posterior (MAP) method. The ML estimator of θ, which maximizes the sample likelihood $p(D \mid \theta)$ with respect to θ, is: $\theta_{ijk}^{ML} = \frac{N_{ijk}}{N_{ij}}$, where N_{ijk} is the number of records such that variable X_i is in its k th configuration and its parents $Pa(X_i)$ are in the j th configuration and $N_{ij} = \sum_{k=1}^{K_i} N_{ijk}$, where K_i is the cardinality of variable X_i. The MAP estimator is: $\theta_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}$, where we assume the prior distribution of θ_{ij} is Dirichlet with parameters $\{\alpha_{ij1}, \alpha_{ij2}, \cdots, \alpha_{ijK_i}\}$. The Dirichlet distribution is the conjugate prior of the parameters of the multinomial distribution. A popular approach to learning Bayesian network structure is to introduce a scoring function that evaluates the "fitness" of the structure to the sample data. The two commonly used score functions are the Bayesian score and the one based on the principle of minimum description length (MDL). These scoring functions depend only on the sufficient statistics N_{ijk} s of each candidate structure. All the required information for the parameter and structure learning are contained in the sufficient statistics N_{ijk} s of each candidate structure (the fixed structure G only for parameter learning) from the training data D. Therefore, the problem of (privacy-sensitive) Bayesian network learning is equivalent to the problem of calculating the sufficient statistics (in a privacy-sensitive manner). In our method, we estimate those sufficient statistics from the randomized data.

5.2 A Framework for Privacy-preserving Bayesian Network Learning Using PRAM

In this section, we introduce the framework for privacy-preserving Bayesian network learning using PRAM. A framework for parameter learning is introduced in section 5.2.1. A framework for structure learning is introduced in section 5.2.2, which is based on the framework for parameter learning.

5.2.1 Parameter Learning

For parameter learning, the structure G is assumed fixed and known to every party. For the case when there is no data miner, we need the following two definitions from [CSK04]:

- **Definition 1:** (Cross Variable) If a variable X and some of its parents are with different parties, then X is called a cross variable; otherwise it is called a local variable.
- **Definition 2:** (Cross parent of site a_i for site a_j) A variable with party a_i who is a parent of some cross variable of party a_j is called a cross parent of party a_i for party a_j.

We can see that sensitive cross parents are the only variables that need to be randomized in this case. Learning Bayesian network parameters for this case can be done as follows:

For each party a_i,

(1) Randomize cross parents belonging to its own party according to their respective privacy requirements using PRAM as described in Chapter 2. Randomizations are done independently for each (combined) variable and each record.

(2) Send randomized cross parents of party a_i for party a_j to party a_j together with the probability transition matrix used.

(3) Learn parameters for local variables of party a_i. This step does not involve randomized data.

(4) Estimate the sufficient statistics N_{ijk}s for each cross variable at same site a_i using the local data and randomized parent data from other parties.

(5) Estimate the parameters for cross variables using the estimated sufficient statistics $\hat{N}_{ijk}$.

(6) Share the parameters with all other parties.

Parameter learning for the case when there is a data miner can be done as described in the following:

For each party a_i,

(1) Randomize all its sensitive variables according to their respective privacy requirements using PRAM as described in Chapter 2. Randomizations are done independently for each (combined) variable and each record.

(2) Send randomized data and their corresponding probability transition matrices to the data miner.

For the data miner,

(1) Estimate the sufficient statistics N_{ijk}s for each node X_i using the randomized data and probability transition matrices.

(2) Estimate the parameters using the estimated sufficient statistics $\hat{N}_{ijk}$s.

(3) Broadcast the parameters to all parties.

The details of estimation of sufficient statistics N_{ijk} and parameter learning using estimated sufficient statistics $\hat{N}_{ijk}$ (step 4 and 5 above in the case when there is not a data miner, step 1 and 2 for the data miner in the case when there is a data miner) are described in sections 5.3 and 5.4 respectively.

5.2.2 Structure Learning

During the search of a DAG (Structure) that best fit the data, perform randomization and estimation of sufficient statistics as described in section 5.2.1 for each candidate structure as a fixed structure G. The randomization should be done in a way that prevents multiple versions of randomized data since multiple versions of randomized data will degrade privacy preserved by PRAM. Use those estimated sufficient statistics to calculate the score of the candidate structure G. Then, we can choose a structure with maximum score. We use K-2 algorithm to search for a DAG that approximately has the maximum score. The details of structure learning from randomized data using K-2 algorithm are given in section 5.5.

5.3. Estimation of Sufficient Statistics from Randomized Data

The problem of privacy-preserving Bayesian network learning can be decomposed into a series of estimation of N_{ijk} s for each node X_i and each candidate structure, which can be seen from section 5.2. The parents $Pa(X_i)$ of Node X_i are given by a (candidate) structure G.

Consider the following general case: The cardinality of Node X_i is K_i and it has Q parent nodes $Pa_i(1), Pa_i(2), \cdots, Pa_i(Q)$ in the candidate structure. The cardinality of each parent $Pa_i(q)$ is $K_{Pa_i(q)}$. These variables can be arbitrarily vertically partitioned to different parties in both setups. The randomization of each (combined) variable can also be done by grouping the categories of the variable into groups. Our discussion below is independent of the specific partitioning and grouping of the variables.

We have the following different cases for estimating N_{ijk} s from the randomized data $\tilde{D}$ due to simultaneous randomization. Note that only the variables in the same party can be randomized simultaneously.

(a) Node X_i and all of its parents are randomized independently to each other.

(b) Some parents of Node X_i are randomized simultaneously.

(c) Node X_i is randomized simultaneously with some of its parents.

(d) Node X_i is randomized simultaneously with other (not its parent) nodes.

For cases (b) and (c), we can consider the simultaneously randomized variables as a combined variable. For example, if node X_i is randomized simultaneously with one of its parents $Pa_i(1)$, N_{ijk} is equal to the number of records such that $(X_i; Pa_i(1)) = (k; j^1), Pa_i(2) = j^2, \cdots, Pa_i(Q) = j^Q$, where $(X_i; Pa_i(1))$ is a combined variable. Thus, we can estimate the N_{ijk} s from the randomized data by treating $(X_i; Pa_i(1))$ as a single variable with cardinality $|K_i| \times |K_{Pa_i(1)}|$. For case (d), since the current N_{ijk} does not involve the variable randomized simultaneously with X_i, we can get the marginal probability transition matrix from the given probability transition matrix,

which is for the combined variable. Hence, without loss of generality, we can consider case (a) only, that is Node X_i and its parents $Pa_i(1), Pa_i(2), \cdots, Pa_i(Q)$ are randomized independently to each other.

Suppose the probability transition matrices of X_i and its parents are $P^i, P^{Pa_i(1)}, \ldots, P^{Pa_i(Q)}$ respectively. The probability transition matrix for non-randomized variable is an identity matrix. The problem here is to estimate the sufficient statistics N_{ijk} from the randomized data. We denote by $Pa(X_i)$ as a compound variable for all the parents of Node X_i. Hence $Pa(X_i)$ takes $J_i = \prod_{q=1}^{Q} K_{Pa_i(q)}$ different values. The result of estimating frequency counts introduce in chapter 2 can be directly used here. N_{ijk} and N_{ij} are as defined in section 5.1. N_i is the $J_i K_i$ dimensional vector of N_{ijk} values, that is $N_i = (N_{i11}, N_{i12}, \cdots, N_{i1K_i}, N_{i21}, \cdots, N_{iJ_iK_i})^t$, where superscript t denotes matrix transpose and N_{ijk} is the $(K_i(j-1)+k)$th element of N_i. $N_i(l)$ for $1 \le l \le J_i K_i$ is the number of records that $\{X_i, Pa(X_i)\}$ have the l th configuration, where $l = K_i(j-1)+k$ for some j and k such that X_i is in k th configuration while $Pa(X_i)$ in j th configuration. $N_i(l)$ is actually one of the $N_{ijk}s$. $\tilde{N}_{ijk}$, $\tilde{N}_{ij}$ and $\tilde{N}_i$ are defined in a same way as N_{ijk}, N_{ij} and N_i are defined except that they are defined on the randomized data $\tilde{D}$. $\hat{N}_{ijk}$, $\hat{N}_{ij}$, and $\hat{N}_i$ are the estimates of N_{ijk}, N_{ij}, and N_i respectively.

Given the training data set D with N records, a candidate structure G and a randomization scheme characterized by probability transition matrices $P^i, P^{Pa_i(1)}, \ldots, P^{Pa_i(Q)}$, the theorems of estimating frequency counts introduced in chapter 2 directly applies. We have the following theorems for estimation of sufficient

statistics N_{ijk} for learning Bayesian Network. These theorems are a specific version of theorems we proved in Chapter 2.

Theorem 5.1

(a) $E[\tilde{N}_i \mid D] = P^t N_i$, where $P = P^i \otimes P^{pa^i}$ and $P^{pa^i} = P^{pa^i(1)} \otimes P^{pa^i(2)} \otimes \cdots \otimes P^{pa^i(Q)}$, $\otimes$ denotes Kronecker matrix product.

(b) For $l = 1, 2, \cdots, J_i K_i$, $\tilde{N}_i(l)$ is a random variable which is the sum of $J_i K_i$ independent Binomial random variables, that is, $\tilde{N}_i(l) = \sum_{m=1}^{J_i K_i} Y_{ml}^i$, where Y_{ml}^i is a binomial random variable that gives the number of records that is randomized to the configuration corresponding to $\tilde{N}_i(l)$ (the same as configuration corresponds to $N_i(m)$)from the configuration corresponding to mth element of $N_i(m)$. Furthermore, $Y_{ml}^i \sim B(N_i(m), \pi)$, where $\pi = P(m,l)$, the (m, l)th element of the matrix P and B denotes Binomial. Moreover, $Cov\{Y_{ml_1}^i, Y_{nl_2}^i\} = \begin{cases} \text{var}\{Y_{ml_1}^i\} = N_i(m)P(m,l_1)(1-P(m,l_1)) & if \quad n = m, l_1 = l_2 \\ -N_i(m)p(m,l_1)p(m,l_2) & if \quad n = m, l_1 \neq l_2 \\ 0 & if \quad n \neq m \end{cases}$

(c) The covariance matrix $Cov[\tilde{N}_i \mid D] = \sum_{l=1}^{J_i K_i} N_i(l) V_l$, where V_l is a $K_i J_i \times K_i J_i$ covariance matrix such that its $(l_1, l_2)th$ element $V_l(l_1, l_2) = \begin{cases} P(l,l_1)(1-P(l,l_1)) & if \quad l_1 = l_2 \\ -P(l,l_1)P(l,l_2) & if \quad l_1 \neq l_2 \end{cases}$.

The following theorem establishes the bias and variance of the estimator $\hat{N}_i = (P^t)^{-1} \tilde{N}_i$.

Theorem5.2

$\hat{N}_i = (P^t)^{-1}\tilde{N}_i$ is an unbiased estimator for N_i and $Cov\{\hat{N}_i \mid D\} = (P^{-1})^t Cov\{\tilde{N}_i \mid D\}(P^{-1})$.

Besides the two above theorems that directly follow from theorems in chapter 2, $\hat{N}_{ijk}$ can also be well approximated by a normal variable for the same reason we can approximate $\hat{N}_X$ in chapter 2.

5.4 Bayesian Network Parameter Learning From Randomized Data

The ML estimate of the parameter using the estimated sufficient statistics N_{ijk} is $\hat{\theta}_{ijk}^{ML} = \frac{\hat{N}_{ijk}}{\hat{N}_{ij}} = \frac{\hat{N}_{ijk}}{\sum_{k=1}^{\kappa_i}\hat{N}_{ijk}}$ and the MAP estimate of the parameter using the estimated sufficient statistics is $\hat{\theta}_{ijk} = \frac{\alpha_{ijk}+\hat{N}_{ijk}}{\alpha_i j+\hat{N}_{ij}}$. Here, we use the ML estimation and analyze its performance. Results for MAP estimation can be obtained in a similar fashion. We discuss the distribution of estimator $\hat{\theta}_{ijk}^{ML}$ in this section.

5.4.1 Ratio of Two Dependent Normal Random Variables

The estimated parameter is a ratio of two dependent (well approximated) normal random variables with non-zero mean. The exact distribution of the ratio of two normal variables $W = \frac{X_1}{X_2}$ with $(X_1, X_2) \sim N(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$ for arbitrary mean, variance, and correlation is well-known [Hin69]. However, the general distribution is quite complicated.

An approximation to the exact distribution, when the probability $P(X_2 > 0) \to 1$ or when $\frac{\mu_2}{\sigma_2}$ is large is also given in [Hin69]. The probability density function (p.d.f.) of $W = \frac{X_1}{X_2}$ can be well approximated by $f(w) = \frac{b(w)d(w)}{\sqrt{2\pi}\sigma_1\sigma_2 a^3(w)}$, where $a(w) = \sqrt{\frac{w^2}{\sigma_1^2} - \frac{2\rho w}{\sigma_1\sigma_2} + \frac{1}{\sigma_2^2}}$, $b(w) = \frac{\mu_1 w}{\sigma_1^2} - \frac{\rho(\mu_1+\mu_2 w)}{\sigma_1\sigma_2} + \frac{mu_2}{\sigma_2^2}$, and $d(w) = exp\{\frac{b^2(w)-ca^2(w)}{2(1-\rho^2)a^2(w)}\}$, when $\frac{\mu_2}{\sigma_2}$ is large. Furthermore, $Z = \frac{\mu_2 w - \mu_1}{\sigma_1\sigma_2 a(w)}$ is approximately a standard normal distribution if $\frac{\mu_2}{\sigma_2}$ is large. A Taylor series expansion of Z around $\frac{\mu_1}{\mu_2}$ shows that $Z \approx \frac{\mu_2}{\sigma_1\sigma_2\sqrt{\frac{\mu_1^2}{\sigma_1^2\mu_2^2} - \frac{2\rho\mu_1}{\sigma_1\sigma_2\mu_2} + \frac{1}{\sigma_2^2}}}(w - \frac{\mu_1}{\mu_2})$. It follows that the distribution of W can be approximated by a normal distribution with mean $\frac{\mu_1}{\mu_2}$ and variance $\frac{\sigma_1^2\sigma_2^2}{\mu_2^2}(\frac{\mu_1^2}{\sigma_1^2\mu_2^2} - \frac{2\rho\mu_1}{\sigma_1\sigma_2\mu_2} + \frac{1}{\sigma_2^2})$.

5.4.2 Distribution of the Estimator of Parameters

We have $\hat{\theta}_{ijk}^{ML} = \frac{\hat{N}_{ijk}}{\hat{N}_{ij}} = \frac{\hat{N}_{ijk}}{\sum_{k=1}^{K_i}\hat{N}_{ijk}}$. The estimator of parameters can be approximated by the ratio of two dependent jointly normal variables with non-zero means. The jointly normal variable $(\hat{N}_{ijk}, \hat{N}_{ij})$ has (approximate) distribution $N(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$, where $\mu_1 = N_{ijk}$, $\mu_2 = N_{ij}$ and $\sigma_1^2, \sigma_2^2, \rho$ can be obtained from Theorem 5.2 and 5.1. From Theorem 5.2 and 5.1, we can see $\frac{\mu_2}{\sigma_2}$ is of the order of $\sqrt{N_{ij}}$ for a given data set and a transition probability matrix P. So $\frac{\mu_2}{\sigma_2}$ is large even for relatively small sample sizes. Hence, the approximation of the distribution of ratio of two normal random variables with the simpler form in section 5.4.1 works well for us. On the other hand, the normal

approximation using Taylor expansion is also feasible in our case since $0 \le \frac{N_{ijk}}{N_{ij}} \le 1$. Hence, for a given Data set D and a probability transition matrix P, the p.d.f. of $\hat{\theta}_{ijk}^{ML} = \frac{\hat{N}_{ijk}}{\hat{N}_{ij}}$ can be approximated by a normal distribution with mean $\theta_{ijk} = \frac{N_{ijk}}{N_{ij}}$ and variance $\frac{\sigma_1^2\sigma_2^2}{N_{ijk}^2}(\frac{N_{ijk}^2}{\sigma_1^2 N_{ij}^2} - \frac{2\rho N_{ijk}}{\sigma_1\sigma_2 N_{ij}} + \frac{1}{\sigma_2^2}) = \frac{\sigma_1^2\sigma_2^2}{N_{ijk}^2}(\frac{\theta_{ijk}^2}{\sigma_1^2} - \frac{2\rho\theta_{ijk}}{\sigma_1\sigma_2} + \frac{1}{\sigma_2^2})$. We can see the variance is in the order of $\frac{1}{N}$ since σ^2 s are in the order of N from above theorems.

5.4.3 a Simple Example

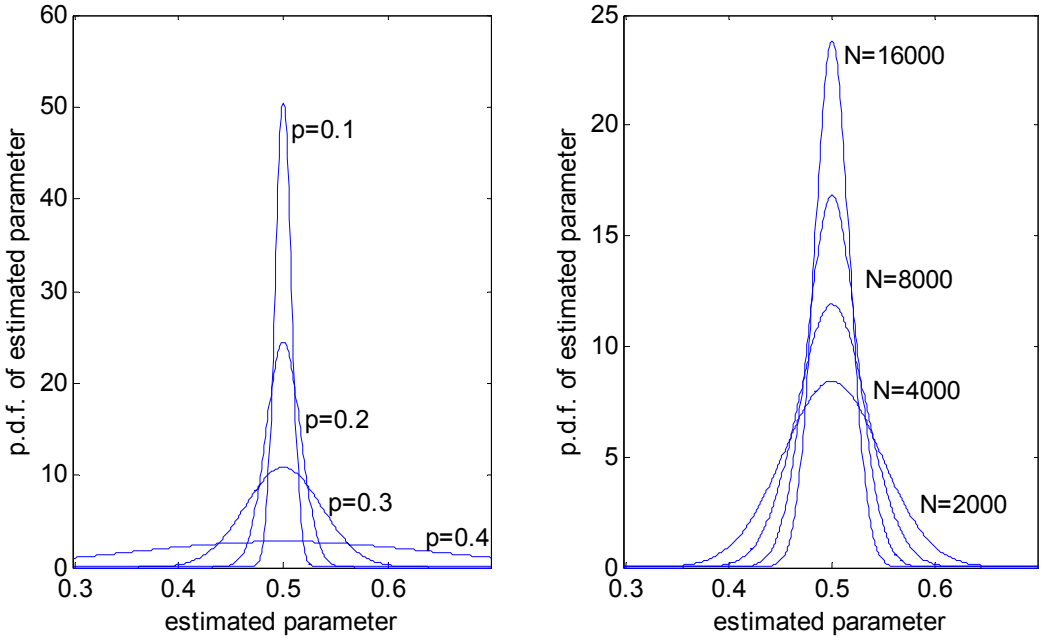
In this section, we give a simple example to quantitatively assess the estimation error using approximated normal distribution. Suppose Node X_2 is the only parent of X_1 and they are at different sites, where both X_2 and X_1 are binary valued. The values of X_2 and X_1 are randomized independently and uniformly by a binary symmetric randomization scheme with parameter p. Suppose $N_{111} = N_{112} = N_{121} = N_{122}$; that is the true parameters are all 0.5. Figure 5.1 (a) gives the approximate distribution of θ_{111} as a function of p, when $N = 8000$. From Figure 5.1(a) we can see that the variance of the estimator increases as p increases. Figure 5.1(b) gives the approximate distribution of θ_{111} for different N values, when $p = 0.25$. From Figure 5.1(b) we can see that the estimator variance decreases as the sample size increases.

