2008

MATRIX DECOMPOSITION FOR DATA DISCLOSURE CONTROL AND DATA MINING APPLICATIONS

Jie Wang
University of Kentucky

ABSTRACT OF DISSERTATION

Jie Wang

The Graduate School

University of Kentucky

2008

MATRIX DECOMPOSITION FOR DATA DISCLOSURE CONTROL AND DATA MINING APPLICATIONS

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By

Jie Wang

Lexington, Kentucky

Director: Dr. Jun Zhang, Professor of Computer Science

Lexington, Kentucky

2008

ABSTRACT OF DISSERTATION

MATRIX DECOMPOSITION FOR DATA DISCLOSURE CONTROL AND DATA MINING APPLICATIONS

Access to huge amounts of various data with private information brings out a dual demand for preservation of data privacy and correctness of knowledge discovery, which are two apparently contradictory tasks. Low-rank approximations generated by matrix decompositions are a fundamental element in this dissertation for the privacy preserving data mining (PPDM) applications. Two categories of PPDM are studied: data value hiding (DVH) and data pattern hiding (DPH). A matrix-decomposition-based framework is designed to incorporate matrix decomposition techniques into data preprocessing to distort original data sets. With respect to the challenge in the DVH, how to protect sensitive/confidential attribute values without jeopardizing underlying data patterns, we propose singular value decomposition (SVD)-based and nonnegative matrix factorization (NMF)-based models. Some discussion on data distortion and data utility metrics is presented. Our experimental results on benchmark data sets demonstrate that our proposed models have potential for outperforming standard data perturbation models regarding the balance between data privacy and data utility.

Based on an equivalence between the NMF and $\mathcal{K}$-means clustering, a simultaneous data value and pattern hiding strategy is developed for data mining activities using $\mathcal{K}$-means clustering. Three schemes are designed to make a slight alteration on submatrices such that user-specified cluster properties of data subjects are hidden. Performance evaluation demonstrates the efficacy of the proposed strategy since some optimal solutions can be computed with zero side effects on nonconfidential memberships. Accordingly, the protection of privacy is simplified by one modified data set with enhanced performance by this dual privacy protection.

In addition, an improved incremental SVD-updating algorithm is applied to speed up the real-time performance of the SVD-based model for frequent data updates. The performance and effectiveness of the improved algorithm have been examined on synthetic and real data sets. Experimental results indicate that the introduction of the incremental matrix decomposition produces a significant speedup. It also provides potential support for the use of the SVD technique in the On-Line Analytical Processing for business data analysis.

KEYWORDS: Matrix Decomposition, Privacy, Security, Data Modification, Data Mining.

Jie Wang

Student's Signature

2008/12/05

Date

MATRIX DECOMPOSITION FOR DATA DISCLOSURE CONTROL
AND DATA MINING APPLICATIONS

By

Jie Wang

Director of Dissertation
Dr. Jun Zhang

Director of Graduate Studies
Dr. Andrew Klapper

RULES FOR THE USE OF DISSERTATIONS

Name	Date

DISSERTATION

Jie Wang

The Graduate School

University of Kentucky

2008

MATRIX DECOMPOSITION FOR DATA DISCLOSURE CONTROL AND DATA MINING APPLICATIONS

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By

Jie Wang

Lexington, Kentucky

Director: Dr. Jun Zhang, Professor of Computer Science

Lexington, Kentucky

2008

ACKNOWLEDGEMENTS

The work with this dissertation has been extensive and trying, but in the first place it is exciting, instructive, and fun. There have been many helping hands in the development of this dissertation. Without them, I would not even be close to being done. Here is my pleasure to express my gratitude to all of them.

First of all, I would like to acknowledge the overwhelming contribution and hours from my supervisor, Professor Jun Zhang, for his inspiring and encouraging way to guide me to a deeper understanding of knowledge, and his invaluable comments during the whole work with this dissertation.

I would like to thank the rest of my Advisory Committee: Dr. Jerzy W.Jaromczyk, Dr. Zongming Fei, and Dr. Sen-ching Samson Cheung who always give insightful comments and useful suggestions on my work. I would also like to thank the outside examiner Dr. Richard Charnigo for his helpful comments on my dissertation.

Besides, a special vote of thanks goes to Dr. Jerzy W.Jaromczyk and Dr. Grzegorz W. Wasilkowski, for their intelligence, friendliness, and sense of humor. Especially, I would like to express my great gratitude on their great help on all the international students in the Department.

Thanks also to all the friendly members in our lab who made the lab a great place to work. Let me say “thank you” to the following people: Dr. Wensheng Shen, Ms. Eun-Joo Lee, Dr. Ning Kang, Mr. Dianwei Han, Mr. Xuwei Liang, Mr. Changjiang Zhang, Mr. Ning Cao, Mr. Hao Ji, Mr. Yin Wang, Mr. Zhenmin Lin, Mr. Lian Liu and all the other group members once in our lab. Working together with all of you has been not only unforgettable experience, but a great pleasure as well.

Last, but not least, for my parents and my brother, no words can suffice for my gratitude on all unconditionally support and encouragement they have given me throughout my whole life. I dedicate this dissertation to them.

The research work with this dissertation was supported in part by:

- U.S. National Science Foundation (NSF) under grants ACR-0202934, ACR-0234270.
- Kentucky New Economy Safety and Security Initiative (NESSI) Consortium.
- Kentucky Science and Engineering Foundation.

Table of Contents

List of Tables

List of Figures

List of Files

Chapter 1

Introduction

A classification of data use can be made on the basis of five aspects: data distribution, data modification, data mining algorithm, data or rule hiding, and privacy preservation [76]. *Data mining* is the principle of sorting through large amounts of data and picking out relevant useful information. It is usually used by business intelligence organizations, and financial analysts, but it is increasingly used in sciences to extract information from the enormous data sets generated by modern experimental and observational methods. It has been described as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [31] and "the science of extracting useful information from large data sets or databases" [35].

The last aspect, *privacy preservation*, is becoming increasingly critical for future development of data mining techniques with greater potential access to datasets containing personal, sensitive, or confidential information. Extracting valid data mining results while preserving privacy of certain data sets is a major challenge for existing data mining algorithms.

Traditionally, data mining techniques have been considered as a useful tool in commercial, industrial and government business for various purposes, ranging from increasing profitability to enhancing national security. For example, inter-organizational collaboration significantly improves supply chains and enables more rapid and less costly transactions among partners. Data mining techniques can be utilized to discover valuable knowledge in private or shared public data. Given the large collections of person-specific information, service providers can mine data to learn patterns, models and trends that can be used to provide more effective personalized services. It can be used to do purchase recommendations on what product to buy, do text or document searching, assist in diagnosis of diseases and so on.