5.5 Structure Learning from Randomized Data

In this section, we discuss structure learning from randomized data. Learning algorithms for structure learning from randomized data are introduced in section 5.5.1. An extra-link problem is discussed in section 5.5.2.

5.5.1 Structure Learning

The problem of learning Bayesian network structure from sample data D is to find a network structure G' that best matches the sample data D. Our problem here is to learn the network structure from the randomized data $\tilde{D}$. We use K-2 algorithm [CH92] for structure learning. K-2 is a greedy search algorithm that searches for a DAG G' that (approximately) maximizes a score function $Score(D, G)$.



(a) (b)

Figure 5.1 Approximate distributions of estimator of parameter (a) Different p with $N = 8000$ and (b) Different N with $p = 0.25$

The two score functions we discuss are Bayesian score and BIC/MDL score. The Bayesian score for a node is $Score(X_i, Pa_i(X_i)) = \prod_{j=1}^{J_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \prod_{k=1}^{K_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ij})}$, where α_{ijk} is from the assumption of prior Dirichlet distribution for θ_{ij} , that is $P(\theta_{ij} \mid G) = Dirichlet(\alpha_{ij1}, \cdots \alpha_{ijK_i})$. The BIC/MDL score for a node is given by $Score(X_i, Pa(X_i)) = \sum_{k=1}^{K_i} \sum_{j=1}^{J_i} N_{ijk} log(\theta_{ijk}) - \frac{N}{2} \#(X_i, Pa(X_i))$, where $\#(X_i, Pa(X_i))$ is the number of parameters we need to represent $p(X_i \mid Pa(X_i))$ .These score functions have the property of decomposability, that is $Score(D, G) = \sum_i Score(X_i, Pa_i(X_i))$, which makes a single operation in K-2 algorithm as the addition of a parent to a variable. The addition of a parent corresponds to two different candidate structures G_1 and G_2. By comparing $P_{old} = score(D, X_i, Pa(X_i))$ and $P_{new} = score(D, X_i, Pa(X_i) \cup \{Z\})$, where Z is a new candidate parent for variable X_i, K-2 algorithm decide if there is a link between candidate parent Z and node X_i . We use the two sets of estimated sufficient statistics $\hat{N}_{ijk}$ s to calculate P_{old} and P_{new}. The estimation of sufficient statistics is done as described in section 5.3 for each variable and the difference between P_{old} and P_{new} is caused only by the sufficient statistics associated with node X_i. The estimation of sufficient statistics is done as section 5.3 for both structures G_1 and G_2. The framework for structure learning was discussed in section 5.2.

The pseudo code for the K-2 algorithm is given below:

Problem: Find a DAG G' that approximately maximizes $Score\ (D, G)$.

Inputs: an ordered set of *m* nodes, an upper bound *u* on the number of parents for a node, and a sample data *D* containing *n* variables.

Output: parents Pa^i for each node X_i, where $0 \le i \le n$, in a DAG that approximately maximizes $Score(D, G)$.

For i=1; i<=n; i++{

$Pa_i = \{\}$;
$P_{old} = score(D, X_i, Pa_i)$;
$findmore = true$;

While ($findmore \&\& |Pa_i| < u$ {

// $\Pr ed(X_i)$ denotes those nodes who precede node X_i in the given order

Z=node in $\Pr ed(X_i) - Pa_i$ that maximizes $Score(D, X_i, Pa_i \cup \{Z\})$;

$P_{new} = Score(D, X_i, Pa_i \cup \{Z\})$;

if $P_{new} > P_{old}$

$P_{old} = P_{new}$;
$Pa_i = Pa_i \cup \{Z\}$;

else

$findmore = false$;}

}

5.5.2 Extra-link Problem in Structure Learning

One problem with structure learning is that estimation errors of the sufficient statistics tend to cause extra links, which are links that appear in the structure learned from randomized data $\tilde{D}$ but not in the structure learned from original data D. Missing links happen only when the randomization is relatively large. Missing links are those links that are learned from the original data D but are not learned from the randomized

data $\tilde{D}$. This extra links problem caused by estimation error can not be explained directly by analyzing the score function since the score function is a complicated nonlinear function. However, it is not difficult to understand from the sensitivity of independence to randomization as discussed in chapter 3, where we used odds ratio to explain this phenomenon. This kind of sensitivity to randomization holds for conditional independence which is encoded by the BN structure. It is also not difficult to understand from the statistical definition of independence. For example, if we have a sample data from two independent discrete random variables A and B, we can conclude statistically that A and B are independent if we have $p(A=i,B=j)=p(A=i)p(B=j)\pm\varepsilon\forall i,j$, where $p(A=i,B=j)$, $p(A=i)$ and $p(B=j)$ are estimated from their respective relative frequency in the sample set and ε depends on the specific independence testing method used. If the data samples of A and B are post-randomized, the relative frequencies can only be estimated using the available randomized samples. The estimation errors tend to cause $p(A=i,B=j)>p(A=i)p(B=j)+\varepsilon$ or $p(A=i,B=j)<p(A=i)p(B=j)-\varepsilon$ for some i and j. If the same independence testing is used, we tend to conclude that A and B are dependent. Similar argument holds for conditional independence which is encoded by the Bayesian network structure. Of course, if the relative frequency is estimated precisely, the extra links problem will not exist. Our experiments show that the extra links exist even for relatively small estimation errors. This kind of effect of estimation error on independence testing usually causes $Score(C,\{AB\})>Score(C,\{A\})$ although C is actually independent of B given A. Thus an extra link $B\rightarrow C$ will usually result. The above discussion suggests that if we want to learn correct structures from the randomized data, we should penalize complex structures. For Bayesian score,

we propose adding a parent only when $\eta P_{new} > P_{old}$, where η is a suitable threshold. For BIC/MDL score, we propose modifying the score function by increasing the penalty term (description length); that is, $Score_B(X_i, Pa(X_i)) = \sum_{k=1}^{K_i} \sum_{j=1}^{J_i} N_{ijk} log(\theta_{ijk}) - C^* \frac{N}{2} \#(X_i, Pa(X_i))$ for some $C^* > 1$.

Experiments show that the threshold η and C^* depend on the level of randomization. The more randomization, the larger threshold η and C^* should be. The underlying relationship between the threshold η (or C^*) and randomization under the available samples can be a further research topic. We believe there are optimal values for threshold η and C^* for a designed randomization scheme.

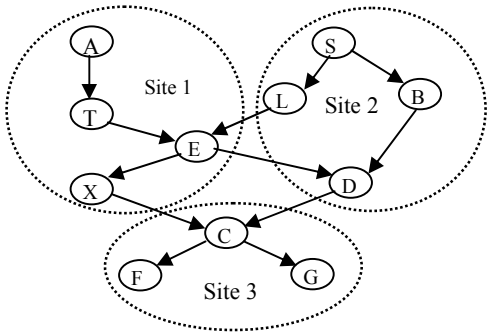
5.6 Experimental Results

In this section, we present some experimental results that demonstrate the accuracy of the Bayesian Network learning for different levels of randomization.

5.6.1 Experiment 1 Parameter learning: non-uniform randomization

In this experiment, we use the Bayesian Network shown in Figure 5.2, where the variables are distributed over three sites. All variables are binary except variables *L* and *B* which are ternary. The conditional probabilities of the different nodes are also shown. The conditional probability table of a Bayesian Network is shown in the way that conditional probabilities of the node given the combinations of its parents are in one row. For example, the conditional probabilities of node *L* are in the row denoted by *L* in figure

5.2. The conditional probabilities in one row are ordered in the way that the probabilities of one possible value of the node given combinations of its parents are followed by the probabilities of other values of the node given combinations of its parents. For example, the conditional probabilities of node L are ordered as the following: $p(L=1/S=1)=0.3$, $p(L=1/S=2)=0.7$, $p(L=2/S=1)=0.4$, $p(L=2/S=2)=0.15$, $p(L=3/S=1)=0.3$ and $p(L=3/S=2)=0.15$. 20,000 samples were generated from this Bayesian Network to form the dataset *D*. This data was then randomized according to the scheme described in Table 5.1, where variables *T*, *S*, and *G* were considered not sensitive and hence not randomized. Note that we use a non-uniform randomization with different levels of randomization for different variables. The corresponding at most γ amplification is also shown in Table 5.1. Table 5.2 shows the parameter of all nodes learnt from the randomized data using the algorithm



A	0.7 0.3	S	0.5 0.5
T	0.1 0.9 0.9 0.1	L	0.3 0.7 0.4 0.15 0.3 0.15
B	0.8 0.15 0.1 0.5 0.1 0.35		
E	0.25 0.8 0.15 0.5 0.3 0.4 0.75 0.2 0.85 0.5 0.7 0.6		
D	0.7 0.65 0.1 0.4 0.8 0.35 0.3 0.35 0.9 0.6 0.2 0.65		
X	0.2 0.6 0.8 0.4		
C	0.9 0.4 0.6 0.25 0.1 0.6 0.4 0.75		
F	0.25 0.9 0.75 0.1	G	0.2 0.4 0.8 0.6

Figure 5.2 A Bayesian Network for Experiment 1

A	Binary symmetric randomization $p_1 = p_2 = 0.25$, $\gamma = 3$
T	Not randomized
S	Not randomized
L	Ternary symmetric randomization $p_1 = p_2 = 0.15$ $\gamma = 4.67$
B	Ternary symmetric randomization $p_1 = p_2 = 0.15$ $\gamma = 4.67$
E	Binary symmetric randomization $p_1 = p_2 = 0.2$ $\gamma = 4$
D	Binary symmetric randomization $p_1 = p_2 = 0.25$ $\gamma = 3$
X	Binary symmetric randomization $p_1 = p_2 = 0.2$ $\gamma = 4$
C	Binary randomization $p_1 = 0.1$ $p_2 = 0.25$ $\gamma = 9$
F	Binary randomization $p_1 = 0.1$ $p_2 = 0.25$ $\gamma = 9$
G	Not randomized

Table 5.1 Randomization performed to the variables

A	0.70(0.70) 0.30(0.70)
T	0.10(0.50) 0.90(0.77) 0.90(0.50) 0.097(0.77)
S	0.50(0.00) 0.49(0.00)
L	0.30(0.49) 0.71(0.57) 0.39(0.64) 0.14(0.55) 0.31(0.93) 0.15(0.45)
B	0.80(0.77) 0.16(0.41) 0.094(0.71) 0.49(0.73) 0.10(0.11) 0.36(0.65)
E	0.25(0.20) 0.81(0.90) 0.14(2.7) 0.51 (1.2) 0.31 (2.6) 0.41(2.34) 0.75(2.0) 0.19(0.90) 0.86(2.7) 0.50(1.2) 0.69(2.64) 0.59(2.3)
D	0.69 (2.2) 0.65(1.3) 0.11(3.3) 0.38(0.77) 0.79(1.7) 0.39(5.65) 0.31(2.3) 0.35(1.3) 0.89(3.34) 0.62(0.77) 0.21(1.7) 0.61(5.7)
X	0.20(0.80) 0.60(1.1) 0.81(0.80) 0.40(1.1)
C	0.90(2.0) 0.38(1.6) 0.61(2.6) 0.25(2.1) 0.10(2.0) 0.62(1.6) 0.39(2.6) 0.75(2.1)
F	0.24(0.73) 0.91(1.1) 0.77(0.73) 0.092(1.1)
G	0.20 (0.30) 0.40(0.29) 0.80(0.30) 0.60(0.29)

Table 5.2 Mean and standard deviation ($\times 10^{-2}$) over 5 runs of parameters learned from the randomized data

described in section 5.2 for the case when there is a data miner(The results of the other case are always better than this case). All the values in the Table are average over 5 runs, with the corresponding standard deviation indicated in parenthesis. It is clear from the

Table that the proposed algorithms can accurately learn the BN parameters for both cases, even for moderate levels of randomization.

5.6.2 Experiment 2 Parameter learning: Uniform randomization

In this experiment, we use a uniform randomization and test the accuracy of the BN parameters as a function of the randomization parameter p. We used the Bayesian Network shown in Figure 5.3, where variables A, B, and C belong to Site 1 whereas variables D, E, and F belong to Site 2. All nodes are binary.

We consider the parameters of node D, which has parents in a different site. 5,000 samples were generated from the above Bayesian network. Binary symmetric randomization with parameter p was used to randomize the variables. Figure 5.4 shows a graph of the parameters as a function of p (mean of the estimated parameters over 10 runs is plotted; plot of mean plus one standard deviation is also included).

.

A	0.5 0.5
B	0.45 0.3 0.55 0.7
C	0.25 0.1 0.75 0.9
D	0.80 0.4 0.1 0.25 0.2 0.6 0.9 0.75
E	0.9 0.3 0.1 0.7
F	0.75 0.15 0.25 0.85

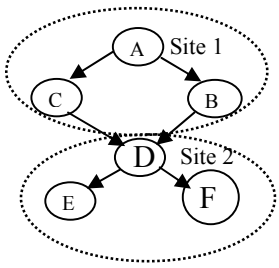


Figure 5.3 A Bayesian Network for experiment 2

It is clear from the figure 5.4 that for randomization parameter $p \le 0.25$, we can estimate the BN parameters with almost no error. Even for p values up to 0.3, we get reasonably good parameter estimates. From a privacy perspective, this corresponds to an amplification factor of $\gamma = \frac{1-0.25}{0.25} = 3$. We have to point out that this is done in 5000

samples. Under the same accuracy requirement, it is obvious that the p value can be

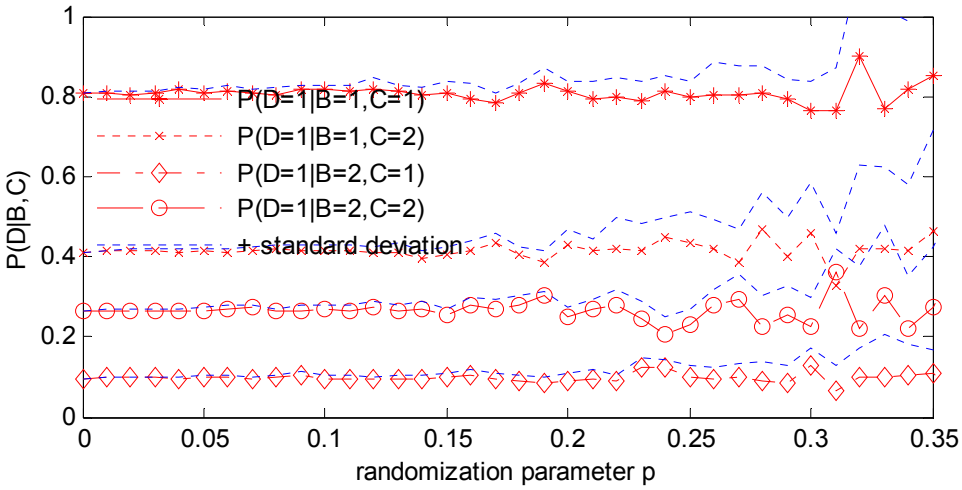


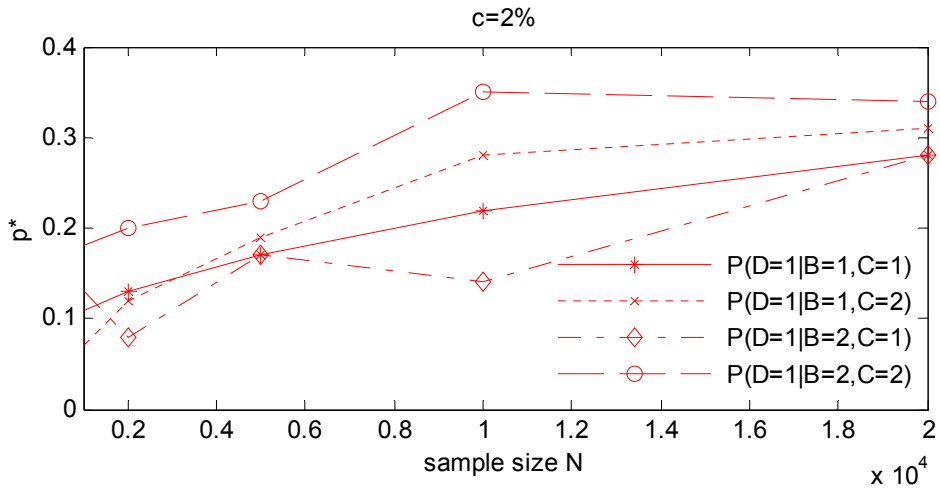
Figure 5.4 Estimated Parameters for Node D

closer to 0.5 if more samples are available. The closer p is to 0.5, the closer γ to 1. Another way to assess the performance of the algorithms is to determine the maximum level of randomization p that we can use for a given accuracy. Towards that end, for a given level of required estimation accuracy, defined in terms of an absolute error threshold c, let p^* be the smallest value of p (obtained by averaging over ten runs) for which the absolute value of the estimation error exceeds c. Figures 5.5 (a) and 5.5(b) show the variation of p^* as a function of the size N, for two different values of c = 2% and c = 4%.

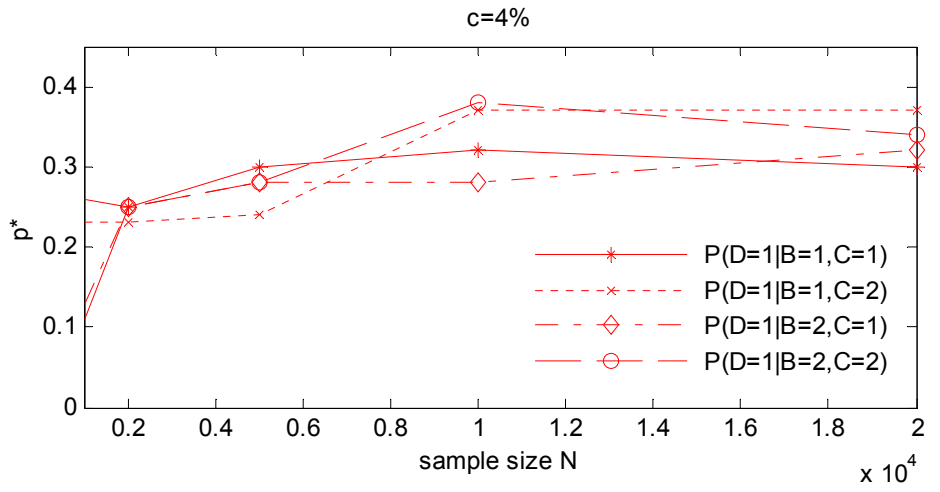
5.6.3 Experiment 3: Trade off between Privacy and Accuracy

In this experiment, we use the Bayesian network shown in figure 5.6, where variables are distributed over two sites. All Variables are binary. We generated 10,000 samples from this Bayesian Network. In order to see the trade off between privacy and

accuracy, we randomize the samples using binary symmetric randomization with



(a)



(b)

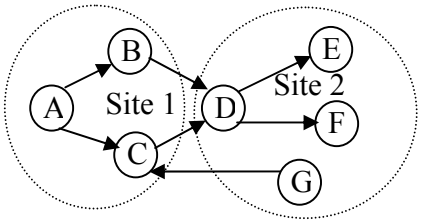
Figure 5.5 p^* versus sample size (a) $c = 2\%$ (b) $c = 4\%$ different levels of $p = p_1 = p_2$

and learn parameters from randomized samples using the method discussed in Section 5.4. As in previous experiments, we only present the results using the case when there is a data miner. Every variable is randomized using the symmetric binary randomization with the same randomization level p. Since parameters associated with a node is nothing but the conditional probability given its parents, the accuracy of parameters associated with a node can be measured by conditional Kullback-Leibler (CKL) distance between the

parameters learned from randomized data and those learnt from non-randomized data. The CKL distance for node i in our case is

$$D(X_i, p) = \sum_{j=1}^{J_i} P(pa_i = j) D_{KL}(P^{(0)}(X_i \mid pa_i = j), P^{(p)}(X_i \mid pa_i = j)),$$

where $P^{(0)}(X_i \mid pa_i = j)$ and $P^{(p)}(X_i \mid pa_i = j)$ are the parameters learnt from non-randomized data and those learnt from randomized data with randomization level p respectively and D_{KL} denotes the ordinary KL distance between two distributions. We present those distances associated with node *C*, node *D* and node *F* in figure 5.7. Those nodes are typical nodes for the given Bayesian network. The averages are over 10 independent runs. Average plus one standard deviation of 10 runs is also depicted (with dotted line). From the figure 5.7, we can clearly see the trade off between accuracy and privacy. Since we use the symmetric binary randomization, more privacy is preserved with bigger p when $p < 0.5$. With 10,000 training samples, the method still gets good accuracy when p=0.3.