The potential benefits of data mining are certainly substantial, but the collection and analysis of sensitive personal data or secure data leads to concerns about individual privacy, data security and intellectual property rights. For example, National Security Agency (NSA) has a huge amount of databases on Americans' phone calls, using the data provided by ATT, Verizon and Bellsouth. The spying agency is using the data to analyze the call patterns in order to detect terrorist activities. In 2002, concerns over government collection of data led to street protests in Japan [75]. In 2003, concerns over the US Total Information Awareness program (TIA) even led to the introduction of a bill in the US Senate that would have stopped any US Department of Defense data mining program [75]. Such public reactions show a lack of understanding of data mining from the general public.

This misunderstanding creates several obstacles on the smooth development of data mining techniques and their applications. Among them is that the applicability of data mining techniques is problematic without an acceptable level of privacy of sensitive information [86, 49]. Furthermore, the quality of collected data may be questionable under the public concerns on privacy. In [23], it was shown that 73% of the respondents in a survey were not willing to provide their personal data without the protection of privacy.

Therefore, in recent years, data mining has been viewed as a threat to privacy by some people. There seem to be a pair of contradictory concepts. If we emphasize the data privacy, it may reduce the benefits of data mining; if we focus on knowledge discovery, there may be no guarantee on the data privacy. Privacy aspects of data mining have an important impact on many data analysis applications. In particular, due to the growth of electronic services, privacy protection has attracted a lot of attention recently. In these electronic services, privacy issues arise because many users have concerns about how and where their personal data and information will be used. Even though many nations have developed privacy protection laws and regulations to guard against improper use of personal information, the existing laws and their conceptual foundations have become outdated because of the rapid changes in data collection and data analysis technologies.

1.1 Privacy-Preserving Data Mining

Let us take a look at the sources of the possible threats on data privacy. It was reported in Wall Street Journal in February 2006 that companies were finding that insiders pose as great a risk to computer security as outside attackers [72]. For the attacks from outside the companies, access control mechanism can be used to assign different levels of rights to different users in order to control data disclosure. For the public access, only the non-confidential part of the date is published to the partners or the public. However, when the threat comes from inside the companies, the problem becomes more complicated. In order to use data analysis tools including data mining methods, some access rights have to be given to some employees for conducting analysis of the data. At this stage, the data privacy would be out of control without any data preprocessing. Thus, in the absence of adequate safeguards, the use of data mining can jeopardize the privacy and autonomy of individuals. Obtaining the potential benefits of data mining with privacy-aware technologies can enable a wider social acceptance of a multitude of new services and applications based on knowledge discovery.

A practical requirement from the above described privacy concerns is a trade-off between sharing confidential information for analysis and keeping individual, corporate and national privacy. For this requirement, organizations and enterprises must fulfill two seemingly contradictory missions. One is to share data or information within the companies, or with other partners or the public. The other is to protect confidential data and privacy of the data subjects.

This challenge, between data sharing and privacy preserving, has captured the attention of many researchers and administrators from many different communities, and motivated a great amount of research aimed to answer the questions such as: How can data be exchanged securely for cooperative analysis or outsourcing analysis? How can important structure and underlying patterns be found within a large data set without jeopardizing privacy? How and when can hidden structure be extracted from missing data or transformed data that is imprecise or partially incorrect?

By incorporating privacy protection mechanism, algorithms can be developed to hide sensitive data before executing data mining algorithms so that data mining activities will not breach privacy. As a result, the increasing concerns on privacy and related research brings out a new branch in data mining, known as **privacy preserving data mining (PPDM)**. Since the primary task in data mining is the development of models for decision making, developing accurate models without access to precise information in the original data is a natural objective for PPDM.

With the consideration on a number of different methods of PPDM from different communities, PPDM can be categorized from different view points. For example, they can be divided by different data set types (numerical-valued data vs. categorical-valued data or mixed-type data), or data location (centralized data vs. distributed data), or data mining methods (classification, clustering, association rule mining and so on). In our work in the dissertation, data disclosure control is emphasized, and "data" here is understood as an abstract word for a combination of "attribute" value in the "data" and "knowledge underlying data". In the dissertation, the following two categories are used to describe the characteristics of our privacy-preserving methods:

- **Data Value Hiding (DVH):** Data value hiding is to protect sensitive data values but maintain data patterns in order to prevent improper use of data. A graphical representation is shown in Figure 5.6. Our goal is to maximize the difference between an original data set A and its modified data set $\tilde{A}$ and minimize the different between the data mining results on A and $\tilde{A}$. The classical purpose of PPDM belongs to the category of DVH, where attribute values are typically modified so that disclosure risk of sensitive/ confidential attributes is minimized and the associated negative impact of data modification on data mining results is minimized [24, 9, 20, 77]. Consider a dataset T of customer profile having attributes of {`name, sex, birth date, city, purchased items, purchase values, salary`}. {`name`} is a direct identifier of the individual and {`salary, purchased items`} are sensitive variables containing sensitive information of the individual. The subset of {`sex, birth date, city`} can provide inference on individual identification. If assuming that the release of the entire T is required for some purpose; no access to sensitive variables is allowed; no inference on identification is allowed; and users are allowed to perform various data mining tools over a released data version $\widetilde{T}$, such as frequent items mining, regression, classification and clustering, then data modification (perturbation) is a commonly recommended practice to compute $\widetilde{T}$ [21]. A large amount of existing PPDM methods, roughly over 90%, fall into this category. A crucial problem in data value hiding is a trade-off between ***data privacy*** and ***data utility/information loss***. Data utility is that data patterns are maintained so that the mining accuracy is kept at a satisfied level on the modified data set. Since modification on data values is supposed to degrade data mining accuracy, how to achieve a balance between these two contradictory ends is a primary goal for this research line.

- **Data Pattern Hiding (DPH):** In many cases, the results of data mining activities can compromise the privacy too. The second category of PPDM, data pattern hiding, is another security concern growing out of the context of collaboration where sharing data is required among partners. It draws attention to disguise of confidential knowledge hidden in databases. For individual members in a collaborative project, preventing other partners from discovering some business-sensitive knowledge is vi-

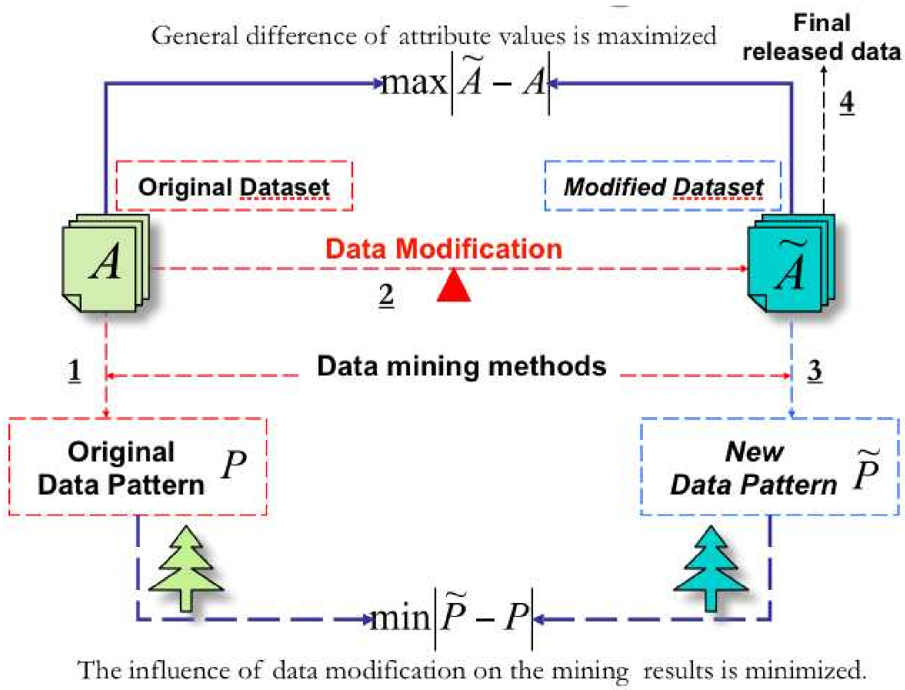


Figure 1.1: Data value hiding

tal when competitors or partners can use data mining algorithms to extract valuable (but potentially damaging to the data owners) knowledge from the shared data. It was indicated as another threat to database security by O'Leary in [62] and later by Clifton and Marks in [20]. A well designed scenario is provided in [20] and Verykios *et al.* analyzed it to indicate the need not only to hide data attribute values, but also to prevent data mining techniques from discovering sensitive knowledge [78].

To make an analysis of the assertion that the data mining technology has potential to jeopardize the profit of data owners, an example is illustrated in Figure 1.2. Assume that Alice and Bob are two manufacturers of the same products. Alice builds her customer profile database with the same structure as T described above. Due to some negotiation between Alice and Bob, Alice grants Bob the right of access to T. Bob carries out some data clustering technique to group the existing customers of Alice into two clusters: high potential valued customers and low potential valued customers; or a ranking algorithm is performed and a ranking of customer value is generated. In either case, Bob can take advantage of the outcome of data mining and design a marketing strategy to win over the customers having high possibility of future purchasing behavior. Probably, Alice will lose her customers and her business as well. In that case, it is highly recommended that Alice modify the original T before its release so that Bob has little chance of discovering the valuable customers.

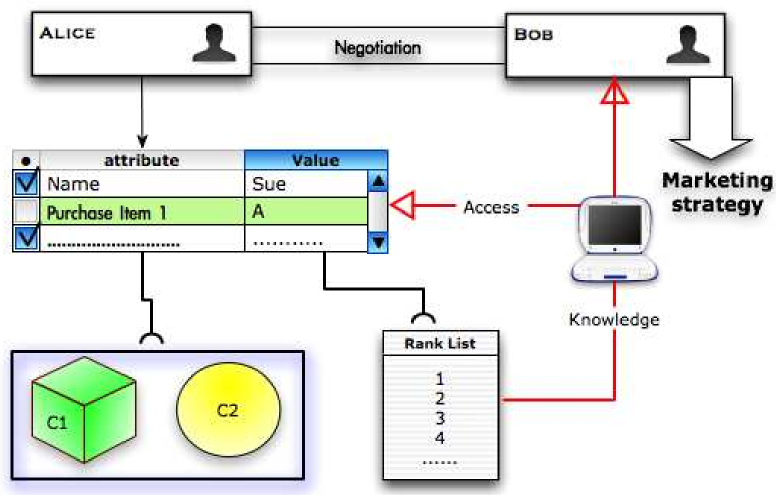


Figure 1.2: An example of data pattern hiding.

However, there are quite few published research works in this line, the existing research works are mainly limited to protect sensitive association rules in connection with some association rule mining algorithms [10, 25, 78, 84].

1.2 General Survey of Privacy-Preserving Data Mining

In this section, we will provide a general survey of PPDM models and methods, which are grouped into two categories as defined previously: data value hiding and data pattern hiding.

1.2.1 Current Status of Data Value Hiding Techniques

Before we start explaining the techniques that we have developed, let us take a look at what has been done in the field of PPDM, and what techniques are available. A number of techniques such as randomization and k-anonymity have been proposed in recent years in order to perform privacy-preserving data mining. Furthermore, the problem has been discussed in multiple communities such as the database community, the statistical disclosure control community and the cryptography community. In some cases, some different methods from different communities are quite similar, and there does not seem to be sufficient information exchange between these communities.

Bertino *et al.* [14] defined privacy in the context of data mining as the right of an entity to be secure from unauthorized disclosure of sensitive information about oneself that is contained in an electronic repository or that can be derived as aggregate and complex information from data stored in an electronic repository.

Intuitively there are three approaches to hiding sensitive data values. One is to transform the original data into protected, publishable data by using data perturbation. An alternative to data perturbation is to generate a new dataset (synthetic dataset), not from the original data, but from random values that are adjusted in order to have the same feature patterns as the original data. A third possibility is to build a hybrid dataset as a mixture of a distorted one and a synthetic one [39]. Most methods in literature for hiding sensitive data are based on element-wise random perturbation.

Most privacy control methods are developed specifically to target one of the following data types: statistical data/microdata, biological data/microarray, quantitative data, ordinal data, nominal data and categorical data. Statistical disclosure control (SDC) may be one of the earliest fields in data privacy preservation. Several reconstruction-based or randomization-based methods adding some noise to the original data have been widely used for privacy protection [30, 58]. Random projection approaches, most of which are multiplicative perturbations in the context of computing inner product matrix, have also been studied. The more recent approach in data distortion is based on the data matrix decomposition strategies [88].

In addition to these methods based on distorting the original data values, Clifton *et al.* proposed another class of approaches to modify data mining algorithms so that they allow data mining operations on distributed datasets without knowing the exact values of the data or without directly accessing the original data [19].

A condensation approach aiming at general cases was proposed in [3] to preserve data correlation that is the basis of many data mining algorithms like decision trees. However, data reduction or multiparty computations are not considered. It is more concerned with hiding the identities of objects.

Statistical Disclosure Control (SDC)

Statistical database (SDB) system is a database system that enables its users to retrieve only aggregate statistics for a subset of the entities represented in the database. The topic of PPDM has often been studied extensively by the data mining community without sufficient attention to the conventional work done by the statistical disclosure control community. Some work has been presented in parallel with similar work done in the area of database and data mining, such as k-anonymity, swapping, randomization, micro-aggregation and synthetic data generation.

The problem of protecting sensitive information in a database while allowing statistical queries has been studied extensively since the late 1970's [5, 69]. Early in 1989, Adam and Wortmann [5] conducted a comprehensive survey on security-control methods for statistical disclosure control. The methods are classified under four general approaches: conceptual, query restriction, data perturbation, and output perturbation. The survey introduced probability-distribution perturbation and fixed-data perturbation approaches.