A	0.5 0.5	G	0.3 0.7
B	0.8 0.2 0.2 0.8	E	0.8 0.2 0.2 0.8
C	0.5 0.9 0.7 0.35 0.5 0.1 0.3 0.65		
D	0.95 0.15 0.75 0.1 0.05 0.85 0.25 0.9		
F	0.8 0.2 0.2 0.8		

Figure 5.6: A Bayesian Network for experiment 3

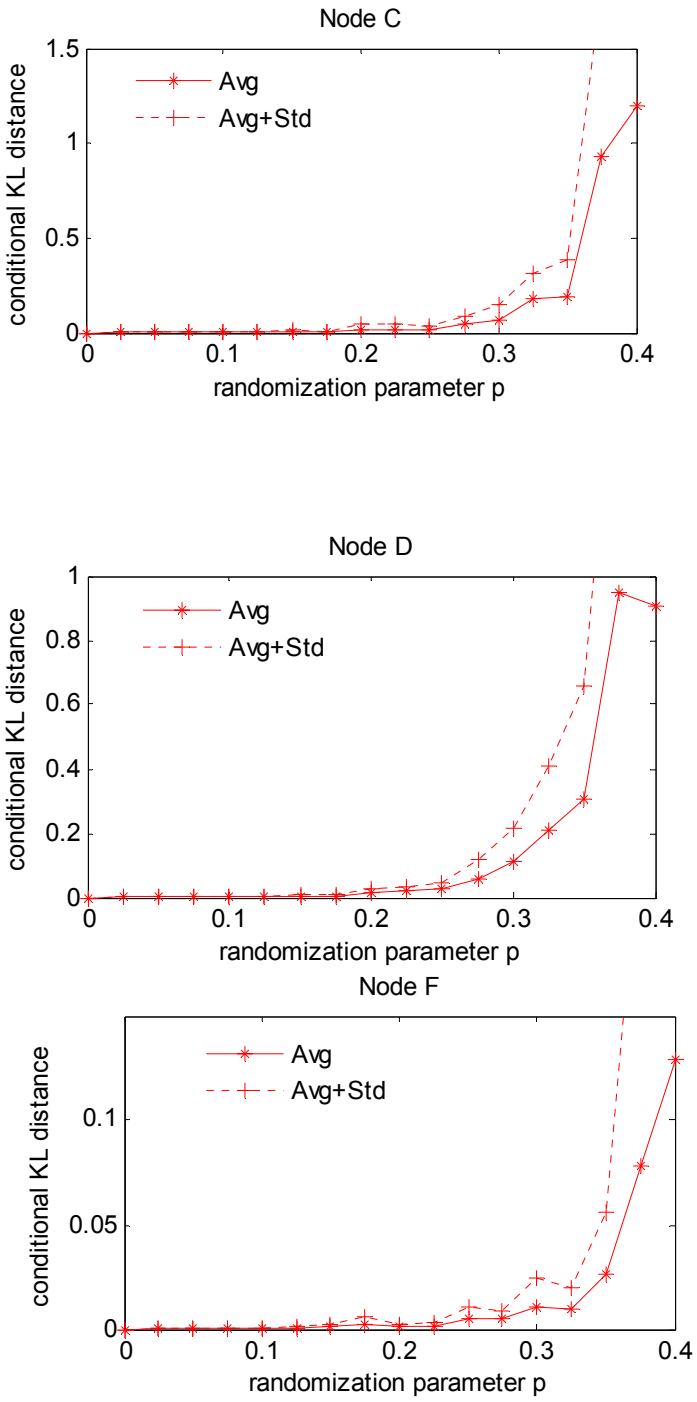
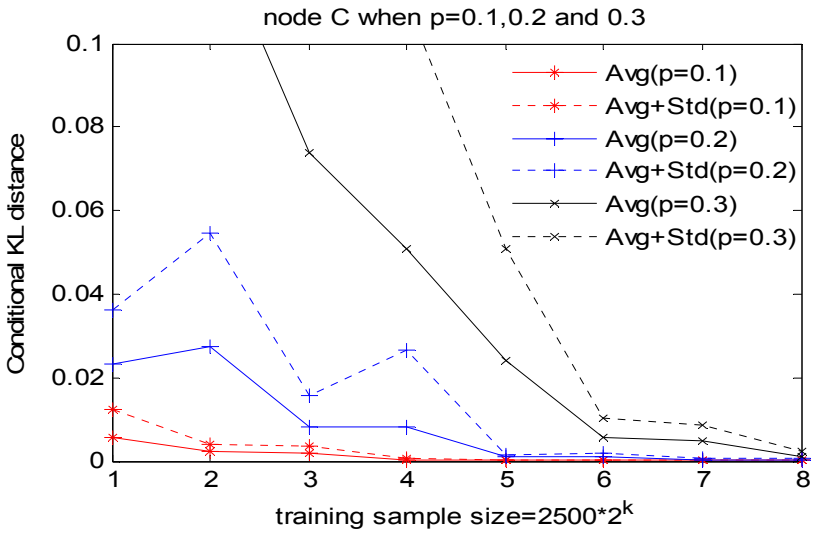


Fig. 5.7 CKL distance vs. randomization level p

As pointed out in Section 5.4, the variance of the estimator of parameter θ_{ijk} is of the order of one over the sample size N. Thus, under the same accuracy requirement,

more privacy can be preserved if there are more training samples. This experiment is performed to illustrate the effect of training sample size. We generated 2500×2^8 training samples using Bayesian network in figure 5.6. The proposed method in Section 5.4 is used to learn the Bayesian parameters from randomized data with randomization levels $p=0.1$, $p=0.2$, $p=0.3$, and $p=0.4$ with training sample size 2500×2^k $(k=1\cdots 8)$ respectively. The experiment results are shown in figure 5.8. The average is over 10 independent runs and the average plus one standard deviation is also shown. The experiment results for randomization level $p=0.4$ are shown separately. Those conditional distances out of the scale of vertical axis are not shown in the figures. From this experiment, we can clearly see that training sample size plays a key role in the trade off between accuracy and privacy. We can see that when the training sample size is very large, we can have both good privacy and good accuracy.



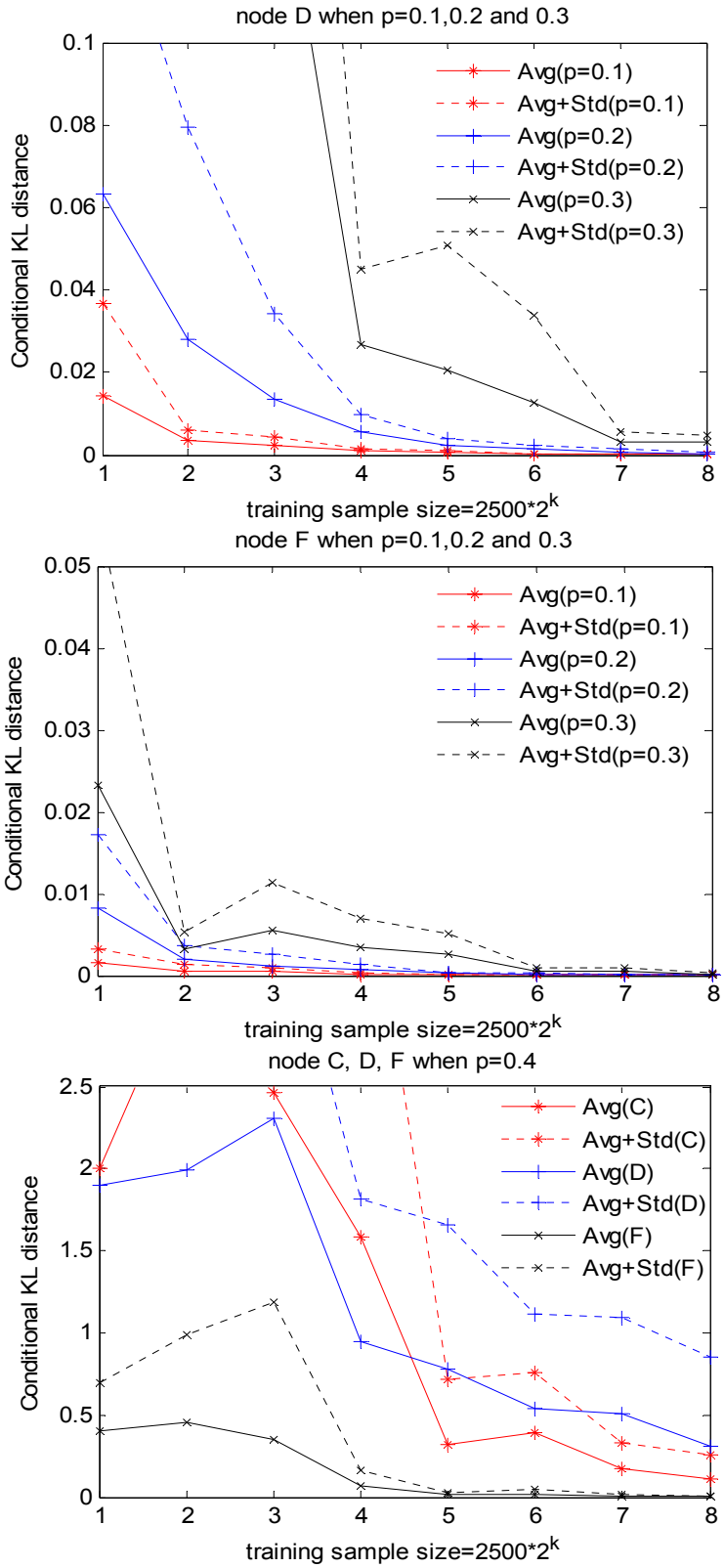
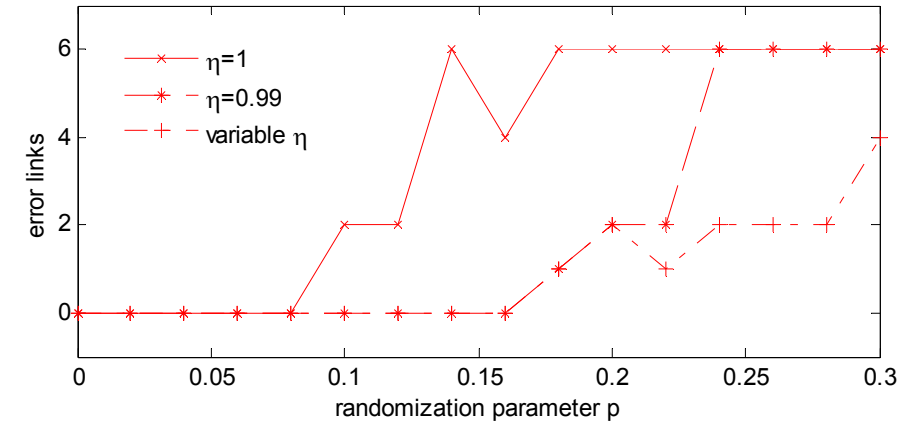


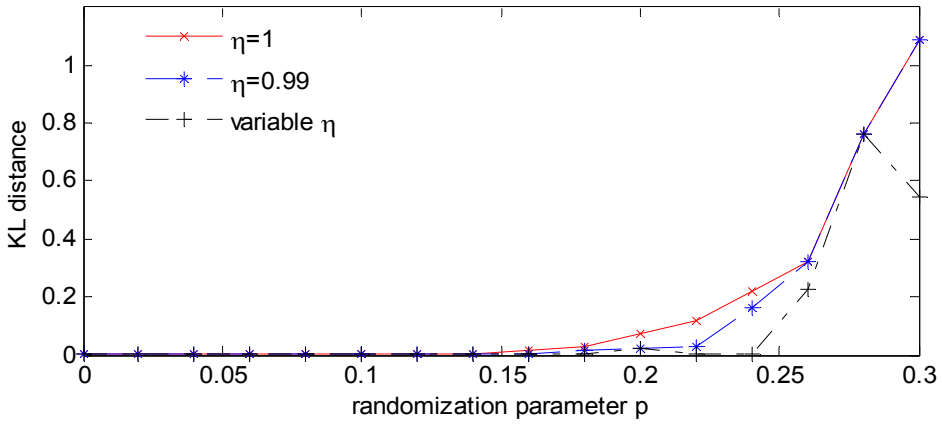
Figure. 5.8: CKL distance vs. training sample size

5.6.4 Experiment 4 Structure learning

In this experiment, we test the accuracy of BN structure learning from randomized data. 10,000 samples from the BN in figure 5.3 were used. The data was randomized using a binary symmetric randomization with parameter p. The K-2 algorithm with threshold η for Bayesian score or penalty term c^* for BIC/MDL score



(a)

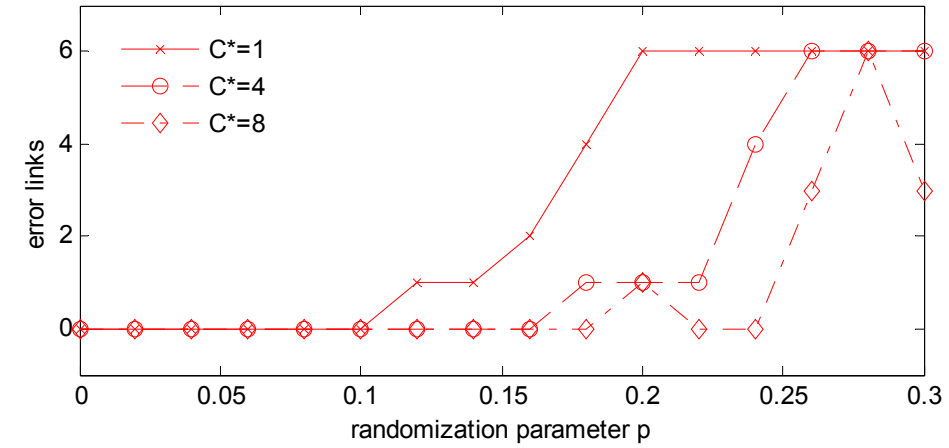


(b)

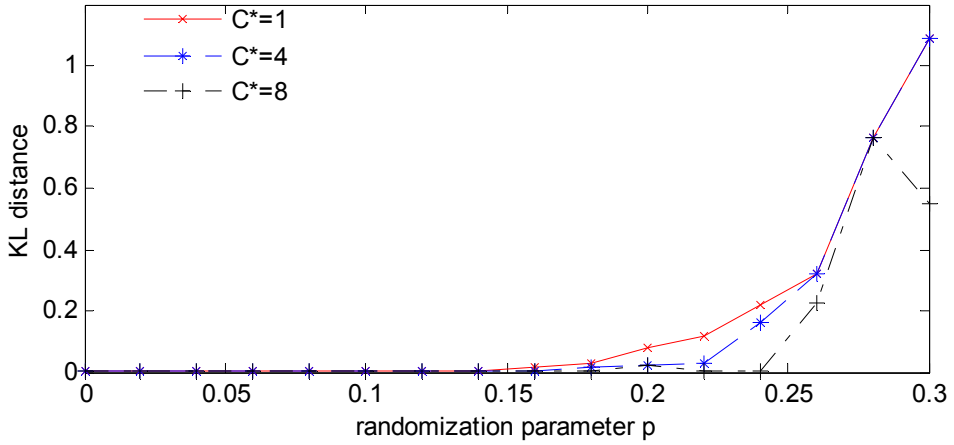
Figure 5.9: Structure Learning using Bayesian Score

was used to learn the BN structure from the randomized data. Node ordering $\{A, B, C, D, E, F\}$ was assumed and the maximum number of parents was set to 3. We quantify the error in structure learning with two different measures: (a) Sum of missing links and extra

links and (b) KL-distance between the joint probability of the learnt BN and the true BN.



(a)



(b)

Figure 5.10: Structure learning using BIC/MDL Score

The latter actually incorporates errors in the structure as well as the parameters and might be better. Figure 5.9 (a) shows the number of links in error (sum of missing links and extra links) as a function of the randomization parameter *p*, for the case of Bayesian scores. Figure 5.9 (b) depicts a similar graph for KL-distance. Three different choices of the threshold η were considered: η = 1, η = 0.99 and a variable η value depending on the

randomization parameter value p as $\eta = \begin{cases} 0.99 & 0 < p \le 0.15 \\ 0.98 & 0.15 < p \le 0.25 \\ 0.97 & 0.25 < p \le 0.3. \end{cases}$ Figure 5.10 (a), (b) are similar results using BIC/MDL score. Again three different values for the penalty term C^* were considered: C^* = 1, C^* = 4, and C^* = 8. We would like to add that the structure error was always contributed by extra links, with just one exception (when Bayesian score with variable η is used) where we had one missing link.

CHAPTER SIX

PRIVACY-PRESERVING LINEAR CLASSIFIER LEARNING FROM RANDOMIZED DATA

In this chapter, we will introduce learning linear classifier from randomized data for PPDM. Linear classifier learning is based on cost minimization. The general framework for learning models based on cost optimization was discussed in chapter 4.

A linear classifier uses a decision hyperplane in an *m*-dimensional feature space to classify samples. The hyperplane can be represented as $g(x) = w^T x + w_0 = 0$, where $w = [w_1, w_2, \ldots, w_n]$ is known as the weight vector, w_0 as the threshold and $x = [x_1, \cdots x_n]$ is the feature vector. Classes of instances are decided based on the sign of $g(x)$. The primary problem in the design of a linear classifier is finding the weights $w_i, i = 0, \ldots, n$, defining the decision hyperplane. Conceptually, multiple class problem can be convert to a series of binary classification problems. Several algorithms have been proposed to learn a linear classifier. The cost function usually used by those algorithms are perceptron cost functions. We will discuss linear classifier learning from randomized data using perceptron cost in section 6.2 . Other cost functions used in learning linear clsssifier can similarly be adjusted for learning from randomized data.

6.1 Randomization

Let *D* be a data set represented by the following $N \times (n+1)$ matrix:

$$D = \begin{bmatrix} x_{11} & \cdots & x_{1n} & c_1 \\ \vdots & \cdots & \vdots & \vdots \\ x_{N1} & \cdots & x_{Nn} & c_N \end{bmatrix} = \begin{bmatrix} x_1 & c_1 \\ \vdots & \vdots \\ x_N & c_N \end{bmatrix}$$

Each column of the data set represents an attribute, so there are $n+1$ variables

$X_1 \cdots, X_j \cdots X_n$ and C. $c_1 \cdots c_N$ are class labels of corresponding instances. The above data set D is assumed heterogeneosly distributed among several parties. Those parties who own a vertical partition of the data set D are interested in collaboration and they want to benefit from a joint classifier based on their joint data. For example, they might want to derive a prediction model based on the joint data set. However, they are not willing to fully share their sensitive data with other parties. Each party first randomizes its data and sends the randomized data to the data miner together with the probablity transition matrix used in randomization. The data used for linear classifier are often numerical. Before randomization, the numerical variables are discretized using techniques such as MGAS we introduced in chapter 2. We assume that all variables are numerical except the class variable. After discretization, each data entry is replaced by the mean value of the interval it belongs to. Let m_{ij} be the mean value of the interval to which x_{ij} belongs. Let $S_j = \{s_{j1}, \cdots, s_{jk}, \cdots, s_{j|S_j|}\}$ be a set of means of all intervals of attribute X_j where $1 \le k \le |S_j|$. For any $1 \le i \le N$ and $1 \le j \le n$, there alwasys exists some k such that $m_{ij} = s_{jk}$ and $1 \le k \le |S_j|$. After discretization, the above Data set D is represented as

$$D_d = \begin{bmatrix} m_{11} & \cdots & m_{1n} & c_1 \\ \vdots & \cdots & \vdots & \vdots \\ m_{N1} & \cdots & m_{Nn} & c_N \end{bmatrix} = \begin{bmatrix} m_1 & c_1 \\ \vdots & \vdots \\ m_N & c_N \end{bmatrix}.$$

The jth column of D_d represents discretized variable M_j corresponding to X_j. In order to preserve privacy of each individual value, data owners can randomize their part of data by using PRAM schemes we discussed in chapter 2. The randomized variables

corresponding to $M_1 \cdots M_j \cdots M_n$ are denoted by $\tilde{M}_1 \cdots \tilde{M}_j \cdots \tilde{M}_n$ respectively. For example, we can apply multi- category randomization to M_j, which takes values from set S_j. The randomization scheme used for M_j can be characterized by a probablity transition matrix $P^j = \{p^j_{k_1 k_2}\}$, where $p^j_{k_1 k_2} = p(\tilde{M}_j = s_{jk_2} \mid M_j = s_{jk_1})$ and $1 \le k_1, k_2 \le |S_j|$. The randomization can be implemented simultaneously to several variables. Without loss of generality, we discuss the case when all variables $M_1 \cdots M_j \cdots M_n$ are randomized independently . We can also group values in set S_j into several groups and values in one group can only be randomized to the values in the same group. Simultaneous randomization and grouping will not affect our discussion of learning linear classifier from randomized data since they only affect the specific probablity transition matrix. The class variable can also be randomized using PRAM characterized by a transition matrix $P_C = \{p_{c_1,c_2}\}$ where $p_{c_1,c_2} = p(\tilde{C} = c_2 \mid C = c_1)$ and $1 \le c_1, c_2 \le |S_c|$ where S_c is the set of all possible classes. After PRAM, we obtain a randomized version of D_d, which is denoted by $\tilde{D}_d = \begin{bmatrix} \tilde{m}_{11} & \cdots & \tilde{m}_{1n} & \tilde{c}_1 \\ \vdots & \cdots & \vdots & \vdots \\ \tilde{m}_{N1} & \cdots & \tilde{m}_{Nn} & \tilde{c}_N \end{bmatrix}$, $\tilde{m}_{ij}$ and $\tilde{c}_i$ ($1 \le i \le N$ and $1 \le j \le n$) denote the randomized value corresponding to m_{ij} and c_i respectively. The objective of the data miner is to find the unknown parameters w_i, $i = 0, \ldots, n$ using this randomized data set $\tilde{D}_d$ and the probablity transition matrices $P_1 \cdots P_n$ and P_C. The usual way to learn linear classifier is to introduce a cost function and then the weight vector is obtained by optimizing the cost function. In the following section, we discuss learning the weight vector from the randomized data using perceptron cost function.

6.2Learning Linear Classifier from Randomized Data

Since multi-class problem can be decomposed to a set of binary classification problem, without loss of generality, we discuss binary linear classification problems in this section. Let the two classes be denoted by $S_c = \{c_1, c_2\}$.

The perceptron cost is $J(w) = \sum_{i=1}^{N} \delta_i(w) w^T m_i$ where $m_i = \{m_{i1}, m_{i2} \cdots m_{in}\}$ and $\delta_i(w) = 0$ if ith record is correctly classified, $\delta_i(w) = -1$ if $c_i = c_1$ and misclassified (i.e. $w^T m_i < 0$), $\delta_i(w) = 1$ if $c_i = c_2$ and misclassified (ie. $w^T m_i > 0$). Perceptron learning algorithm is usually used for separable classes and Pocket algorithm is usually used in nonseparable classes where contradictory class labels of samples may exist. Pocket algorithm is a variant of perceptron algorithm suggested by [Gal90]. Using the framework proposed in chapter 4, we can minimize the following cost function instead of the original cost function to learn the parameters from the randomized data:

$$J(w) = \sum_{i=1}^{N} E(\delta_M(w) w^T M \mid \tilde{m}_i) = \sum_{i=1}^{N} E(\delta_M(w) w^T M \mid \tilde{m}_{i1} = s_{1k_1}, \tilde{m}_{i2} = s_{2k_2} \cdots \tilde{m}_{in} = s_{nk_n}, \tilde{c}_i = c_{k_{n+1}}),$$

where $M = \{M_1, M_2, \cdots M_n\}$, $\delta_M(w)$ denotes a variable that depends on M and C (when $M = m$ and $C = c$, δ_M takes the value according to the definition we give to δ_i depending on m_i and c_i at the beginning of this section) and $s_{1k_1} \in S_1 \cdots s_{nk_n} \in S_n$ and $c_{k_{n+1}} \in S_C$. Furthermore, we have

$$\sum_{i=1}^{N} E(\delta_M(w) w^T M \mid \tilde{m}_i) = \sum_{i=1}^{N} E(\delta_M(w) w^T M \mid \tilde{m}_{i1} = s_{1k_1}, \tilde{m}_{i2} = s_{2k_2} \cdots \tilde{m}_{in} = s_{nk_n}, \tilde{c}_i = c_{k_{n+1}})$$

$$= \sum_{i=1}^{N} \sum_{l_1=1}^{|S_1|} \cdots \sum_{l_n=1}^{|S_n|} \sum_{l_{n+1}=1}^{|S_C|} \delta_{sc_{l_{n+1}}}(w) w^T sp(M = s, C = c_{l_{n+1}} \mid \tilde{m}_{i1} = s_{1k_1}, \tilde{m}_{i2} = s_{2k_2} \cdots \tilde{m}_{in} = s_{nk_n}, \tilde{c}_i = c_{k_{n+1}})$$

where $s = \{s_{1l_1}, s_{2l_2}, \cdots s_{nl_n}\}$, $s_{1l_1} \in S_1 \cdots s_{nl_n} \in S_n$ and $\delta_{sc_{l_{n+1}}}(w) = \delta_{MC}(w)$ when $M = s$ and $C = c_{l_{n+1}}$. By ordering the values of $V = \{M_1, M_2 \cdots M_n, C\}$ in such a way that M_1 changes the slowest and C changes the fastest, we can denote $V = v_l$ if $\{s_{1l_1}, s_{2l_2}, \cdots s_{nl_n}, c_{l_{n+1}}\}$ is the *lth* value of V for $1 \le l \le L = |C| \prod_{i=1}^{n} |S_i|$ and similar notations apply to $\tilde{V} = \{\tilde{M}_1, \tilde{M}_2 \cdots \tilde{M}_n, \tilde{C}\}$. We denote by $\delta_l(w)$ the corresponding $\delta_{MC}(w)$ associated with v_l. The cost function then becomes

$$J(w) = \sum_{i=1}^{N} \sum_{l=1}^{L} \delta_l(w) w^T v_l p(V = v_l \mid \tilde{V} = v_i).$$

Let T_l, $\tilde{T}_l$, $\hat{T}_l$ respectively be the frequency counts of samples such that $\{m_i, c_i\} = v_l$ in discretized data set D_d, frequency counts of samples such that $\{\tilde{m}_i, \tilde{c}_i\} = v_l$ in the randomized data set $\tilde{D}_d$ and the estimate of T_l. Let $T, \tilde{T}$ and $\hat{T}$ respectively be the L dimensional vectors of T_l, $\tilde{T}_l$ and $\hat{T}_l$. We can use the moment estimator or maximum likelihood estimator to get $\hat{T}_l$. For moment estimation, we have the unbiased estimator $\hat{T} = P^{-1}\tilde{T}$, where $P = P^1 \otimes P^2 \otimes \cdots P^n \otimes P^C$.

For the pocket algorithm, it is convenient to define a new data set with NL samples as follows:

$$\begin{matrix} v_1^i & \delta_1^i(w) \\ \vdots & \vdots \\ v_L^i & \delta_L^i(w) \end{matrix}$$

for any $1 \le i \le N$ and $\delta_l^i = \delta_l(w) p(V = v_l \mid \tilde{V} = v_i)$.

We can directly use the pocket algorithm with the above defined NL data samples

v_l^i and δ_l^i for $1 \le i \le N$ and $1 \le l \le L$. The basic idea of the pocket algorithm is to run perceptron learning while keeping an extra set of weights "in your pocket". Whenever the perceptron weights have a longest run of consecutive correct classification of randomly selected training examples, these perceptron weights replace the pocket weights. Finally, the pocket weights are the outputs of the algorithm, i.e., the learned weight vector. The differences between implementation of the algorithm using randomized and original (nonrandomized) data lies in calculating run_w , number of consecutive correct classifications, and num_ok_W, the total number of training example that w correctly classifies. When v_l^i is correctly classified, run_w increases by δ_l^i instead of increasing by 1 during implementation of the pocket algorithm. Similarly, when v_l^i is correctly classified while checking all training samples, num_ok_w increases by δ_l^i instead of increasing by 1 .

6.3 Experimental Results

We used our linear classifier learning algorithm on a standard dataset from the UCI repository[UCI]. Since learning from randomized data usually requires more samples than regular learning, we generated a bigger dataset (by resampling) for our simulation based on the data set from UCI data repository. The two data sets we used are the Lenses dataset and Iris Plant dataset.

Two simulations using data sets generated based on Lenses database and Iris Plant database are described below. The results are shown respectively in Figures 6.1 and 6.2.

(1) We generated independently 6,000 samples of four categorical variables $\{X_1, X_2, X_3, X_4\}$ independently with the PMF shown in Table 6.1. For each generated data sample $\{X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = x_4\}$, the class label are assigned according to the class label assigned to the four variables in the Lens database. Class 1 and class 2 in lens database are combined into a single class, with class 3 being a separate class. These yield the two classes in our (binary) linear classification problem.

Variables	1	2	3
X_1	0.5	0.3	0.2
X_2	0.9	0.1	0
X_3	0.2	0.8	0
X_4	0.2	0.8	0

Table 6.1 Probabilities of Variables $X = \{X_1, X_2, X_3, X_4\}$

Since the variables are categorical, discretization is not needed. The randomization applied to the four variables are as shown in Table 6.2.

Variables	Randomization Schemes Used
X_1	Ternary symmetric randomization with parameter p
X_2	Binary symmetric randomization with parameter p
X_3	Binary symmetric randomization with parameter p
X_4	Binary symmetric randomization with parameter p

Table 6.2 Randomization to the variables

The proposed algorithm (modified pocket algorithm described in section 6.2) was used to learn the weight vector. The original (non randomized data) was used to test the classification accuracy. Different values of randomization parameter p (from 0 to 0.5) were used in the simulation. Each value of randomization parameter is simulated 5 times. The simulation results are shown in Figure 6.1. The average classification accuracy of 5

simulation runs and averge minus one standard deviation are shown. For the purpose of comparison, the simulation results for classification accuracy using weight vector learned directly from the randomized data by using pocket algorithm are also shown in Figure 6.1.

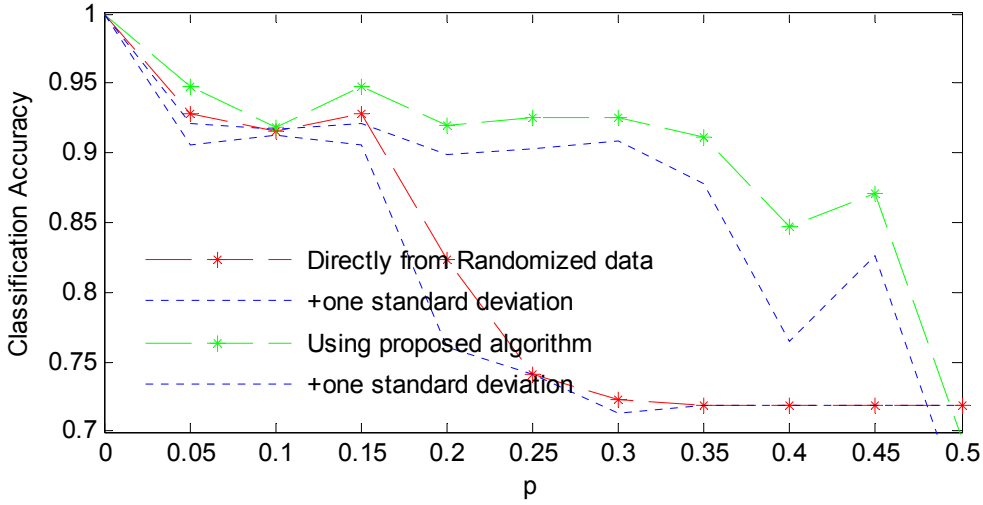


Figure 6.1 Classification Accuracy vs. randomization parameter p for data set generated from the Lens data set

(2) Iris plant dataset has 4 numerical variables $\{X_1 \cdots X_4\}$ and a class variable C. We generate a data set D by repeating Iris plant database 100 times. A random noise with uniform distribution in $[-0.2 \quad 0.2]$ was added to each variable (except the class label) in D. MGAS was applied to numerical values in D for discretization and PRAM was applied to each discretized numerical variable. The PRAM scheme used was M-category randomization with parameter p, i.e., each category of the variable is randomized to all other categories with probability $\frac{p}{|X_i|-1}$, where $|X_i|$ is the number of categories resulting from MGAS discretization. The reconstructed distribution from the discretized variables and the original histogram are shown in Figure 6.3. Class 1 and class 2 are combined into a single class and class 3 is another separate class. The proposed algorithm

(modified pocket algorithm in section 6.2) was used to learn the weight vector. The original (non randomized data) was used to test the classification accuracy. Different values of randomization parameter p (from 0.1 to 0.6) were used in the simulation. Each value of randomization parameter is simulated 5 times. The simulation results are shown in Figure 6.2. The average classification accuracy of 5 simulation runs and average minus one standard deviation are shown. For the purpose of comparison, the simulation results for classification accuracy using weights learned directly from the randomized data using pocket algorithm are also shown in Figure 6.2.

From the above two simulation, we can see that the classification accuracies of the weight vector learned from randomized data are reasonable. Especially for the second simulation, the classification accuracy is almost as good as the weighting vector learned from the original data even with large randomization.

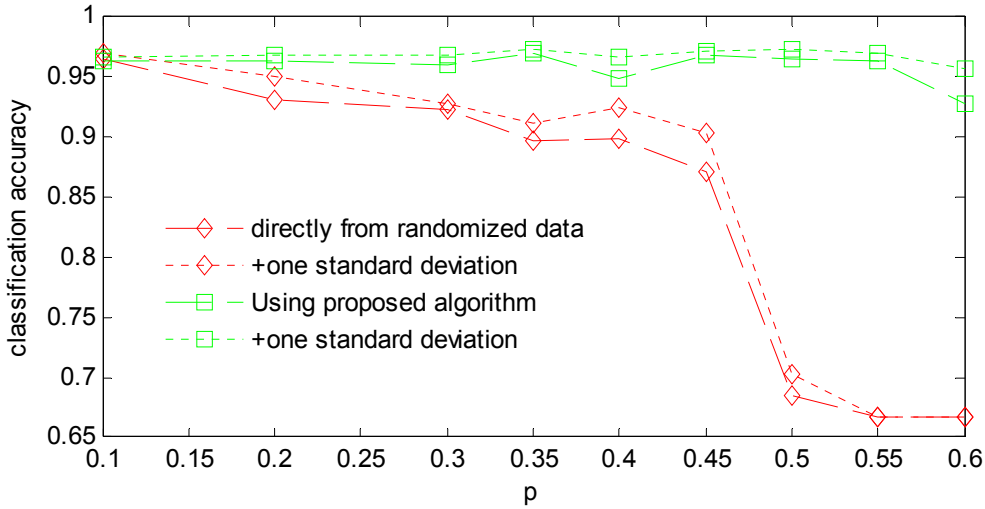


Figure 6.2 Classification Accuracy vs. randomization parameter p for data set generated from Iris data set

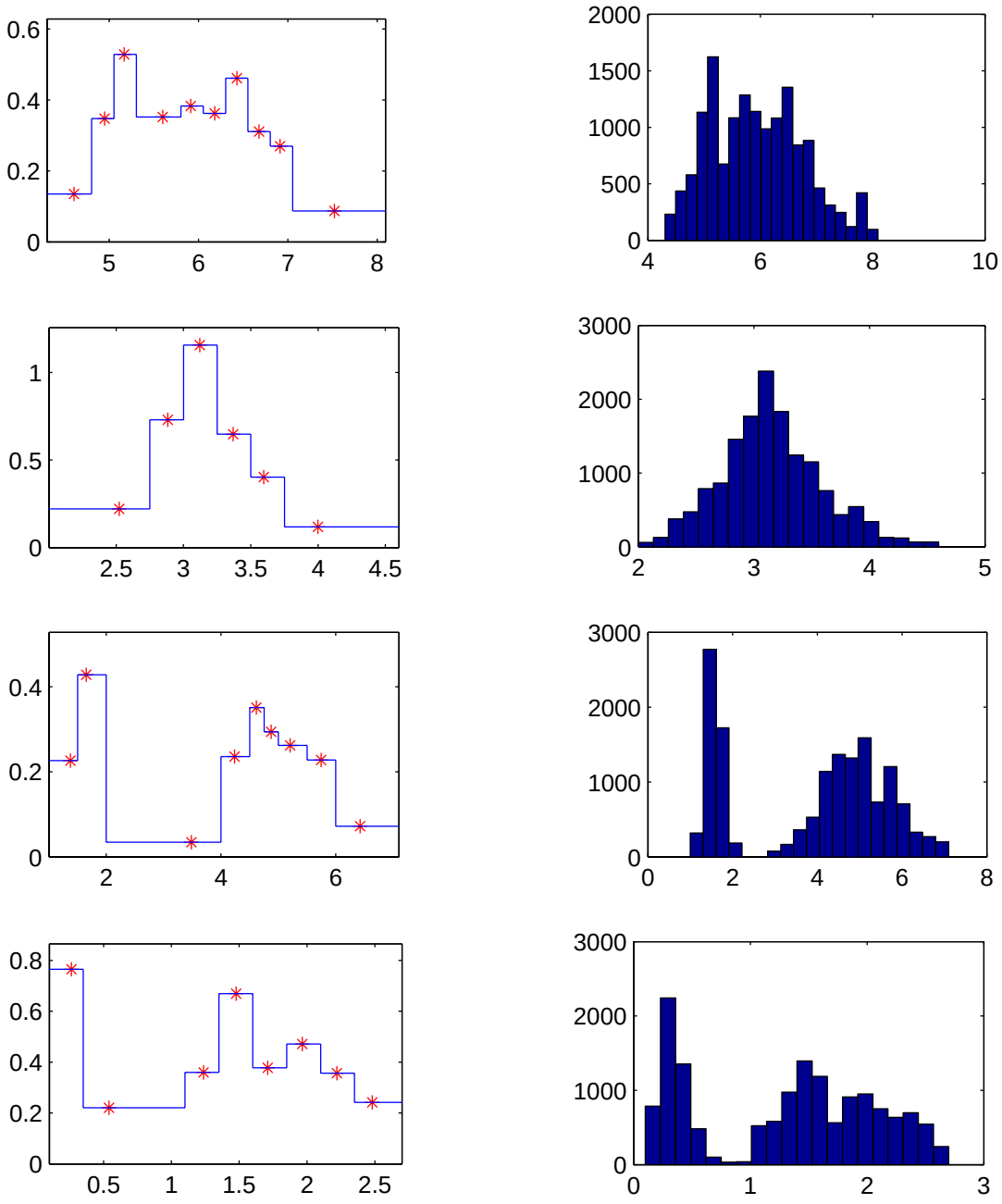


Figure 6.3 Variables after MGAS (Left) Reconstructed Distribution (Right) Histogram of Original Data

CHAPTER SEVEN

PRIVACY-PRESERVING ASSOCIATION RULE MINING AND DECISION TREE INDUCTION

In this chapter, privacy-preserving association rule mining and decision tree induction using PRAM are discussed. Association rule and decision tree are both models based on summaries of data sets. The framework for learning these kinds of models was discussed in chapter 4. Privacy-preserving association rule mining using PRAM is discussed in section 7.1 whereas privacy-preserving decision tree induction using PRAM is discussed in section 7.2.