Within the probability-distribution approach, Reiss [67, 68] suggested approximate data swapping to deal with multicategorical attributes. The original database is replaced with a randomly generated database having approximately the same t-order statistics as the original database. Liew *et al.* [50] proposed data distortion by probability distribution in 1985. Its operating principle is to obtain a protected dataset by randomly drawing from the underlying distribution of the original dataset. For fixed-data perturbation approach, Traub *et al.* [74] developed an additive-perturbation method for numerical attributes by adding or multiplying a random variable to a true value. It might be the first randomization scheme in privacy protection. The randomization for PPDM proposed by Agrawal and Srikant [9] in 2000 is the same as that by Traub *et al.* [74] and Abul-Ela *et al.* [3]; these proposals reduced multiple-value categorical attributes to two values, which results in a considerable information loss.

Data Perturbation

Data perturbation techniques are one of the most popular models. Before data owners publish their data, they modify the data in a statistical way to disguise confidential information by adding random noise to numerical attributes.

A large fraction of them use randomized data distortion techniques to mask the data by randomly modifying the data values. The simplest version is noise-additive approach [45, 7, 16, 30].

Noise-Additive Model: The modification is element-wise. The owner of a data set returns a value $a_i + v$, where a_i is the original data and v is a random value drawn from a certain distribution. The most commonly used distributions are the uniform distribution over an interval $[-\alpha, \alpha]$ and Gaussian distribution with the mean $\mu = 0$ and standard deviation σ. The n original data values $a_1, a_2, \ldots, a_n$ are realizations of n independent and identically distributed $(i.i.d)$ random variables. n independent noises, $v_1, v_2, \ldots, v_n$, are drawn from a distribution. Using the matrix format,this process can be written as

$$\widetilde{A} = A + V, \tag{1.1}$$

where A is the original data matrix, $\widetilde{A}$ is the perturbed data matrix and V is the noise matrix. The approach is intuitive and easy to understand. However, this model does not preserve Euclidean distances between the subjects. They are not suitable for widely used distance-based data mining algorithms such as the k-means clustering and the k-nearest neighbor classification. Furthermore, the element-wise data perturbation techniques do not reduce the data rank.

Given an assumption that an attacker has prior knowledge on the zero mean, μ, and the variance of the added noise, σ^2, it has been recently claimed that this model has privacy breaches, and some privacy intrusion techniques can be used to reconstruct private data from the randomized data [44, 45, 89]. The spectral properties of randomized matrix could help the attacker separate noise, V, from the perturbed data, $\widetilde{A}$. In particular, a spectral filtering-based method is proposed based on random matrix theory to reconstruct the original data from the randomized data [44, 45]. Two other data reconstruction methods, Principal Component Analysis-based and Bayes Estimate-based, are proposed in [36] to restore the original data from the perturbed data. It is suggested that the amount of original information that can be revealed is related to data correlation, and the more the correlation of noises resembles that of the original data, the better privacy preservation can be achieved [36].

Random Projection / Matrix Multiplication Model: It directly uses the concept of random mapping, a dimensionality reduction method, to reduce the data dimensionality and preserve enough structure of the original data set. It is mostly multiplicative perturbation in the context of computing inner product matrix [43, 57]. This model is based on the Johnson Lindenstrauss Lemma [41], which places bounds on Euclidean distance distortion due to any dimensionality reduction transform. The lemma states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. It is proved in the Johnson Lindenstrauss Lemma that, for a set of points of size n in a p-dimensional Euclidean space, there exists a linear transformation of the data into a q-dimensional space,

$$q \geq \mathcal{O}(\frac{\log(n)}{\epsilon^2})$$

that preserves distances up to a factor $(1 + \epsilon)$, $\epsilon \in (0, 1)$ [41].

Let $R \in \mathbb{R}^{m \times k}$ be a matrix generated with entries randomly chosen from a given distribution $\mathcal{N}(0, \sigma_r)$ with zero mean, $\mu = 0$ and variance σ_r^2, across columns, we have

$$\widetilde{A} = AR \tag{1.2}$$

for right multiplication. For left multiplication, it becomes

$$\widetilde{A} = RA. \tag{1.3}$$

In the random projection-based method, let $k = m$, since the dimension size should be maintained.

If R is nonorthogonal, according to the Johnson Lindenstrauss Lemma [41], the Euclidean distance is approximated on expectations up to a constant factor, and the random projection methods may suffer from the loss of Euclidean distances due to the nonorthogonal matrix R. We denote this method as Arp and rpA. The computational complexity is due to a matrix multiplication and is of the order $\mathcal{O}(nmm)$, and if A is sparse with about c nonzero entries per row, the complexity is of the order $\mathcal{O}(cnm)$.

If R is orthogonal, then the projection exactly preserves the inner product of A, which is the squared Euclidean distance,

$$\widetilde{A}\widetilde{A}^T = ARR^TA^T = AA^T. \tag{1.4}$$

We denote this by $Arpo$ and $rpoA$. The complexity will be increased with the cost incurred by orthogonalizing R, which is in the order of $\mathcal{O}(n^3)$. $Arpo$ is of the order $\mathcal{O}(nm^2 + n^3)$ and is always computationally expensive.

It is claimed that because this model preserves Euclidean distance with either small or no error, it allows many important data mining algorithms to be applied to the perturbed data and produce results very similar to, or exactly the same as those produced by the original data, *e.g.*, k-means clustering, k-nearest neighbor classification, and hierarchical clustering [43].

However, the issue of how well A is hidden is not clear and deserves more study. It might not be able to provide enough privacy protection. Therefore, a balance between data privacy and data utility is not guaranteed with this model. Another issue is that orthogonalizing R is unfortunately computationally expensive.

As with the noise-additive model, several researchers have investigated the vulnerabilities of the random projection model using various forms of prior knowledge [17]. The assumptions on prior knowledge include that R is orthogonal, or some samples are known. The covariance matrix may also be used to estimate the original distribution.

Secure Multi-party Computation (SMC)

The Secure Multi-party Computation (SMC) approach considers the problem of evaluating a function of two or more parties' secret inputs, such that each party finally gets the designed function output and nothing else is revealed, except what is implied by the party's own inputs and outputs. Du *et al.* gave a comprehensive review on SMC [28]. SMC and cryptography techniques can be combined for distributed data mining. Without generality, numerous distributed algorithms are task-specific. These tasks include privacy preserving information retrieval, geometric computation, statistical analysis and scientific computations.