7.1 Association Rule Mining

The core computation in association rule mining is to find frequent itemsets, that is, a set of items whose support in a data set is larger than a specified threshold sup_{min}. Frequency counts estimation methods in chapter 2 can be directly used to estimate the support of itemsets from post randomized data set. In Mask algorithm proposed by [RH02], the randomization is done independently to each item of each record by using a simple binary randomization. PRAM provides a more flexible framework for mining association rule in a privacy-preserving fashion. Data owners have more choices in randomizing their data and can choose a randomization scheme according to their specific privacy concerns. Although mask algorithm is convenient to implement, however, it usually results in large simultaneous $\gamma - amplification$ for a group of items. Instead of applying binary randomization to all items in a group independently, a group

of items can be randomized simultaneously in our framework. Simultaneous randomization will reduce $\gamma - amplification$ for a group of items and help limit more privacy breaches. In section 7.1.1, Apriori algorithm for association rule mining and its efficient implementation are briefly introduced. Association rule mining from randomized data is presented in section 7.1.2, where we deal with estimation of support of candidate frequent itemsets and corresponding implementation of Apriori algorithm, which have to be adapted to learning from the randomized data. The simultaneous $\gamma -$ amplification for a group of items by simultaneous randomization is compared with $\gamma -$ amplification for a group of items by applying binary randomization independently to each item of the group in section 7.1.3. Experimental results are provided in section 7.1.4.

7.1.1 Association Rule Mining and its Implementation

Association rules, proposed by Agarwal et al. [AIS93], are a class of simple but powerful regularities in binary data. An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are sets of items. The intuitive meaning of such a rule is that in the rows of the database where the attributes in X have value true, attribute Y also has value true (with high probability) [AS94]. There may be hundreds of association rules in a given data set depending on its size and complexity. The process of mining for such rules is called association rule mining.

There are primarily two measures of quality for each rule, support and confidence. The rule $X \Rightarrow Y$ has support s% in the transaction set D if s% of transactions in D contains both X and Y. The rule has confidence a% if a% of transactions in D that contain

X also contain Y. The goal of association rule mining is to find all the rules with support and confidence exceeding some user specified thresholds.

The most common approach to finding association rules is to break up the problem into two parts:

a. Find frequent itemsets,

b. Generate rules from frequent itemsets.

A frequent item set is an itemset whose number of occurrences is above a threshold sup_{min}. We use the notation L to indicate the complete set of frequent itemsets and $l \in L$ to indicate a specific frequent itemset. Finding frequent itemsets generally is conceptually quite easy but computationally very costly. The naïve approach would be to count all itemsets that appear in any transaction t. Given a set of itemsets of size k, there are 2^k subsets. Because of the explosive growth of this number, the challenge of solving the association rule mining problem is often viewed as how to efficiently determine all frequent itemsets. Most association rule mining algorithms are based on smart ways to reduce the number of itemsets to be counted. These potentially frequent itemsets c are called candidates, and the set of all counted (potentially frequent) itemsets are called candidate set C. One performance measure used for association rule algorithms is the size of C. Another problem to be solved by association rule algorithms is what data structure to use during the counting process. A trie or hash tree is commonly used. When all frequent itemsets are found, generating the association rule is straightforward. In the sequel, we only discuss finding the frequent itemsets.

Apriori algorithm is the most well known association rule algorithm and is used in most commercial products. The basic idea of the Apriori algorithm is to generate

candidate itemsets of a particular size and then scan the database to count these to see if they are frequent. During scan k *of* the data set, the supports of all $c \in C_k$ are counted. Only those candidates that are frequent are used to generate candidates for next pass. That is L_k are used to generate C_{k+1}. An itemset is considered a candidate only if all its subsets are also frequent. To generate candidates of size $k+1$, joins are made of frequent itemsets found in the previous pass. An algorithm called Priori-Gen is used to generate the candidate itemsets for each pass after the first. All singleton itemsets are used as candidates in the first pass. Here the set of large itemsets of the previous pass L_{k-1} is joined with itself to determine the candidates. Individual itemsets must have all but one item in common in order to be combined. The Apriori-Gen and Apriori algorithm itself are show below. Sup_c is used to denote the count for candidate itemset $c \in C$.

Apriori-Gen Algorithm:

Input : L_{k-1} // frequent itemsets of size k-1
Output: C_k //Candidates of size k
Apriori-gen algorithm
    $C_k = \phi$
for each $I \in L_{k-1}$ do
    for each $J \neq I \in L_{k-1}$ do
        if $k-2$ of the elements in I and J are equal then
            $C_k = C_k \cup \{I \cup J\}$;

Apriori Algorithm

Input: $I, D, \sup_{\min}$ // Itemsets, Database of transactions and minimum support respectively
Output: L //frequent Itemsets
Apriori algorithm
    $i = 0$; //i is used as a scan number
    $L = \phi$;
    $C_1 = I$; //initial candidates are set to be the items.
    repeat
        $i = i + 1$;

$L_i = \phi$;
for each $I_k \in C_i$ do
$Sup_{I_k} = 0$ //Initial counts for each itemset are 0.
for each $t_j \in D$ do
for each $I_k \in C_i$ do
if $I_k \in t_j$ then
$Sup_{I_k} = Sup_{I_k} + 1$;
for each $I_k \in C_i$ do
if $Sup_{I_k} \geq (\sup_{\min} \times |D|)$ do
$L_i = L_i \cup I_k$;
$L = L \cup L_i$;
$C_{i+1} = Apriori - Gen(L_i)$
until $C_{i+1} = \phi$

One efficient implementation was introduced by [Bor03] using data structure trie. We use a simple example with a transaction database with five items $I = \{I_1, I_2, I_3, I_4, I_5\}$ to explain how trie data structure is used in the association rule mining.

Figure 7.1 presents the trie that stores the candidates $\{I_1, I_2, I_4\}$, $\{I_2, I_3, I_4\}$, $\{I_2, I_3, I_5\}$, and $\{I_2, I_4, I_5\}$, where 2-itemsets $\{I_1, I_2\}$, $\{I_1, I_4\}$, $\{I_2, I_3\}$, $\{I_2, I_4\}$, $\{I_2, I_5\}$, and $\{I_4, I_5\}$ have been found frequent and $\{I_1, I_3\}$, $\{I_1, I_5\}$, $\{I_3, I_4\}$, and $\{I_3, I_5\}$ are found not frequent and pruned from the tree. Labels 1 to 5 of the links in figure 7.1 correspond to items I_1, I_2, I_3, I_4, I_5. The problem now is to count the supports of those frequent 3-itemsets candidates. *d* in figure 7.1 denotes the depth of a trie.

In counting the support, the transactions from the database are taken one by one. For a transaction *t*, we consider all ordered *k*-subsets *X* of *t* (the items of *t* has been ordered) and search for them in the trie structure. If *X* is found as a candidate in the tree,

the support of the node corresponding to the candidate *k*-itemsets is increased by 1. One optimization that can be

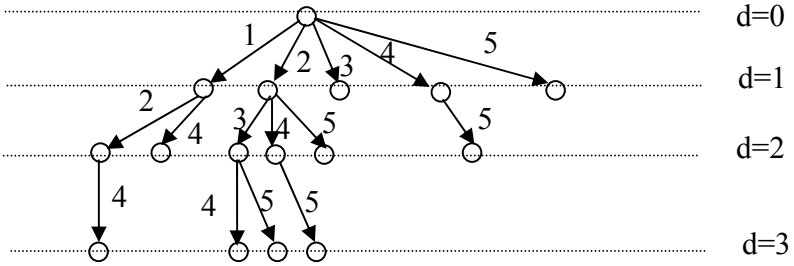
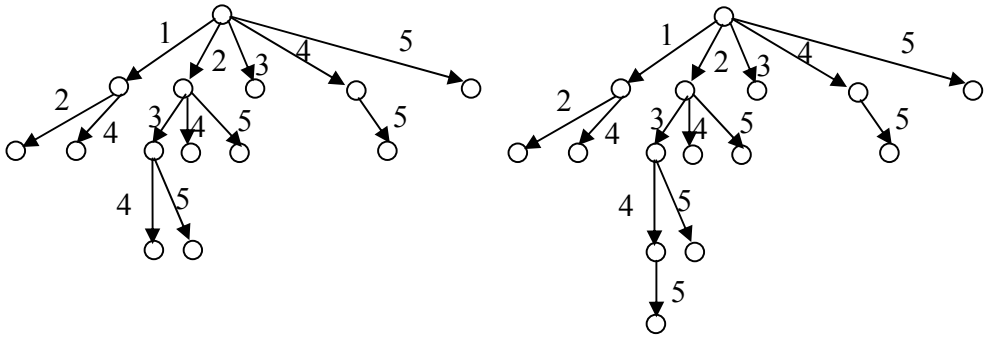


Figure 7.1 A trie Data Structure for Association Rule Mining

done is that it is not necessary to generate all *k*-subsets of *t*. If we are at a node at depth d by following the link labeled by the item with index *j* of transaction *t* (the first item of *t* is indexed as 0), moving forward on links labeled by items of *t* with index greater than *j* but less than $|t|-k+d+1$ is enough. For the above example, we can do the following when we are counting the supports for candidate 3-itemsets and if the transaction *t* is $\{I_2, I_3, I_4, I_5\}$: (1) if we are at the node following links labeled 2 at depth 1 (I_2's index is 0 in *t*), it is only necessary to move forward from links that have label 3 (I_3's index is 1 in *t*) to links that have label 4 (index 2 in *t*, $|t|-k+d+1=4-3+1+1=3$). (2) if we are at the node following links labeled 3 (index 1 in *t*) at depth 2, it is only necessary to move forward from links that have label 4 to links that have label 5 (index 3 in *t*, $|t|-k+d+1=4-3+2+1=4$). After the whole database is read, the supports of the candidates of *k*-itemsets are known. If the support of a candidate of *k*-itemsets does not meet the minimum support requirement, it is removed from the tree. For the above example, if it is found that item sets $\{I_1, I_2, I_4\}$ and $\{I_2, I_4, I_5\}$ do not have enough

supports, then they are removed. Figure 7.2 (a) denotes the trie after the two candidates are pruned.



(a) (b)

Figure 7.2 A trie (a) after Pruning (b) After generating new candidate

After the frequent *k*-itemsets are found, we can generate the candidates for frequent (*k*+1)-itemsets. Generating candidates for frequent (*k*+1)-itemsets is completed by simply copying the sibling nodes following links labeled with items (which are later in the given item order) as child nodes of the nodes following links labeled with items (which are earlier in the order). For the previous example, only one candidate $\{I_2, I_3, I_4, I_5\}$ can be generated by copying the node following the link labeled by 5 as the child node of the node following the link labeled by 4. Figure 7.2(b) illustrates the trie after a new candidate is generated.

7.1.2 Mining Association Rules from Randomized data

Mining association rules from randomized data is discussed in this section. First, we introduce how to implement PRAM simultaneously to a group of items in section 7.1.2.1. In section 7.1.2.2, estimator for support of candidate itemsets is provided. Finally,

implementation of finding frequent itemsets from randomized data set using data structure trie is discussed.

7.1.2.1 Randomization

Let D be a dataset over a set of items I. Every row in D represents a record, which is an instance of the items. To protect the specific values in D, we apply a randomization process on D, and output a randomized version D' of D.

The randomization is done as follows:

(1) The items are divided into several groups. How to group the items is a matter of design choice, which depends largely on the specific privacy concerns for the different items. Only those items that are owned by a single owner can be grouped together. For example, if $\{I_1, I_2, I_3, I_4, I_5\} \subset I$, $\{I_1, I_2, I_3, I_4, I_5\}$ can be divided into two groups, $\{I_1, I_2, I_4\}$ and $\{I_3, I_5\}$.

(2) Apply PRAM independently to each record and each group. A set SG_G of all possible combination of items in one group G is randomized using PRAM schemes discussed in chapter 2. The randomization is characterized by a $2^{|G|} \times 2^{|G|}$ probability transition matrix $P_G = \{p_{kl}\}$, where p_{kl} is the probability with which combination $s_l \in SG_G$ is randomized to $s_k \in SG_G$. We discuss only the symmetric M-categorical randomization in the sequel.

Taking the above example:

(a) The set of all possible combination of items in group $G = \{I_1, I_2, I_4\}$ is $SG_G = \{\overline{I}_1\overline{I}_2\overline{I}_4, \overline{I}_1\overline{I}_2\overline{I}_4, \overline{I}_1 I_2\overline{I}_4, \overline{I}_1 I_2 I_4, I_1\overline{I}_2\overline{I}_4, I_1\overline{I}_2 I_4, I_1 I_2\overline{I}_4, I_1 I_2 I_4\}$, where ‘ − ’ denotes the

transaction does not contain the given item. The randomization of any $s_i \in SG_{\{I_1,I_2,I_4\}}$ ($i = 1 \cdots 8$) in a transaction t can be done by the randomization scheme characterized by a 8×8 probability transition matrix $P_G = \{p_{ij}\}$, where

$$\begin{cases} p_{ii} = p(\tilde{G} = s_i \mid G = s_i) = (1-p) \\ p_{ij} = p(\tilde{G} = s_i \mid G = s_j) = \dfrac{p}{7} \qquad i \neq j \end{cases}$$ for $1 \le i, j \le 8$ and $\tilde{G}$ denote the randomized item group.

(b) The set of all possible combination of the item group $G = \{I_3, I_5\}$ is $SG_G = \{\overline{I}_3\overline{I}_5, \overline{I}_3 I_5, I_3\overline{I}_5, I_3 I_5\}$. The randomization of any $s_i \in SG_G$ ($i = 1 \cdots 4$) in a transaction t can be done by the randomization schemes characterized by a 4×4 probability transition matrix $P_G = \{p_{ij}\}$, where $\begin{cases} p_{ii} = p(\tilde{G} = s_i \mid G = s_i) = (1-p) \\ p_{ij} = p(\tilde{G} = s_i \mid G = s_j) = \dfrac{p}{3} \qquad i \neq j \end{cases}$ for $1 \le i, j \le 4$.

7.1.2.2 Estimating Support of k-itemsets

After partitions of a data set D are randomized by their respective owners, they are sent to a data miner. How the items are grouped and each group's probability transition matrix is also sent to the data miner. The task of the data miner is to estimate supports of candidate k-itemsets, learn all frequent k-itemsets from the randomized data $\tilde{D}$ and then generate the association rules from all frequent itemsets. The main difference between mining association rules from $\tilde{D}$ and D lies in estimating supports of candidate k-itemsets.

(a) Estimation of supports of single itemsets

From the probability transition matrix P_G of a group of items G, we can get the marginal transition probability matrix P_{I_i} for any item $I_i \in G$, where $i = 1 \cdots |G|$ and $P_{I_i} = \begin{bmatrix} p(\bar{I}_i \mid \bar{I}_i) & p(I_i \mid \bar{I}_i) \\ p(\bar{I}_i \mid I_i) & p(I_i \mid I_i) \end{bmatrix}$. We have the following proposition regarding P_{I_i}.

Proposition 7.1:

If the set of all possible combinations of items in a group $G = \{I_1, I_2, \cdots I_{|G|}\}$ is randomized by using symmetric *M*-categorical randomization with parameter *p*, then the marginal transition probability matrix P_{I_i} has two properties:

(1) $p(\bar{I}_i \mid \bar{I}_i) = p(I_i \mid I_i) = 1 - p + \frac{p(\frac{J}{2} - 1)}{J - 1}$ and $p(I_i \mid \bar{I}_i) = p(\bar{I}_i \mid I_i) = \frac{pJ}{2(J-1)}$, where $J = 2^{|G|}$.

(2) Denote by T_s, $\tilde{T}_s$, and $\hat{T}_s$, supports of $s \in SG_G$ in D, $\tilde{D}$ and estimate of T_s, respectively. In this case, $SG_G = \{\bar{I}_i, I_i\}$. $\begin{bmatrix} \hat{T}_{\bar{I}_i} \\ \hat{T}_{I_i} \end{bmatrix} = P_{I_i}^{-1} \begin{bmatrix} \tilde{T}_{\bar{I}_i} \\ \tilde{T}_{\bar{I}_i} \end{bmatrix}$ is an unbiased estimator of $\begin{bmatrix} T_{\bar{I}_i} \\ T_{I_i} \end{bmatrix}$. This case is a special case of case (b) below, where proofs are provided. The estimate of support of item I_i is $\hat{T}_{I_i}$. For example (a) in section 7.1.2.1,

$P_{I_i} = \begin{bmatrix} 1-\frac{4}{7}p & \frac{4}{7}p \\ \frac{4}{7}p & 1-\frac{4}{7}p \end{bmatrix}$ for $i = 1, 2, 4$ and for example (b) in section 7.1.2.1,

$P_{I_i} = \begin{bmatrix} 1-\frac{2}{3}p & \frac{2}{3}p \\ \frac{2}{3}p & 1-\frac{2}{3}p \end{bmatrix}$ for $i = 3, 5$.

(b) Estimation of supports of k- itemsets for $k > 1$

The general case of estimating support of a candidate *k*-itemsets is discussed below. Suppose there are *N* records in a data set D of a set of items I with size *m* and the *m* items are grouped into *g* groups, $G_1, G_2 \cdots G_g$. Let T_C be a vector of supports of all possible combinations SG_C of items in a candidate *k*-itemset C, $\tilde{T}_C$ be the corresponding vector of supports in $\tilde{D}$ and $\hat{T}_C$ be an estimator of T_C. We have the following two cases for obtaining $\hat{T}_C$.

Case (1) All items of the candidate *k*-itemsets C are from a single group

In this case, we have $C \subset G$. From the probability transition matrix P_G of the group *G*, we can get the probability transition matrix P_C, which is the marginal probability transition matrix for items in *C*. We have the following proposition.

Proposition 7.2

(1) $P_C = \begin{bmatrix} p_{11} & \cdots & p_{1J} \\ \vdots & \ddots & \vdots \\ p_{J1} & \cdots & p_{JJ} \end{bmatrix}$, where $J = 2^{|C|}$, $p_{ii} = (1-p) + \frac{p}{2^{|G|}-1}(2^{|G|-|C|}-1)$ for $i = 1 \cdots J$ and

$p_{ij} = \frac{p}{2^{|G|}-1} 2^{|G|-|C|}$ for $i \neq j$ and $i, j = 1 \cdots J$.

(2) $\hat{T}_C = P_C^{-1} T_C$ is an unbiased estimator of T_C.

Proof: $s_i \in SG_C$ $(i = 1, \cdots 2^{|C|})$ is one of the possible itemsets. In the following, we denote by C_c the items that belong to G but not C.

$$p(\tilde{C} = s_i \mid C = s_i) = \frac{p(\tilde{C} = s_i, C = s_i)}{p(C = s_i)} = \frac{\sum_{k=1}^{2^{|C_c|}} \sum_{j=1}^{2^{|C_c|}} p(\tilde{C} = s_i, \tilde{C}_c = w_k, C = s_i, C_c = w_j)}{\sum_{j=1}^{2^{|\bar{C}|}} p(C = s_i, C_c = w_j)}$$

$$= \frac{\sum_{k=1}^{2^{|C_c|}} \sum_{j=1}^{2^{|C_c|}} p(\tilde{C} = s_i, \tilde{C}_c = w_k \mid C = s_i, C_c = w_j) p(C = s_i, C_c = w_k)}{\sum_{j=1}^{2^{|C_c|}} p(C = s_i, C_c = w_j)}$$

From P_G, $p(\tilde{C} = s_i, \tilde{C}_c = w_k \mid C = s_i, C_c = w_j) = \begin{cases} 1-p & k = j \\ \dfrac{p}{2^{|G|}-1} & k \neq j \end{cases}$.

$$\sum_{k=1}^{2^{|C_c|}} \sum_{j=1}^{2^{|C_c|}} p(\tilde{C} = s_i, \tilde{C}_c = w_k \mid C = s_i, C_c = w_j) p(C = s_i, C_c = w_j)$$

$$= \sum_{j=1}^{2^{|C_c|}} [p(C = s_i, C_c = w_j)(1 - p + \frac{2^{|C_c|}-1}{2^{|G|}-1} p)] = (1 - p + \frac{2^{|C_c|}-1}{2^{|G|}-1} p)$$

$$= (1 - p + \frac{2^{|C_c|}-1}{2^{|G|}-1} p) \sum_{j=1}^{2^{|C_c|}} p(C = s_i, C_c = w_j)$$

So, $p(\tilde{C} = s_i \mid C = s_i) = (1 - p + \frac{2^{|C_c|}-1}{2^{|G|}-1} p) = 1 - p + p\frac{2^{|G|-|C|}-1}{2^{|G|}-1}$. Also,

$$p(\tilde{C} = s_m \mid C = s_i) = \frac{p(\tilde{C} = s_m, C = s_i)}{p(C = s_i)} = \frac{\sum_{k=1}^{2^{|C_c|}} \sum_{j=1}^{2^{|C_c|}} p(\tilde{C} = s_m, \tilde{C}_c = w_k, C = s_i, C_c = w_j)}{\sum_{j=1}^{2^{|\bar{C}|}} p(C = s_i, C_c = w_j)}.$$

From P_G, $p(\tilde{C} = s_m, \tilde{C}_c = w_k, C = s_i, C_c = w_j) = \frac{p}{2^{|G|}-1}$.

So, $p(\tilde{C} = s_m \mid C = s_i) = 2^{|C_c|}\frac{p}{2^{|G|}-1} = \frac{2^{|G|-|C|}p}{2^{|G|}-1}$.

Hence, $P_C = \begin{bmatrix} p_{11} & \cdots & p_{1J} \\ \vdots & \ddots & \vdots \\ p_{J1} & \cdots & p_{JJ} \end{bmatrix}$, where $J = 2^{|C|}$, $p_{ii} = (1-p) + \frac{p}{2^{|G|}-1}(2^{|G|-|C|}-1)$ for $i = 1 \cdots J$

and $p_{ij} = \frac{p}{2^{|G|}-1}2^{|G|-|C|}$ for $i \neq j$ and $i, j = 1 \cdots J$.

Part (2) of proposition 7.2 directly follows from Theorem 2 of chapter 2 for estimation of frequency counts.

Case (2) Items of the candidate C are from several groups

Suppose items of the candidate C are from m groups, $C_1 = G_1 \cap C, \cdots, C_m = G_m \cap C$. We can get the marginal probability transition matrices $P_{C_1} \cdots P_{C_m}$ from $P_{G_1}, \cdots, P_{G_2}$, respectively, as in case (1). We have the following proposition.

Proposition 7.3

$\hat{T}_C = P_C^{-1}\tilde{T}_C$ is an unbiased estimator of T_C and $P_C = P_{C_1} \otimes P_{C_2} \cdots P_{C_m}$, where elements of T_C are arranged in the order such that items in C_1 change the slowest and items in C_m change the fastest.

Proposition 7.3 also follows directly from Theorem 2 in chapter 2. The last element of $\hat{T}_C$, i.e. $\hat{T}_C(2^{|C|})$ is the estimate of support of the k-itemsets and is denoted as $\widehat{\sup}_C$.

7.1.2.3 Implementation of Apriori Algorithm in Mining from Randomized Data

From the previous section 7.1.2.2, every *k*-itemset candidate *C* is associated with a probability transition matrix P_C, which can be computed from $P_{G_1}, \cdots, P_{G_g}$.

A trie-based data structure can be used but it needs some modification. The main adaptation is adding an estimation step that estimates the support of candidate *k*-itemsets before non-frequent candidate *k*-itemsets are pruned from the tree. Differences between mining association rules from randomized data and from (non randomized) data directly are (1) Reference to supports of nodes that are in higher levels is necessary (2) Supports of itemsets $\widetilde{\sup}_C$ are counted from $\tilde{D}$ using trie data structure, where $\widetilde{\sup}_C$ is the vector of supports of all possible itemsets formed by the items in *C*. Let IS_C denote a set of all possible itemsets formed by the items in *C*. IS_C is related to but different from GS_C. For example, if $C = \{I_1 I_2\}$, $GS_C = \{\overline{I}_1\overline{I}_2, \overline{I}_1 I_2, I_1\overline{I}_2, I_1 I_2\}$ whereas $IS_C = \{I_1, I_2, I_1 I_2\}$. $\widetilde{\sup}_C$ is counted using the trie data structure. However, $\tilde{T}_C$ is not directly counted but can be calculated from $\widetilde{\sup}_C$. The inclusion/exclusion principle from set theory can be used to get $\tilde{T}_C$ from the supports of itemsets $\widetilde{\sup}_C$ that includes the supports of nodes in higher level.

Proposition 7.4

(1) For any $s_i \in SG_C$, s_i can be written as $C_2\overline{C}_1$, where $C_1, C_2 \in IS_C$, $\tilde{T}_{S_i} = \sum_{C_2 \subseteq C_3 \subseteq C} (-1)^{|C_3|-|C_2|} \widetilde{\sup}_{C_3}$. Thus, $\tilde{T}_C$ can be calculated from $\widetilde{\sup}_C$.

(2) $\hat{T}_C = P_C^{-1}\tilde{T}_C$ is an unbiased estimator of T_C and $\hat{T}_C(2^{|C|})$, i.e. the $2^{|C|}$ th element of vector $\hat{T}_c$ is an estimator for Sup_C, i.e., $\widehat{S\,\mathrm{up}}_C$.

If $\widehat{S\,\mathrm{up}}_C < \sup_{\min}|D|$, then the itemsets C is not frequent and will be pruned.

We use the previous example with a transaction database of five items $I = \{I_1, I_2, I_3, I_4, I_5\}$ to explain how association rule mining from randomized data is implemented. The following steps are used.

(1) In depth 1 of the tree level, the whole database is traversed once and $\widetilde{\sup}_{I_i}$ for $i = 1, \cdots, 5$ are obtained. P_{I_i} for $i = 1, \cdots, 5$ can be calculated from group probability transition matrices given by the data owners. We have $\tilde{T}_{I_i} = \begin{bmatrix} N - \widetilde{S\,\mathrm{up}}_{I_i} \\ \widetilde{S\,\mathrm{up}}_{I_i} \end{bmatrix}$, where N is the number of records in D. $\hat{T}_{I_i} = \begin{bmatrix} N - \widehat{S\,\mathrm{up}}_{I_i} \\ \widehat{S\,\mathrm{up}}_{I_i} \end{bmatrix}$ and $\hat{T}_{I_i} = P_{I_i}^{-1}\tilde{T}_{I_i}$ for $i = 1, \cdots, 5$. $\hat{T}_{I_i}(2)$ is $\widehat{S\,\mathrm{up}}_{I_i}$.

If $\widehat{S\,\mathrm{up}}_{I_i} \geq |D|\sup_{\min}$, 1-itemsets $\{I_i\}$ are frequent and otherwise not. Generate candidates of 2-itemsets from a set L_1 of all frequent 1-itemsets. If only $\{I_3\}$ is not frequent, $L_1 = \{\{I_1\}, \{I_2\}, \{I_4\}, \{I_5\}\}$ and a set of all candidate frequent 2-itemset generated from L_1 is $C_2 = \{\{I_1, I_2\}, \{I_1, I_4\}, \{I_1, I_5\}, \{I_2, I_4\}, \{I_4, I_5\}\}$.

(2) The randomized database is read once again. We get $\widetilde{\sup}_C$ for any $C \in R_2$. Take a candidate $C = \{I_1, I_2\} \in R_2$, for example $SG_C = \{\bar{I}_1\bar{I}_2, \bar{I}_1 I_2, I_1\bar{I}_2, I_1 I_2\}$. From $\widetilde{\sup}_C$, $\widetilde{\sup}_{I_1}$ and

$\widetilde{\sup}_{I_2}$ (from last step), $\tilde{T}_C = \begin{bmatrix} N - \widetilde{\sup}_{I_2} - \widetilde{\sup}_{I_1} + \widetilde{\sup}_C \\ \widetilde{\sup}_{I_2} - \widetilde{\sup}_C \\ \widetilde{\sup}_{I_1} - \widetilde{\sup}_C \\ \widetilde{\sup}_C \end{bmatrix}$. P_C can be computed from the group probability transition matrices given by data owners. $\hat{T}_C = P_C^{-1}\tilde{T}_C$ and $\hat{T}_C(4)$ is $\widehat{\sup}_C$, which is the estimate of the support of candidate itemset C. If $\widehat{S\sup}_C \geq \sup_{\min}|D|$ then $\{I_1, I_2\}$ is a frequent 2-itemsets. In this way, we can get L_2, a set of all frequent 2-itemsets. From L_2, a set C_3 of all candidate 3-itemsets can be generated.

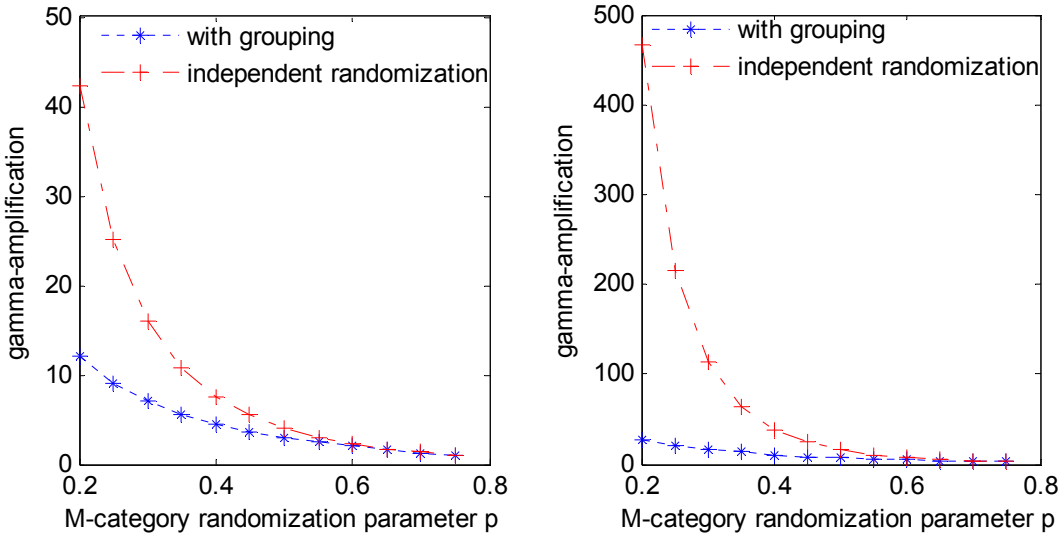
(3) Estimate the support of those candidate 3-itemsets and then obtain L_3 and generate C_4. The above steps are repeated similarly until all frequent itemsets are found.

7.1.3 Comparing γ-amplification

As we have pointed out at the beginning of section 7.1, simultaneous randomization will reduce simultaneous $\gamma - amplification$ for the items and help limit more privacy breaches to a group of items simultaneously (compared with using the same $\gamma - amplification$ for each individual item). Simultaneous $\gamma -$ amplification is defined similar to $\gamma -$ amplification but the amplification is defined on a set of variables. A PRAM operator for a set of variables $\{X_1, \cdots X_n\}$ with transition probability matrix P is at most simultaneous γ-amplifying for $\{\tilde{X}_1 = k_1, \cdots, \tilde{X}_n = k_n\}$ if $\forall l_1, \ldots, l_2$ and $\forall m_1, \cdots, m_n$, $\frac{p(\tilde{X}_1 = k_1, \cdots, \tilde{X}_n = k_n \mid X_1 = l_1, \cdots, X_n = l_n)}{p(\tilde{X}_1 = k_1, \cdots, \tilde{X}_n = k_n \mid X_1 = m_1, \cdots, X_n = m_n)}$, where $k_1, l_1, m_1 < |X_1|, \cdots k_n, l_n, m_n < |X_n|$. A PRAM operator is at most γ-amplifying for variable

$X = \{X_1, \cdots, X_n\}$ if it is at most γ-amplifying for $\forall k_1, \cdots, k_n$. As we proved in section 7.1.2, if the symmetric *M*-categorical randomization with parameter *p* is applied to a set of items, we can get the marginal probability matrix for each item. By the definition of simultaneous γ-amplification, the symmetric *M*-categorical randomization has simultaneous $\gamma_s = \frac{1-p}{p}(2^{|G|}-1)$ if $1-p > \frac{p}{2^{|G|}-1}$. The marginal probability transition matrix for each item is $P_{I_i} = \begin{bmatrix} p(\bar{I}_i \mid \bar{I}_i) & p(I_i \mid \bar{I}_i) \\ p(\bar{I}_i \mid I_i) & p(I_i \mid I_i) \end{bmatrix}$, where $p(\bar{I}_i \mid \bar{I}_i) = p(I_i \mid I_i) = 1-p+\frac{p(2^{|G|-1}-1)}{2^{|G|}-1}$ and $p(I_i \mid \bar{I}_i) = p(\bar{I}_i \mid I_i) = \frac{p2^{|G|}}{2(2^{|G|}-1)}$. If we had applied randomization schemes with probability transition matrix P_{I_i} independently to $|G|$ items, then simultaneous $\gamma - amplification$ will be $(\gamma_{ind})^{|G|}$ where $\gamma_{ind} = \frac{1-p+\frac{p(2^{|G|-1}-1)}{2^{|G|}-1}}{\frac{p2^{|G|}}{2(2^{|G|}-1)}}$ if $1-p+\frac{p(2^{|G|-1}-1)}{2^{|G|}-1} > \frac{p2^{|G|}}{2(2^{|G|}-1)}$.

If $|G| = 2$, $\gamma_s = \frac{3(1-p)}{p}$ whereas $(\gamma_{ind})^{|G|} = (\frac{1-\frac{2}{3}p}{\frac{2}{3}p})^2$. If $|G| = 3$, $\gamma_s = \frac{7(1-p)}{p}$ whereas $(\gamma_{ind})^{|G|} = (\frac{1-\frac{4}{7}p}{\frac{4}{7}p})^3$. The following figure 7.3 gives γ_s and $(\gamma_{ind})^{|G|}$ versus p. From the graphs, we can see that simultaneous randomization significantly reduces simultaneous $\gamma - amplification$.



(a) (b)

Figure 7.3 comparison of simultaneous $\gamma - amplification$

7.1.4 Experimental Results

We applied association rule mining to datasets from the UCI repository [UCI]. Since learning from randomized data usually requires more samples than regular learning, we generated a larger data set for our simulation based on the data set from UCI data repository. The two data sets we used are the Mushroom database and Adult database.

The results are shown respetively in figure 7.4 and figure 7.5.

Experiment 1:

We selected 9 variables from Mushroom database. Each category of those variables is converted into an Item. Totally there are 23 items. We generated a dataset *D* by repeating the above data set 30 times. The 23 items are grouped into 13

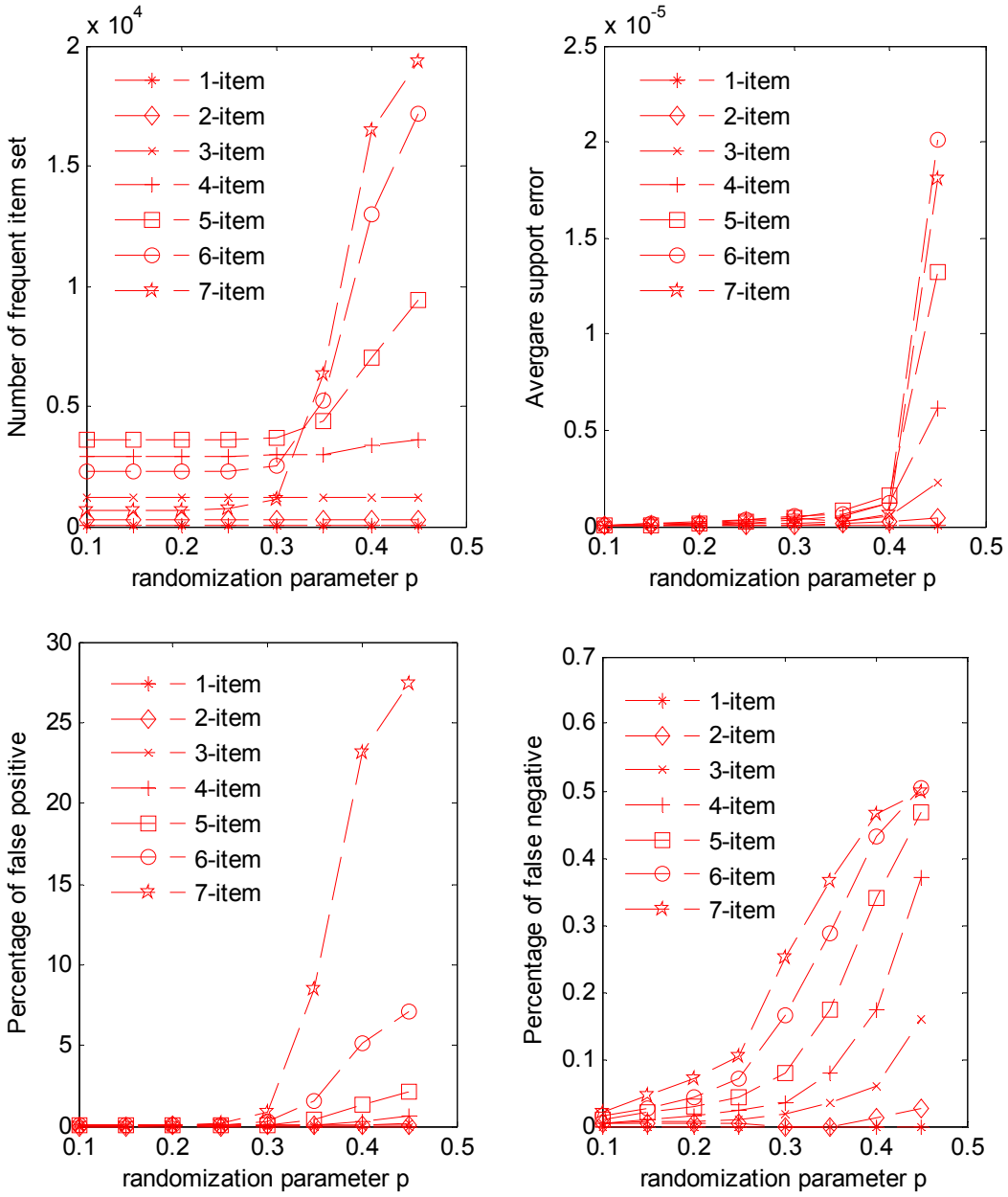


Figure 7.4 Simulation results for data set based on Mushroom database

groups (four 3-item groups, two 2-item groups and seven 1-item groups). The database was randomized by implementing symmetric *M*-category randomization, independently for each group. The parameter of the symmetric *M*-categoy randomization p was varied from 0.1 to 0.5. The simulation results are shown in figure 7.4, where the four graphs are

respectivly plots of Number of frequent itemsets learned, Average support error of the frequent itemsets, the Percentage of false positive and Percentage of false negative versus the randomization parameter p. $\sup_{\min} = 0.05$ was used in the simulations.