1.2.2 Current Status of Data Pattern Hiding

For association rule hiding, two approaches based on heuristic modification have been proposed to prevent association rules from being generated [25]. One is to hide the frequent sets from which rules are derived. The second is to reduce their importance by setting their confidence below a user-specified threshold. Verykios *et al.* [78] presented five algorithms to hide sensitive association rules by insertion or removal of records. Three of them belong to the first approach that decreases either the confidence or the support of a set of sensitive rules until the rules are hidden. The other two use the second approach to decrease the support of a set of large itemsets until they are below a user-specified threshold so that no rule can be derived from the selected itemsets. However, the approaches make a strong assumption of no overlapping, i.e., all the items in a sensitive rule do not appear in any other sensitive rule. Some undesirable side effects may not be avoided, such as lost rules (nonsensitive rules falsely hidden) and ghost rules (spurious rules falsely generated). In order to limit side effects, Wu *et al.* [84] proposed heuristic methods for increasing the number of hidden sensitive rules and reducing the number of modified entries. Atallah *et al.* [10] used an itemset graph to hide sensitive itemsets referred to as data sanitization.

For classification rule hiding, a reconstruction-based framework for categorical datasets is proposed by Natwichai *et al.* [59, 60]. After extracting sensitive rules, a new decision tree is built on nonsensitive subset of rules. A new dataset is generated from the decision tree. It is claimed that even though the difference in representation between the new and original datasets can be found, the approach can maintain high level data usability.

1.3 Applications of Privacy-Preserving Data Mining

The problem of privacy-preserving data mining has numerous applications in homeland security, medical database mining, bio-terrorism and customer transaction analysis [8].

- **Homeland Security Applications:** A number of applications for homeland security are inherently intrusive because of the very nature of surveillance. In [71], a broad overview is provided on how privacy-preserving techniques may be used in order to deploy these applications effectively without violating user privacy. Some examples of such applications are as follows:

    1. **Credential Validation Problem:** This is to make a match between the subject of credential and the person presenting the credential. For example, the theft of social security number presents a serious threat to homeland security. The credential validation approach tries to exploit the semantics associated with the

social security number to determine whether the person presenting the social security number credential truly owns it.

2. **Web Camera Surveillance:** Web camera is widely used for surveillance to detect unusual activities. It has been hypothesized in [71] that unusual activities can be detected only in terms of facial count rather than using more specific information about particular individuals.

- **Video Surveillance.** There has been a tremendous proliferation of video surveillance cameras in public locations such as stores, ATMs, schools, subway stations, and airports. When sharing video-surveillance data, facial recognition software can match the facial images in videos to the facial images in a driver license database. If each face is blacked out, then all facial information will be wiped out. In [61], selective downgrading is used on facial information in order to limit the ability of facial recognition software to reliably identify faces, while maintaining facial details in images. k-Same algorithm is designed for this purpose [61]. The idea is to create new synthesized data by identifying faces which are somewhat similar, and then to construct new faces which generate combinations of features from these similar faces. Thus, the identity of the underlying individuals is protected to a certain extent, but the video continues to be useful.

- **Genomic Privacy.** DNA data is considered extremely sensitive since it contains almost uniquely identifying information about an individual. As in the case of multi-dimensional data, simple removal of directly identifying data such as social security number is not sufficient to prevent re-identification. A software *CleanGene* can determine the identifiability of DNA entries independent of any other demographic or identifiable information [55]. The software relies on publicly available medical data and knowledge of particular diseases in order to assign identifications to DNA entries. In [55], it was shown that $98-100\%$ of the individuals are identifiable using the approach. The identification is done by taking the DNA sequence of an individual and then constructing a genetic profile corresponding to the sex, genetic diseases, the location where the DNA was collected. One way to protect the anonymity of the sequence is with the use of *generalization lattices* which are constructed in such a way that an entry in the modified database cannot be distinguished from at least $(k-1)$ other entries.

1.4 Data Privacy and Data Perturbation

1.4.1 High-Accuracy Data Hiding

As seen from the previous section, most privacy-preserving methods apply a transformation which reduces the effectiveness of the underlying data when the data mining methods or algorithms are applied to the transformed data. The process of privacy-preservation may lead to loss of input information for data mining purposes. This loss of input information can also be considered as loss of utility for the data mining purposes. Considering the numerical data sets, we found that noise-additive model is easy to implement by two steps, random matrix generation and matrix addition operation. The complexity is of order $\mathcal{O}(nm)$. Its disadvantage is that the addition of external noise leads to the information loss and it might significantly degrade the data mining results. For the random projection model, it is claimed that it can perform quite well for Euclidean distance-based data mining algorithms. But for other kinds of data mining algorithms, there is no work to show its accuracy-maintenance. Also its complexity is much higher than the noise-additive model.

The problem of utility-based privacy-preserving data mining was first studied formally in [46] for the method of k-anonymous on categorical data sets. In fact, there is a natural tradeoff between privacy and data mining accuracy, though this tradeoff is affected by the particular algorithm which is used for privacy-preservation. A key issue in PPDM is to maintain maximum utility of the data without compromising the underlying privacy constraints.

Therefore, we intend to design a new privacy preservation model for numerical data sets, which can achieve a better balance between data value protection and data pattern maintenance (i.e., data mining accuracy). Here we call it *high-accuracy data hiding*.

The real-world data sets are unavoidably perturbed by the noises from different sources. It is generally acknowledged that most of the information gathering devices or methods at present have only finite bandwidth. One thus cannot avoid the fact that the data collected often are not exact. For example, signals received by antenna arrays often are contaminated by instrumental noises; astronomical images acquired by telescopes often are blurred by atmospheric turbulence; database prepared by document indexing often are biased by subjective judgment; and even empirical data obtained in laboratories often do not satisfy intrinsic physical constraints. Furthermore, in many situations the data observed from complex phenomena represent the integrated result of several interrelated variables acting together. When these variables are less precisely defined, the actual information contained in the original data matrix might be overlapping, fuzzy and no longer clear cut. Assume that the original data set, A, is the result from an unknown function f of the inherent data, $\mathring{A}$, and the inherent noise, $\dot{N}$, as

$$A = f(\mathring{A}, \dot{N}). \tag{1.5}$$

An implementation of the PPDM model, a modification strategy $\mathcal{L}$, is applied on A

$$\widetilde{A} = \mathcal{L}(A); \tag{1.6}$$

so that our idea can be roughly described in terms of the following three characteristics:

1. $\mathcal{L}$ should be able to modify the original data value so that the difference, $\|A - \widetilde{A}\|$, is significant, and the original data values are protected at a sufficient level.

2. At the same time, $\mathcal{L}$ should remove the inherent noise data $\dot{N}$ from the original data A so that the data utility is improved, sometimes, and it even produces a better data mining accuracy because of the removal of the inherent noise.

3. $\mathcal{L}$ should have a reasonable computational complexity for high-dimensional data sets.

Thus, a low-rank approximation is one of the candidates for $\widetilde{A}$ in order to realize the above characteristics of the model. Matrix decomposition or factorization techniques can reduce the data rank, *i.e.*, extract the significant variances from the data; and ignore the nonsignificant variances so that the inherent noise can be removed. These unique characteristics of the matrix decomposition forms a basis for the models we will describe in the chapters of §3, §4 and §5.

1.4.2 Dual Privacy Protection

Moreover, many research works are focused on either one of these two categories of PPDM. Since the mechanism of most PPDM algorithms is distortion or transformation of original datasets by different algorithms, it is common that the final version of the distorted datasets may not satisfy both data value hiding and data pattern hiding. This suggests that two different modified versions of the original dataset may be needed for these two disparate subtasks. To the best of our knowledge, there has been no effort made on achieving both data value hiding and data pattern hiding by using the same modified dataset.

We design a method which tends to preserve the privacy of some sensitive end results of the applications (here we call it *dual privacy protection*). In the §5 of the dissertation, a matrix decomposition-based method is designed to the application where k-means clustering is conducted on centralized numerical data sets. Next, a simple introduction of two matrix decompositions is given since they construct the core of our proposed model and methods.

1.4.3 Two Matrix Decomposition Techniques

Conventionally, matrix decomposition in numerical linear algebra is used as a computationally convenient means to obtain the solution to the original linear system or to understand certain properties of the matrix. Within the field of data mining, its major purpose is to obtain some form of simplified low-rank or low-dimensional approximations to original dataset for understanding the structure of data, particularly the relationship within the objects and within the attributes and how the objects relate to the attributes [37]. Low-rank factorization techniques not only enable users to work with reduced-rank models, they also often facilitate more efficient statistical classification, clustering and organization of data, and lead to faster searches and queries for patterns or trends.

Many of the existing data distortion methods inevitably fall into the context of matrix computation. For instance, having the longest history in privacy protection area and adding random noise to the data, additive noise method can be viewed as a random matrix method and therefore its properties may be understood by studying the properties of random matrices [54, 4].

Matrix decomposition renders a compact representation with reduced-rank while preserving dominant data patterns. These characteristics motivate us to utilize it to achieve the seemingly contradictory tasks: high data value protection and high data mining accuracy. The goals of this Ph.D. dissertation study are to use matrix decomposition techniques to achieve high-accuracy data disclosure control, and dual privacy protection. Singular value decomposition and nonnegative matrix factorization are two techniques used in the dissertation.

Singular Value Decomposition (SVD)

Based on eigenvalue and eigenvector analysis, singular value decomposition of a matrix is probably the most well-known member of the family of matrix decomposition methods. Given a matrix, $A \in \mathbb{R}^{n \times m}$, with rank r, the singular value decomposition, the SVD of A, is defined as

$$A = U \Sigma V^T, \tag{1.7}$$

where $U \in \mathbb{R}^{n \times n}$, Σ is a diagonal matrix of size $n \times m$, having only r nonzero entries (the singular values of A) as its diagonal entries in the descending order, and $V \in \mathbb{R}^{m \times m}$. U

and V consists of orthonormal eigenvectors associated with the r nonzero eigenvalues of AA^T and A^TA. Hence, the r columns of U corresponding to the nonzero singular values span the *column space*, and the r columns of V span the *row space* of the matrix A. U and V contain the *left* and the *right* singular vectors, respectively.

The popularity of the SVD covers a wide range of areas. In data mining, SVD inspires web search techniques such as the latent semantic indexing (LSI) technique for text mining to find similarities among documents or clustering documents [32]. In [32], Gao and Zhang proposed a sparsified SVD (SSVD) to reduce storage requirements in SVD based text mining applications.

Despite the large number of attributes, most datasets arising in practical application result in a representation having a good low-dimensional approximation. SVD is a popular method of dimension reduction in data mining and information retrieval [66], since it has a mathematical feature to find a rank-k approximation of a matrix with minimal change on its pattern to that matrix for a given value of k [29]. It is mainly used to reduce the dimensionality of the original dataset.

Its promise on the minimal change on data patterns makes it particularly interesting for our application. In §3, an SVD-based data hiding model is designed for numerical data sets. It is experimentally demonstrated that SVD is of great worth in constructing a decision model insensitive to distorted data values, therefore high accuracy data hiding can be achieved.

Nonnegative Matrix Factorization (NMF)

Another matrix decomposition we use in the dissertation is the Nonnegative Matrix Factorization (NMF). Given a nonnegative valued matrix, $A \in \mathbb{R}_+^{n\times m}$, there exists some $K \leq \min\{n, m\}$, *s.t.*

$$A \approx HW, \tag{1.8}$$

that minimizes the objective function $f(H, W) = \frac{1}{2}\|A - HW\|_F^2$ where $H \in \mathbb{R}_+^{n \times K}$ and $W \in \mathbb{R}_+^{K \times m}$. $\|A - HW\|_F^2$ is the Frobenius norm of $(A - HW)$.

The idea of positive matrix factorization is developed by Paatero, and later become popular in the computational science community [42]. Interest in positive matrix factorization increased when a fast algorithm for nonnegative matrix factorization, based on iterative updates, was developed by Lee and Seung [48] (refer to §4.2.1). They were able to show that it produced intuitively reasonable factorizations for a face recognition problem. They showed that NMF facilitates the analysis and classification of data from image or sensor articulation databases made up of images showing a composite object in many articulations, poses, or observation views. They also found NMF to be a useful tool in text data mining [64]. In the past few years, several papers have discussed NMF techniques and successful applications to various databases where the data values are nonnegative [27].

NMF has recently been shown to be a very useful technique in approximating high dimensional data where the data are comprised of nonnegative components [38, 34, 65, 53, 83, 85]. Xu *et al.* [85] demonstrated that NMF-based indexing outperforms traditional vector space approaches in information retrieval such as latent semantic indexing for document clustering on a few benchmark test collections.

1.4.4 Real-time Performance

Besides the efficiency and accuracy, a good data modification method should be practically robust for different data sources. Usually, it should be scalable to large size data and computationally applicable to high-dimensional data. Secondly, it should be adaptive to the external perturbations, including the addition of new data, removal of old data and so on. Considering that the data streaming is more and more popular in the network and online environment, it is desirable that a good PPDM method can be implemented in real time. How to improve the real-time performance of our proposed models with respect to these properties, is one of the topics in this dissertation.

Computational cost has not traditionally been emphasized in previous work on PPDM. The data source may change or new data elements may be added, like in situations of financial transaction streams and network activity streams. Another scenario is that the data source is in an online setting where data must be incorporated into the data value hiding model as it arrives. The data value hiding model is required to be updated in real-time. We know that matrix operations are the core of implementation in the random projection model and the matrix-decomposition-based model. Eventually, the computational performance of these two models are subjected to the size of the data sets. If the data is frequently updated with increasing size, the computation of new models at each time would incur a sizable delay. It is important to figure out how to adjust the models dynamically for a real-time response when dealing with changes to the data matrix.