Experiment 2:

We selected 11 variables from the Adult dataset. Those variables were transformed into variables with respective cardinalities 4,3,3,3,2,3,2,3,2,3,2 by combining categories. Each category is converted into an item. Totally there are 30 items. We generated a dataset D by repeating the above data set 30 times. The 30 items are grouped into 17 groups (four 3- item groups, five 2-item groups, and eight 1-item groups). The dataset was randomized by implementing symmetric M-category randomization, independently to each group. The parameter of the symmetric M-categoy randomization p was varied from 0.1 to 0.45. The simulation results are shown in figure 7.5, where the four graphs are respectivly plots of Number of frequent itemsets learned, Average support error of the frequent itemsets, the Percentage of false positive and Percentage of false negative versus the randomization parameter p. $\sup_{\min} = 0.05$ was used in simulations.

From the graphs, we can see that the results are reasonable when randomization parameter is less than 0.4. With higher randomization parameter, the worst error is the percentage of false positive, which is the ratio between the number of frequent itemsets found in the randomized data but are actually not frequent and the number of frequent itemsets found from the original non-randomized data set.

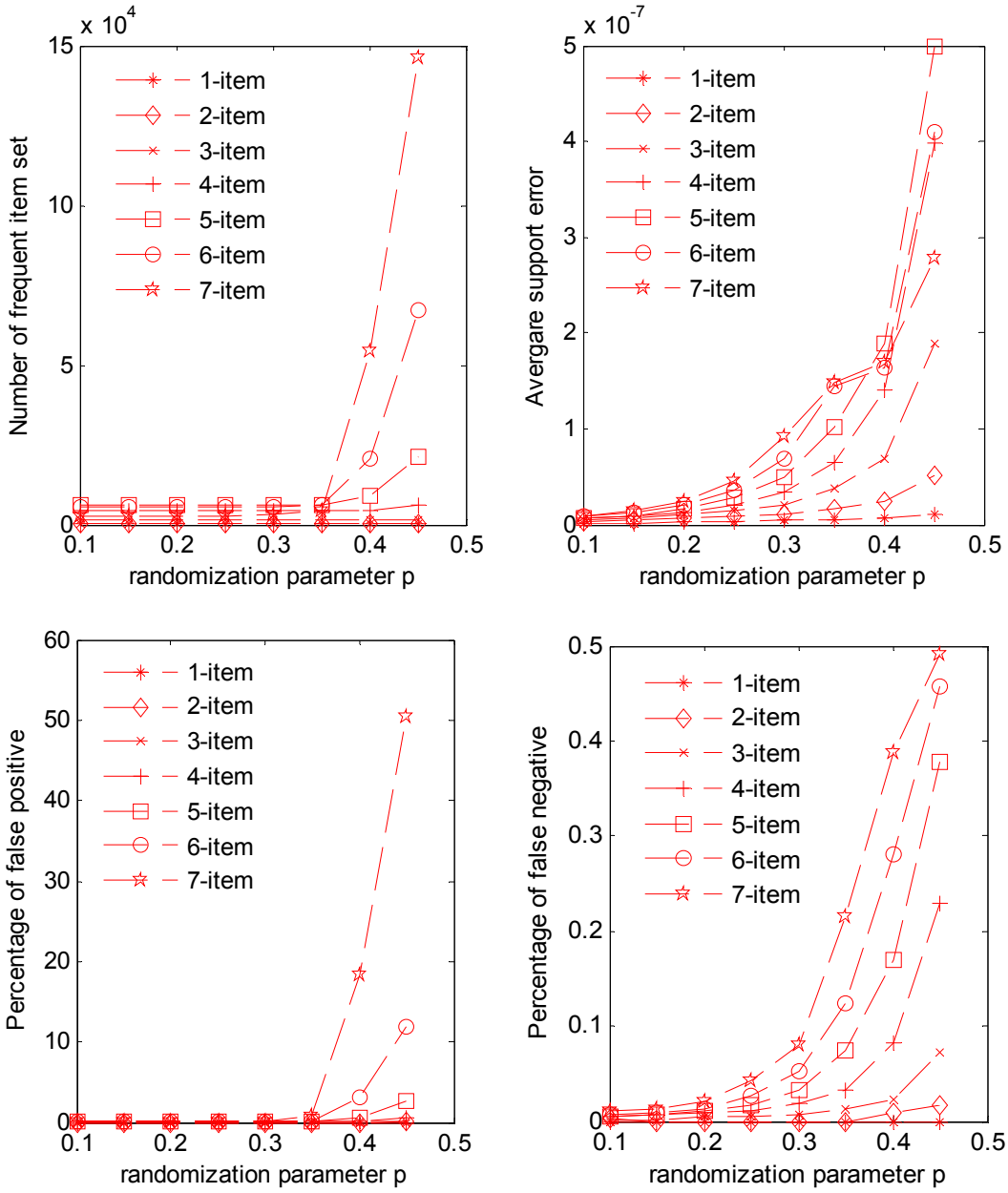


Figure 7.5 Simulation Results for data set based on Adult database

7.2 Decision Tree Induction

In this section, decision tree induction from randomized data for PPDM is discussed. A brief introduction to decision trees and ID3 algorithm for decision tree

learning is provided in section 7.2.1. Building a decision tree from randomized data is discussed in section 7.2.2. Simulation results are provided in section 7.2.3.

7.2.1 Introduction to Decision Tree and ID3 algorithm

A decision tree is a predictive modeling technique used in classification, clustering and prediction tasks. Decision trees use a "divide and conquer" technique to split the problem search space into subsets. Formally, a decision tree is a tree where the root and each internal node is labeled with a question, the arcs emanating from each node represent each possible answer to the associated question. Each leaf represents a prediction of a solution to the problem under consideration. In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the root to a leaf corresponds to a conjunction of attribute tests and the tree itself corresponds to a disjunction of these conjunctions. An example of a decision tree is illustrated in figure 7.6 [Qui93]. This decision tree classifies "Saturday mornings" according to whether they are suitable for playing tennis.

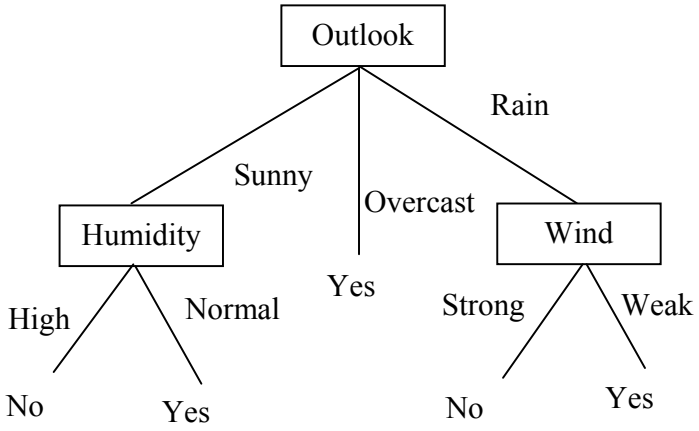


Figure 7.6 Decision Tree

The decision tree is most useful in classification problems. Once a tree is built, it is applied to each tuple in the database and results in a classification for that tuple. We discuss only learning decision tree from randomized data for preserving privacy of each individual record.

ID3 algorithms [Qui93] are among some of the most widely used algorithms for building decision trees from data. Their popularity is due in part to their ability to: select from all attributes used to describe the data, a subset of attributes that are relevant for classification; identify complex predictive relations among attributes; and produce classifiers that are easy to comprehend for humans. The ID3 (Iterative Dichotomizer 3) algorithm proposed by Quinlan [Qui93] and its more recent variants represent a widely used family of decision tree learning algorithms. The ID3 algorithm searches in a greedy fashion, for attributes that yield the maximum amount of information for determining the class membership of instances in a training set D of labeled instances. The result is a decision tree that correctly assigns each instance in D its respective class. The construction of the decision tree is accomplished by recursively partitioning D into subsets based on values of the chosen attribute until each resulting subset has instances that belong to exactly one of the m classes. The selection of an attribute at each stage of construction of the decision tree maximizes the estimated expected information gained from knowing the value of the attribute in question.

Consider a set of instances D which is partitioned into m disjoint subsets (classes) $D_1 \cdots D_m$ such that $D = \bigcup_{i=1}^{m} D_i$ and $D_i \cap D_j = \phi$ if $i \neq j$, where $1 \leq i, j \leq m$. The probability that a randomly chosen instance $x \in D$ belongs to the class D_j is $p_j = \frac{|D_j|}{|D|}$,

where $|\ |$ denotes the cardinality of a set. The entropy of a set D measures the expected information needed to identify the class membership of instances in D, and is defined as follows:

$$entropy(D) = -\sum_{j} \frac{|D_j|}{D} \log_2 (\frac{|D_j|}{|D|}).$$

We can define the estimated information gain for an attribute A, relative to a collection of instances D as follows:

$$IGain(D, A) = entropy(D) - \sum_{k \in V(A)} \frac{|D_v|}{|D|} entropy(D_k),$$

where $V(A)$ is the set of all possible values of attribute A, D_k is a subset of D for which attribute A has value k.

Thus, the information requirements of ID3-like decision tree learning algorithms can be expressed in terms of relative frequencies computed from the relevant instances at each node. Different algorithms for decision tree induction differ from each other in terms of the criterion that is used to evaluate the splits that correspond to tests on different candidate attributes. The choice of the attribute at each node of the decision tree greedily maximizes (or minimizes) the chosen splitting criterion. The ID3 algorithm is described below where D represents the training samples and AL represents the attributes list:

$ID3(D, AL)$

1. Create a node V.

2. If D consists of samples with all the same class C then return V as a leaf node labeled with class C.

3. If AL is empty, then return V as a leaf-node with the majority class in D.

4. Select test attributes (TA) among the AL with the highest information gain.

5. Label node V with TA

6. For each know value k of TA

(a) Grow a branch from node V for the condition $TA = k$.

(b) Let D_k be the set of samples in D for which $TA = k$

(c) If D_k is empty then attaches a leaf labeled with the majority class in D.

(d) Else attach the node returned by $ID3(D_k, AL - TA)$.

7.2.2 **Building Decision Tree from Randomized Data**

If the data set is horizontally distributed among several parties, every party can directly send sufficient statistics of their part to the data miner (or other parties) during construction of a decision tree, i.e., send $|D_k|$s and $|D|$s to the data miner which are necessary in computing the information gain and entropy. Disclosing $|D_k|$s and $|D|$s will not disclose the individual value of each instance. For the analysis and learning algorithm, there are no essential differences between the heterogeneously distributed dataset and the randomized version $\tilde{D}$ of the dataset D, where each attribute has been randomized using some PRAM scheme known to the data miner. Every party who owns part of the data set simply randomizes their data according to their privacy concerns. To keep things simple, we assume that all the attributes are discrete or categorical. However, all the discussion below can be easily generalized to numerical attributes by using proper discretization. Often, decision tree algorithms also include a pruning phase to alleviate the problem of over fitting the training data. We limit our discussion to decision tree construction

without pruning. The proposed algorithm is similar to [DZ03]. Every attribute in [DZ03] is assumed to be binary. The randomization used in [DZ03] is a randomized response technique, where the randomization is done by the customer who is surveyed. The customer, either (1) tells the truth for all attributes or (2) tells the opposite for all attributes. In this randomization scheme, if one attribute value of a record happened to be known to other parties, the whole record will be known to those parties. So, the probability of privacy breach is quite high since the knowledge of one attribute will result in knowledge of all other attributes. We provide a more general and flexible framework to deal with privacy-preserving decision tree induction using PRAM, where the undesirable properties of randomization mentioned above can be prevented easily.

Assume that given a partially constructed decision tree, we want to choose the best attribute for the next split. Let $a_j(\pi)$ denote the attribute at the j th node along a path π starting from the attribute $a_1(\pi)$ that corresponds to the root of the decision tree, leading up to the node in question $a_l(\pi)$ at depth l. Let $k_j(\pi)$ be the value of the attribute $a_j(\pi)$ along the path π and the cardinality of the set of all possible values of $a_j(\pi)$ be K_j. Let C be the class attribute, which can take m values $C_1, \cdots, C_m$. If the data are not randomized, the samples to be considered for adding a node below $a_l(\pi)$ have the constraints of the values of attributes $a_1(\pi) = k_1(\pi), \cdots, a_l = k_l(\pi)$. It is straightforward to calculate the information gains for a candidate attribute A (for node at level $l+1$) if the data is not randomized. When the data are randomized, we have to estimate the information gain through an estimation of a set of frequency counts. We use entropy to get the information gain. It is obvious that $entropy(D)$ and $entropy(D_k)$ in the

information can be computed by using the same operator, where $entropy(D_k)$ has further constraints on the value of candidate attributes $a = k$. Without loss of generality, we only consider an estimator of $entropy(D)$, where we need to estimate the frequency counts of the dataset such that $a_1(\pi) = k_1(\pi), \cdots, a_l = k_l(\pi)$ and $C = C_i$ for $i = 1, \cdots, m$. Estimation of $entropy(D_k)$ can be computed by estimating the frequency counts of the dataset such that $a_1(\pi) = k_1(\pi), \cdots, a_l = k_l(\pi), a = k$ and $C = C_i$ for $i = 1, \cdots, m$. The samples to be considered in estimating the information gain will be the values of those attributes of all the records if *M*-category randomization has been implemented to every attribute through the link $a_1(\pi), \cdots, a_l(\pi)$. Suppose the probability transition matrices associated with attributes $a_1(\pi), \cdots, a_l(\pi)$ and the class attribute are $P_1, \cdots, P_l$ and P_C, respectively. Let T be a vector of frequency counts from original data set *D* with dimension $m\prod_{i=1}^{l} K_i \times 1$, i.e. each element of vector T is the number of records in D such that $a_1(\pi) = k_1(\pi), \cdots, a_l = k_l(\pi)$ and $C = C_j$, where $1 \le k_1(\pi) \le K_1 \cdots 1 \le k_l(\pi) \le K_l$ and $1 \le j \le m$. The order of the elements in T is such that C changes the fastest and $a_1(\pi)$ the slowest. Let $\tilde{T}$ be the corresponding vector in $\tilde{D}$ and $\hat{T}$ be an estimate of T.

By the theorem 2 in chapter 2 for estimation of frequency counts, $\hat{T} = P^{-1}\tilde{T}$ is an unbiased estimator of T. From $\hat{T}$, we can get an estimator $\widehat{Entropy}(D) = -\sum_j \frac{\widehat{|D_j|}}{\sum_{j=1}^{m}\widehat{|D_j|}} \log_2 (\frac{\widehat{|D_j|}}{\sum_{j=1}^{m}\widehat{|D_j|}})$ of $Entropy(D)$, where $\widehat{|D_j|}$ is the estimated number of samples such that $a_1(\pi) = k_1(\pi), \cdots, a_l = k_l(\pi)$ and $C = C_j$. $\widehat{|D_j|} = \hat{T}(Index)$,

where $Index$ is the element order of $\hat{T}$ and $Index = \sum_{i=1}^{l}(k_l(\pi)-1)(m\prod_{m=1}^{l-1}K_m)+j$. Similarly, let T be a vector of frequency counts from original data set D with dimension $mK\prod_{i=1}^{l}K_i \times 1$, i.e. each element of vector T is the number of records in D such that , $a_1(\pi)=k_1(\pi),\cdots,a_l=k_l(\pi)$, $a=k$ and $C=C_j$, where $1\le k_1(\pi)\le K_1,\cdots,1\le k_l(\pi)\le K_l$, $1\le k\le K$ and $1\le j\le m$. Estimate of $entropy(D_k)$ can be computed from this $\hat{T}$ for $k=1,\cdots,K$. $\widehat{Entropy}(D_k) = -\sum_j \frac{\widehat{|D_j|}}{\sum_{j=1}^{m}\widehat{|D_j|}}.\log_2(\frac{\widehat{|D_j|}}{\sum_{j=1}^{m}\widehat{|D_j|}})$, where $\widehat{|D_j|}$ is the estimated number of samples such that $a_1(\pi)=k_1(\pi),\cdots,a_l=k_l(\pi)$, $a=k$ and $C=C_j$. $\widehat{|D_j|}=\hat{T}(Index)$, where $Index$ is the element order of $\hat{T}$ and $Index=\sum_{i=1}^{l}(k_l(\pi)-1)(MK\prod_{q=1}^{l-1}K_q)+(m-1)M+j$. With $\widehat{Entropy}(D_k)$ and $\widehat{Entropy}(D)$, we can get $IGain(D,A)$.

In the following, an example is provided to illustrate the method. We assume that a decision tree has been partially constructed as shown in Figure 7.7. The splitting node is to be decided following the link $a_1=3$ and $a_2=1$ (The cardinalities of a_1 and a_2 are $K_1=3$ and $K_2=2$). Suppose a_3 with cardinality $K_3=3$ is the candidate splitting

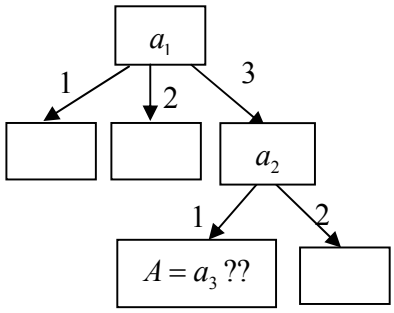


Figure 7.7 A Partially Constructed Decision Tree

attribute being considered. Let the probability transition matrices of attributes a_1, a_2, a_3 be P_1, P_2, P_3 respectively. The class attribute has cardinality 2 and its probability transition matrix is P_C. Using unbiased estimator $\hat{T} = P^{-1}\tilde{T}$, where $P = P_1 \otimes P_2 \otimes P_C$, we can calculate Entropy as follows:

$$\widehat{Entropy}(D) = -\frac{\widehat{|D_1|}}{\widehat{|D_1|}+\widehat{|D_2|}}\log_2\left(\frac{\widehat{|D_1|}}{\widehat{|D_1|}+\widehat{|D_2|}}\right) - \frac{\widehat{|D_2|}}{\widehat{|D_1|}+\widehat{|D_2|}}\log_2\left(\frac{\widehat{|D_2|}}{\widehat{|D_1|}+\widehat{|D_2|}}\right),$$ where $\widehat{|D_1|}$ and $\widehat{|D_2|}$ are $\hat{T}(9)$ and $\hat{T}(10)$ respectively since $(3-1)\times 2\times 2 + (1-1)\times 2 + 1(2) = 9(10)$.