Our proposed models consist of matrix computations primarily from the matrix computation community. For the SVD-based models, there are many SVD computation software packages available, such as *Lanczos SVD* in MATLAB. First we want to mention that the algorithm is extremely stable. However, fundamentally, computing a full SVD is an $\mathcal{O}(nmm)$ time problem. The SVD is usually computed by a batch $\mathcal{O}(nm^2 + n^2m + m^3)$ time algorithm, meaning that all the data must be processed at once, and computing the SVD of very large data sets is essentially infeasible. Therefore, in order to make our model scalable, for large sized data sets, we consider the SVD of the complete data as a SVD updating problem. An initial SVD of a selected basis from the original data is computed by the usual stable algorithm, then this original SVD is updated on adding new data subjects by an incremental SVD algorithm.

Therefore, we will explore the computational needs of PPDM algorithms so as to handle growth and change in data sources. The work in §6 focuses upon improving the real-time performance of the SVD-based data value hiding model on a frequently-updated data source.

1.5 The Contributions of the Dissertation

Our research work in this dissertation is focused on studying the privacy aspects of data mining and designing methods to protect privacy in the process of data mining. Our main attention is the use of matrix decomposition techniques in data distortion for data value hiding and data pattern hiding in databases. In terms of the contributions of the dissertation, our research work can be broadly divided into *four* parts.

1. For the first part (§2), the objective is to make an attempt on designing some quantitative metrics to measure the distortion level of PPDM models. Up to now, there is no commonly accepted and uniformly applied metric in the field of PPDM. It is not an easy task since the privacy or distortion is an abstract concept. For a clear description of our research work in this dissertation, we divide PPDM into two classes: data value hiding and data pattern hiding. With respect to the data value modification and the data pattern modification, two classes of metrics are designed with their efficacy experimentally examined in our work. We call this part of our work **data distortion measurement**.

2. For the second part (§3 and §4), our goal is to develop techniques to hide to the outside world sensitive data, and simultaneously preserve the underlying data patterns and semantics of a data set, so that a decision model on the distorted data can be constructed. This decision model should be equivalent to or even better than the model using the original data from the viewpoint of decision accuracy [79]. A desirable solution must consider not only privacy safeguards, but also accurate data mining results. We call this part of our work **high-accuracy data hiding**.

3. For the third part (§5), our goal is to simultaneously hide data values and user-specified confidential patterns without undesirable side effects on nonconfidential patterns. The difficulty of data security increases considerably if we aim to achieve the goal of sensitive attribute value hiding and confidential pattern hiding at the same

time in data mining applications. With carefully designed data distortion techniques, we can make sure that, with the distorted datasets, when using certain data mining tools, confidential patterns will be incorrectly extracted while nonconfidential patterns will be correctly extracted. We call this part of our work **dual privacy protection**.

4. For the fourth part (§6), we focus on solving computation cost problem of the SVD-based model in a dynamic environment. The computation cost is very expensive if the SVD of the data set is repeatedly computed on the full size of the data set. In order to improve performance of the SVD-based model in a situation with dynamical and frequent addition of new records, an SVD updating algorithm is designed based on an incremental algorithm in [87, 73]. We call this part of our work **model dynamics enhancement**.

Specifically, we have done the following studies in the course of this research work.

1. We defined two classes of evaluation measures for evaluating data distortion level. The class of data value distortion evaluation measures consists of five metrics, and the class of data pattern distortion measures includes five metrics.

2. We have designed matrix decomposition-based methods for data hiding in high-accuracy data disclosure control. Even though the application of matrix computation in data mining field is not a new concept, the use of such techniques in privacy-preserving data mining has just recently started.

3. We studied basic procedures for matrix decomposition-based PPDM methods. The basic idea is to generate a distorted low-rank version of an original dataset by conducting NMF or SVD or their variants. Truncation and sparsification strategies are designed to adjust the level of data distortion. Selective sparsified SVD can be used for distributed datasets or for reducing computation cost for centralized datasets [79].

4. We designed matrix decomposition-based PPDM methods for data distortion in high-accuracy data hiding.

    - Singular value decomposition-based data hiding model.

        – thin SVD-based data hiding method.

        – sparsified SVD-based data hiding method.

        – selective sparsified SVD-based data hiding method.

    - Nonnegative matrix factorization-based data hiding model.

5. We proposed a novel approach to achieving the goal of dual privacy protection with one single perturbed dataset. We demonstrated the equivalence between nonnegative matrix factorization and $\mathcal{K}$-means clustering technique. On the basis of this equivalence theorem, factor swapping schemes are designed for simultaneously hiding data values and data patterns in datasets. In addition, our experimental results demonstrate that, by imposing certain restrictions on the computation of the NMF iterations, it is possible to compute an optimal solution for a particular dataset with particular security requirements, in which the user-specified confidential memberships or relationships are hidden without undesirable alterations on nonconfidential memberships [80].

6. We examined the efficiency of all the proposed data hiding methods on synthetic and real-world data sets. This was achieved by comparing our techniques with similar techniques developed by other researchers, noise-additive methods and random projection methods. By the extensive experiments, some properties of noise-additive methods and random projection methods were found.

7. We improved the dynamics of the matrix-based data hiding models by introducing an SVD updating algorithm. This performance improvement makes the proposed data hiding model adaptable to the real time or online environment. At the same time, the

algorithm is able to reduce the computation cost of the SVD-based data hiding model for the large scale dynamically-updated data sets.

Chapter 2

Preliminaries

Our study will be focused on data distortion as a means to provide privacy protection for datasets. Our target dataset is defined as a centralized database that contains records with several numerical attributes from some continuous real domain and a single categorical attribute (class label).

We consider a dataset T consisting of n subjects or data points, each of which has m features/attributes. For supervised learning, class labels are assigned to subjects prior to data processing. For unsupervised learning, class labels are unknown. Unsupervised learning methods can be used to find the cluster property of the data with a prior assumption of the number of clusters k. T is partitioned into k subsets which are referred to as clusters or classes. Each subject is a member of a particular cluster or subset. We can define a binary relation R over the membership of the subjects in § 5.

This chapter describes basic concepts that will be used in the dissertation, including definitions, basic data preprocessing steps, distortion and accuracy metrics and four real data sets for our experiments.

2.1 Definitions

In order to make a clear and consistent representation, we give a few definitions related to our study.

Data Model T

Given a dataset T consisting of n independent subjects in an m-dimensional feature space, with each subject having m numerical features. If we denote the ith subject of T as T_i, then

1. $T = \{T_i\}_{i=1}^{n}$
2. $T_i = \{t_{i1}, t_{i2}, \ldots, t_{ij}, \ldots, t_{im}\}, 1 \leq i \leq n, 1 \leq j \leq m.$

Vector Space Data Model A

Given a data model T, which can be represented by a matrix A, $A \in \mathbb{R}^{n \times m}$, with the rows corresponding to the n subjects and the columns to the m features. If the ith row is denoted by A_i, then A_i represents T_i. The jth feature is represented by the jth column of A, denoted by $A_{.j}$.