For estimation of $Entropy(D_k)$, we use $\hat{T} = P^{-1}\tilde{T}$, where $P = P_1 \otimes P_2 \otimes P_3 \otimes P_C$.

$$\widehat{Entropy}(D_k) = -\frac{\widehat{|D_1|}}{\widehat{|D_1|}+\widehat{|D_2|}}\log_2\left(\frac{\widehat{|D_1|}}{\widehat{|D_1|}+\widehat{|D_2|}}\right) - \frac{\widehat{|D_2|}}{\widehat{|D_1|}+\widehat{|D_2|}}\log_2\left(\frac{\widehat{|D_2|}}{\widehat{|D_1|}+\widehat{|D_2|}}\right),$$ where $\widehat{|D_1|}$ and $\widehat{|D_2|}$ are $\hat{T}(25)$ and $\hat{T}(26)$ if $k=1$, $\widehat{|D_1|}$ and $\widehat{|D_2|}$ are $\hat{T}(27)$ and $\hat{T}(28)$ if $k=2$, $\widehat{|D_1|}$ and $\widehat{|D_2|}$ are $\hat{T}(29)$ and $\hat{T}(30)$ if $k=3$. With $\widehat{Entropy}(D)$ and $\widehat{Entropy}(D_k)$, we can get $IGain(D, a_3)$.

7.2.3 Experimental Results

We applied decision tree induction to a dataset from the UCI data repository[UCI]. Since learning from randomized data usually requires more samples than regular learning, we generated a larger dataset for our simulation based on the data set from UCI data repository. The two data sets we used are the Adult dataset and Breast Cancer dataset.

Experiment 1:

We used the data set generated for experiment 2 in section 7.1.4, ie. a data set generated based on the Adult database. We used those variables without treating each category of a variable as an item. The cardinality of those variables are 4,3,3,3,2,3,2,3,2,3,2 respectively. The dataset was randomized by implementing symmetric *M*-category randomization independently to each variable. The parameter of the symmetric *M*-categoy randomization p was varied from 0.1 to 0.45. The simulation results are shown in Figure 7.8, where the graph represents classification accuracy using the constructed decision tree versus the randomization parameter p. Average from five independent simulation runs and the average plus one standard deviation are shown in figure 7.8.

Experiment 2:

A dataset was generated by repeating the Breast Cancer dataset [UCI] five times. The cardinality of variables are 3 2 3 2 3 2 3 2 3 2 respectively. The dataset was randomized by implementing symmetric *M*-category randomization independently to each variable. The parameter of the symmetric *M*-categoy randomization p was varied from 0.1 to 0.45. The simulation results are shown in Figure 7.9, where the graph shows classification accuracy using the constructed decision tree versus the randomization parameter p. Average from five independent simulation runs and the average plus one standard

deviation are shown in figure 7.9.

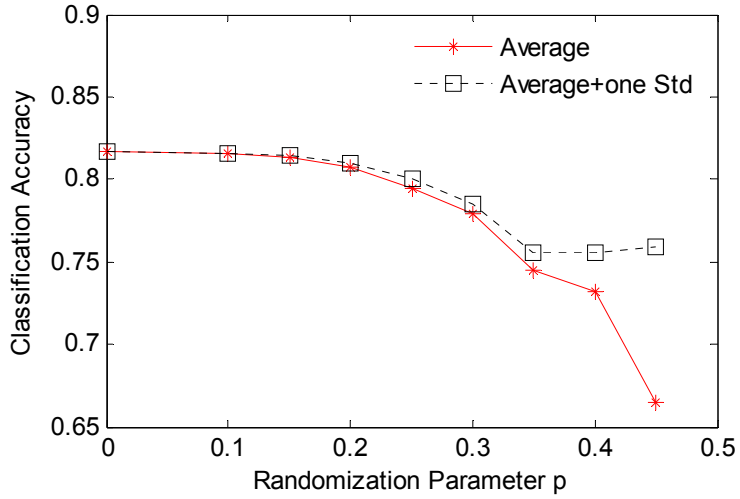


Figure 7.8: Classification Accuracy versus *p* for Data Set Based on Adult Database

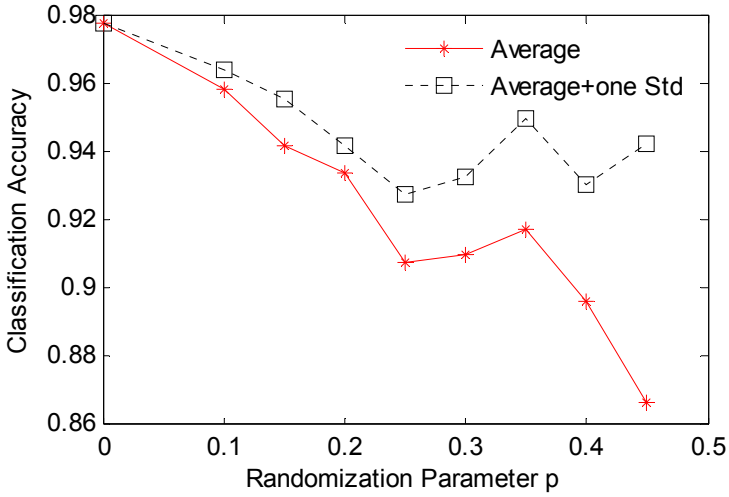


Figure 7.9: Classification Accuracy versus *p* for Data Set Based on Breast Cancer Database

From the above simulation results, we can see that there is a trade off between classification accuracy and privacy. As randomization parameter *p* increases from 0 to 0.5, more privacy is preserved. The standard deviation of simulations becomes larger when *p* increases from 0 to 0.5. The variance is caused by two factors. One is sample size

and the other is the randomization. Since the same sample size is used for simulation in each experiment, the variances for the classification accuracy are mainly due to different levels of randomization. From both experiments, we can see that the results are still reasonable when $p = 0.3$.

CHAPTER EIGHT

CONCLUSIONS AND FUTURE WORK

We have introduced a framework for PPDM using PRAM. Our method estimates frequency counts from the randomized data, which are subsequently used to learn or mine the database.

We quantified the privacy of our randomization scheme using the concept of γ-amplification and probablistic K-anonymity. Information loss due to randomization was quantified using two aspects. One is the loss of accuracy, which can be quantified by the distance between two distributions. The other aspect is independence loss.

Randomization of the dataset can be implemented independently to individual variables and can also be implemented simultaneously to a group of variables. Implementing the randomization simultaneously to a group of variables ignores the possibility of different privacy requirement for different variables. Randomizing non-sensitive variables or providing more randomization to some not so highly sensitive variables usually degrades the accuracy of data mining. However, implementing randomization simultaneously to a group of variables can also reduce the simultaneous γ-amplification, which means higher ability to limit simultaneous privacy breaches. PRAM provide a flexible framework for handling different privacy requirements. Data owners have flexibility in choosing a randomization scheme that meet their privacy conerns. Choice of different post randomization schemes only affects the specific probability transition matrix but does not affect learning algorithms.

PRAM was traditionally designed for categorical variable. Numerical variables can be discretized for the benefit of privacy using MGAS and can then be randomized using PRAM schemes.

Different data mining models like linear classifier, association rule mining, decision tree and Bayesian networks were considered. We showed that we obtain a fairly good level of privacy and reasonable accuracy in almost all cases. Our experiments show that the PRAM is an efficient, flexible and easy-to-use method in PPDM. There is a trade off between accuracy and privacy and sample size is key in this tradeoff.

We now discuss some directions for future work and some problems that need to be addressed.

(a) We discussed the learning of linear classifier from randomized data using perceptron cost. The same method can also be used to linear classifier learning by minimizing mean square error. Maximum likelihood estimation of model parameters is a widely used method in data mining. Our proposed method can further be extended to learn models based on maximum likelihood parameter estimation.

(b) Further research in Probabilistic K-Anonymity to get a better privacy quantification. In our defined K-Anonymity, any possible category that can be randomized to the given category contributes to the K^P even if this probability is extremely low. This problem can be remedied by adding reasonable threshold to this probability.

(c) There is trade off between privacy and accuracy in our method. A natural further problem is to choose optimized randomization parameters to increase privacy under the same accuracy or increase accuracy under the same privacy.

(d) In Bayesian network structure learning, we modified the score function by

introducing some penalty parameter in order to cope with the extra-link problems. It will be beneficial to do further research in how to choose those penalty parameters according to different levels of randomization.

(e) Another research direction is theoretical analysis of the perturbation method in PPDM, for example, the theoretical limit of the randomization method in PPDM.

Bibliography

[AA01] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining Algorithm. In *Proceeding of ACM SIGMOD*, pages 247-255, 2001.

[AS00] R.Agrawal and R. Srikant, Privacy preserving data mining. In *Proceeding of SIGMOD Conference on Management of Data*, pages 439-450, May 2000.

[AIS93] R. Agrawal, T. Imielinski and A. Swami. Mining associations between sets of items in massive databases. In *Proceeding of the ACM SIGMOD*, pages 207–216, 1993.

[AKPB97] J. Aronis, V. Kulluri, F. Provost, and B. Buchanan. TheWoRLD: Knowledge discovery and multiple distributed databases. In *Proceedings of the Florida Artificial Intellegence Research Symposium (FLAIRS-97)*, pages 11–14, 1997. Also available as Technical Report ISL-96-6, Intelligent Systems Laboratory, Department of Computer Science, University of Pittsburgh.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceeding of the 20th Int'l Conference on Very Large Databases*, pages 487-499, 1994.

[BKS97] E. Bauer, D. Koller, and Y. Singer. Update rules for parameter estimation in Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 3–13. Morgan Kaufmann, 1997.

[BN05] E. Bertino and I. NAIFOVINO. A framework for evaluating privacy-preserving data mining algorithms. *Journal of Data Mining and Knowledge Discovery*, No. 11, pages 121-154, 2005.

[Bor03] F. Bordon. A fast apriori implementation. In B.Goethals and M. J. Zaki, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, volume 90 of CEUR Workshop Proceedings, Melbourne, Florida, USA, 2003.

[CH92] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Journal of Machine Learning*, No. 9, pages 309–347, 1992.

[CKV03] C.Clifton, M. Kantarcioglu, J. Vaidya, X. Lin and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, Vol. 4, No. 2, pages 28-34, 2003.

[CL05] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 589–592, Houston, TX, November 2005.

[CS02] R. Chen and K. Sivakumar. A new algorithm for learning parameters of a Bayesian network from distributed data. In *Proceedings of the IEEE International Conference on Data Mining*, pages 585-588, 2002.

[CS99] V. Crestana and N. Soparkar. Mining decentralized data repositories. *Technical Report CSE-TR-385-99*, University of Michigan, Ann Arbor, MI, 1999.

[CSH03] D. Caragea, A. Silvescu and V. Honavar. A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems*, Vol. 1 No.2, 2003.

[CSK03] R. Chen, K. Sivakumar and H. Kargupta. Learning Bayesian network structure from distributed data. In *Proceedings of 2003 SIAM Conference on Data Mining*, 2003.

[CSK04] R. Chen, K. Sivakumar and H. Kargupta. Collective mining of Bayesian networks from distributed heterogeneous data . *Journal of Knowledge and Information Systems*, Vol. 6 pages 164-187, 2004

[DHC04] W. Du, Y.S. Han and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceeding of SIAM Int'l Conf. Data Mining (SDM04)*, pages 222-233, Apr. 2004.

[DM02] J.Domingo-Ferrer and J.M.Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, pages 189-201, Vol. 14, No. 1, January/Feburary 2002.

[DT05] J. Domingo-ferrer and V. Torra. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Journal of Data Mining and Knowledge Discovery*, No. 11, pages 195-212, 2005.

[DT02] J. Domingo-Ferrer and V. Torra. A quantitative comparison of disclosure control methods for microdata. In L. Zayatz, P. Doyle, J. Theeuwes and J. Lane (Editors*.), Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, North-Holland, pages 113–134., 2002

[Dun03] Margaret H. Dunham. Data Mining Introductory and Advanced Topics. Prentice Hall, 2003.

[DZ03] W. Du and Z. Zhan. Using randomized response techniques for PPDM. In *Proceedings of the 9th ACM SIGKDD*, pages 505-510, Washington, DC, USA, August 2003..

[EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in PPDM. In *Proceeding of the ACM SIGMOD/POD Conference*, pages 211-222, San Diego, CA, June 2003.

[ESAG02] A. Evfimieski, R. Srikant, R. Agrawal and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the 8th ACM SIGKDD*, pages 217-228, Edmonton, Canada, July 2002.

[FW98] S. E. Fienberg and Leon C. R. J. Wallenberg. Introduction to the special issue: Disclosure limitation methods for protecting the confidentiality of statistical data. *Journal of Official Statistics*, pages 337-345, Vol. 14, No 4, 1998.

[Gal90] Stephen I. Gallant. Perceptron-based learning algorithms. *IEEE Transaction on Neural Networks*, pages 179-190, Vol. 1. No. 2, June 1990.

[GKW98] J. M. Gouweleeuw, P. Kooiman, L.C.R.J. Willenborg and P.-P. de Wolf. Post randomisation for statistical disclosure control: theory and implementation. *Journal of official Statistics*, pages 463-478, Vol.14 1998.

[GWL06] S.Guo, X.Wu and Y. Li. On the lower bound of reconstruction error for spectral filtering. *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD06)*, Berlin, Germany, Sept 18-22, 2006.

[HDC05] Z. Huang, W. Du and B. Chen. Deriving private information from randomized data. In *Proceeding of SIGMOD,* pages 37-48, June, 2005, Baltimore, Maryland, USA.

[HGC95] D. Heckerman, D. Geiger and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Journal of Machine Learning*, Vol. 20, pages 197–243, 1995.

[HH02] Van den Hout, A. and Van der Heijden, P.G.M. Randomized response, statistical disclosure control and misclassification: a review. *International Statistical Review* , Vol. 70, pages269-288, 2002.

[HH04] Van den Hout, A. and Van der Heijden, P.G.M. The analysis of multivariate misclassified data with special attention attention to randomized response data. *Sociological Methods and Research*, Vol. 32, 2004.

[Hin69] D. V. Hinkley. On the ratio of two correlated normal random variables. *Biometrika* , pages 635-639, Vol. 56, No. 3, 1969.

[HMC97] D. Heckerman, C. Meek and G. Cooper. A Bayesian approach to causal discovery. *Technical Report MSR-TR-97-05, Microsoft Research*, 1997.

[Hout99] Van den Hout, A. The analysis of data perturbed by PRAM. *WBBM Report Series* 45, Delft: Delft University Press, 1999.

[JW05] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593-599, 2005.

[KC04] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pages 1026-1037, Sept., 2004.

[KDWS03] H. Kargupta, S. Datta, Q. Wang and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the IEEE International Conference on Data Mining*, pages 99-106, Melbourne, FL. November 2003.

[KJC04] Murat Kantarciolu and Jiashun Jin and Chris Clifton. When do data mining results violate privacy?. In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 599-604, 2004.

[KPHJ00] H. Kargupta, B. Park, D. Hershberger and E. Johnson. Collective data mining: A new perspective toward distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 133–184, AAAI/ MIT Press, Menlo Park, California, USA, 2000.

[LB94] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, No.10, pages 262–293, 1994

[LKR06] K. Liu, H. Kargupta and J. Ryan. Random projection-based multiplicative perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 18, No. 1, pages 92–106, Piscataway, NJ, January 2006.

[LP00] Y. Lindell and B. Pinkas. Privacy-preserving data mining. In *Advances in Cryptology-CRYPTO*, pages 36-54, 2000.

[Lucy74] Lucy, L.B. An iterative technique for the rectication of observed distribution. *The Astronomical Journal* , Vol.79,1974.

[MDS05] J. Mateo-sanz , J. Domingo-ferrere and F. Sebe. Probabilistic information loss measures in confidentiality protection of continuous microdata. *Journal of Data Mining and Knowledge Discovery*, Vol. 11, pages 181-193, 2005.

[MG03] Srujana Merugu and Joydeep Ghosh. Privacy-preserving distributed clustering using generative models. In *Proceeding of the 3rd IEEE International Conference on Data Mining*, pages 211-218, Melbourne, Florida, USA, November 2003.

[Mit97] T. Mitchell. Machine Learning. McGraw-Hill, 1997.

[MS05] J. Ma and K. Sivakumar. Privacy- preserving Bayesian network parameter learning. *4th WSEAS International Conference on Computational Intelligence, Man-machine Systems and Cybernetics*, Miami, Florida, November, 2005.

[MS06a] J. Ma and K. Sivakumar. A PRAM framework for privacy-preserving Bayesian network parameter learning. *WSEAS Transactions on Information Science and Applications*, Vol. 3, No. 1, January 2006.

[MS06b] J. Ma and K.Sivakumar. Privacy-preserving Bayesian network learning from heterogeneous data. In *Proceeding of the 2006 International Conference on Data Mining*, Las Vegas, USA., pages 246-252, June, 2006.

[MS06c] J. Ma and K. Sivakumar. Learning from perturbed data for PPDM. to be submitted.

[MSK04] D. Meng, K. Sivakumar and H. Kargupta. Privacy-sensitive Bayesian network parameter learning. In *the Fourth IEEE International Conference on Data Mining*, pages 487-490, Brighton, UK. November 2004.

[MurS03] K. Muralidhar and R. Sarathy. A theoretical basis for perturbation methods. *Statistics and Computing,* Vol. 13, pages 329–335, 2003.

[Nea04] R.E. Neapolitan. Learning Bayesian Networks, Prentice Hall series in artificial intelligent, 2004

[OZ04] S.R.M. Oliveira and O.R. Zaiane. Toward standardization in PPDM. In *ACM SIGKDD 3rd Workshop on Data Mining Standards*, pages 7–17, 2004.

[PB95] F. J. Provost an d B. Buchanan. Inductive policy: The pragmatics of bias selection. *Machine Learning*, Vol. 20, pages 35–61, 1995.

[Qui86] J. Ross Quinlan. Induction of decision trees. Machine Learning, Vol. 1, pages 81–106, 1986.

[Qui93] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[RH02] S. Rizi and J. R . Haritsa. Maintaining data privacy in association rule mining. In *the proceedings of the 28th VLDB Conference* , pages 682-693, Hongkong, China, 2002.

[Swe02a] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 10, No. 5, pages 557-570, 2002.

[Swe02b] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, Vol. 10 , No. 5, pages 571-588, 2002.

[TG00a] K. Tumer and J. Ghosh. Robust order statistics based ensemble for distributed data mining. *In Advances in Distributed and Parallel Knowledge Discovery*, pages 185–210. MIT Press, 2000.

[TK03] S. Theodoridis and K. Koutroumbas, Pattern Recognition Second Edition, 2003

[UCI] UCI Data Repository URL: http://kdd.ics.uci.edu/

[VBFP04] V.S. Verykios, E. Bertino, I.N. Fovino, L.P. Provenza, Y. Saygin and Y. Theodoridis. State-of-the-Art in PPDM. *ACM SIGMOD Record*, Vol. 3, No. 1, pages 50-57, Mar. 2004.

[VC02] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639-644, 2002.

[VC03] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206-215, 2003.

[VEBS04] Vassilios S. Verykios, Ahmed K. Elmagarmid, Elisa Bertino, Yucel Saygin and Elena Dasseni. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering* , Vol. 16, No. 4, pages 434-447, 2004

[WGK97] De Wolf, P.-P., Gouweleeuw, J.M., Kooiman, P., and Willenborg, L.C.R.J. Reflections on PRAM. Research paper no. 9742, Voorburg/ Heerlen: Statistics Netherlands.

[Wu03] C. W. Wu. PPDM: A signal processing perspective and a simple data perturbation protocol. *2nd Workshop on PPDM, IEEE International Conference on Data Mining*, Melbourne, FL, 2003.

[WY04] R. Wright and Z. Yang. Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining,* pages 713-718, 2004.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 162-167, 1986.

[YW05] Z. Yang and R. Wright. Improved privacy-preserving Bayesian network parameter learning on vertically partitioned data. In *Proceedings of the International Workshop on Privacy Data Management*, pages 1196-1206, 2005.

[YZW05] Z. Yang, S. Zhong, R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceeding of the 2005 SIAM International Conference on Data Mining (SDM),* 2005.