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}, \text{ or } A = [\ A_{.1}\ A_{.2}\ \ldots\ A_{.m}\ ].$$

Data Cluster C

Given a dataset of size n from an m-dimensional feature space, $\{T_1, T_2, \ldots, T_n\}$, denoted by T, the number of clusters K and a learning algorithm I, C_1, C_2,…, C_K are K subsets, created by I; c_1, c_2,…, c_K are K cluster centroids, such that,

1. $T = \bigcup_{i=1}^{K} C_i$,
2. $|C_i| =$ the number of data subjects in C_i,
3. $c_i = \frac{1}{|C_i|}(\sum_{T_j \in C_i} T_j)$,
4. $\forall p, q \in \{1, 2, \ldots, K\}, C_p \cap C_q = \Phi, p \neq q$,
5. $\forall i, 1 \leq i \leq n, \exists p, 1 \leq p \leq K, T_i \in C_p$.

Data Modification

Given two datasets T and $\widetilde{T}$ with the matrix models of A and $\widetilde{A}$, and a modification scheme M, a sequence of modifications is a function Ψ to transform A into $\widetilde{A}$, where F indicates the subjects to be modified.

$$\Psi : (A, F, M) \longrightarrow \widetilde{A}.$$

Data Value Hiding (DVH)

Given a data model A, the subjects to be modified F and a learning algorithm I, a data distortion scheme M is selected to execute data modification and compute $\widetilde{A}$: $\Psi : (A, F, M) \rightarrow \widetilde{A}$. Two sets of learning results O and $\widetilde{O}$ are created by performing I on A and $\widetilde{A}$, respectively. F is considered to be hidden in $\widetilde{A}$ if the following conditions are satisfied:

1. In $\widetilde{A}$, disclosure of F is controlled without unauthorized access.
2. The difference of O and $\widetilde{O}$ is limited to a user-defined threshold level.

Data Pattern Hiding (DPH)

Given a data model A, user-defined confidential knowledge P and a learning algorithm I, a data distortion method is selected to execute data modification and compute $\widetilde{A}$: $\Psi : (A, F, M) \rightarrow \widetilde{A}$. Two sets of learning results O and $\widetilde{O}$ are created by performing I on A and $\widetilde{A}$, respectively. P will be considered to be hidden in $\widetilde{A}$ if the following conditions are satisfied:

1. $P \nsubseteq \widetilde{O}$;
2. $P \subseteq O$.

Pairwise Association R

Given a data set T, let T^2 denote $T \times T$, the set of all possible ordered pairs of elements of T, an association R is a binary function $\Psi : (T^2, I, C) \rightarrow \{true, false\}$. $\forall (x, y) \in T^2, \exists p, q, 1 \leq p, q \leq K, suchthat,\ T_x \in C_p, T_y \in C_q$,

1. $p = q \rightarrow xRy = true$,
2. $p \neq q \rightarrow xRy = false$.

Lemma 2.1.1. *R is an equivalence relation.*

Proof. First, R is reflexive as $\forall T_i \in T, T_iRT_i$. Second, it is symmetric, as $\forall i, j, 1 \leq i \leq n, 1 \leq j \leq n, T_iRT_j$ means that T_i and T_j are in the same cluster which implies T_jRT_i. Third, it is transitive, as whenever T_i is in the same cluster as T_j and T_j is the same cluster as T_t, then T_i is in the same cluster as T_t, therefore T_iRT_t. ■

Confidential Association Hiding

Let $\widetilde{T}$ be the dataset after applying a sequence of modifications on T and a pair $(x, y) \in T^2$. xRy will be hidden if the following conditions are satisfied:

1. $l = xRy$ in T,
2. $g = xRy$ in $\widetilde{T}$,
3. $g = \neg\, l$.

2.2 Data Preprocessing

2.2.1 Normalization

In the context of data mining, normalization refers to scaling the data to fall within a small, specified range, thus allowing underlying characteristics of the data sets to be compared. There are several different normalization techniques and the choice is problem-specific. Assuming that the data model A has the mean vector $\overrightarrow{\mu}$ and the standard deviation vector $\overrightarrow{\sigma}$, we have

$$\overrightarrow{\mu} = \frac{1}{n} Z_{1\times n} A. \tag{2.1}$$

And $\overrightarrow{\sigma}$ is the square root of an unbiased estimator of the variance of the distribution from which A is drawn, as long as A consists of independent, identically distributed subjects,

$$\overrightarrow{\sigma} = \left(\frac{\mathbf{diag}\left[(A - \frac{1}{n} Z_{n \times n} A)^T (A - \frac{1}{n} Z_{n \times n} A) \right]}{n-1} \right)^{\frac{1}{2}} \tag{2.2}$$

where **diag** is to form the diagonal elements of a matrix as a row vector, and Z is a vector or a matrix with all the elements being 1.

Centering. It is usual to center the data, *i.e.*, shifting the data and making its column means zero.

$$C = A - \frac{1}{n} Z_{n \times n} A. \tag{2.3}$$

Z-score normalization. It is also known as *data standardization*. The standardization of A is conducted on each attribute as

$$A_{.j} \longleftarrow \frac{A_{.j} \ominus \overrightarrow{\mu}(j)}{\overrightarrow{\sigma}(j)}, \tag{2.4}$$

where $\ominus$ is an element-wise operation.

Range adjustment. It is common that the attributes have different value ranges. We can normalize their value ranges to a unit range. Each attribute is normalized by its value range as

$$A_{.j} \longleftarrow (A_{.j} \ominus \min A_{.j}) \times \left(\frac{1}{\max A_{.j} - \min A_{.j}} \right) \tag{2.5}$$

Unit-length normalization. Each attribute column vector can be normalized to unit length as

$$A_{.j} \longleftarrow \frac{A_{.j}}{\|A_{.j}\|}, \tag{2.6}$$

where $\|A_{.j}\|$ is the length of $A_{.j}$, *i.e.*, the 2-norm of $A_{.j}$.

2.2.2 Whitening

Whitening is to remove correlation between attributes or components, which transforms A into D with mutually uncorrelated components,

$$D = CW \tag{2.7}$$

where C is defined in (2.3) and W is the whitening matrix of A. Usually W can be taken as $(E(C^TC))^{-\frac{1}{2}}$, where $E(C^TC)$ is the expected value of C^TC. After whitening, the covariance matrix $\hat{cov}(D) = I$. I is an identity matrix. Each column of D has the zero mean and unit variance. The covariance for any pair of columns is zero.

Proof.

$$E(D) = E(CW) = E(C(E(C^TC))^{-\frac{1}{2}}) = E(C)(E(C^TC))^{-\frac{1}{2}} = \mathbf{0}_{n\times m}.$$

$$\begin{aligned}
\hat{cov}(D) &= E(D^TD) - E(D)E(D^T) \\
&= E(D^TD) \\
&= E\left(((E(C^TC))^{-\frac{1}{2}})^T C^T C (E(C^TC))^{-\frac{1}{2}}\right) \\
&= ((E(C^TC))^{-\frac{1}{2}})^T E(C^TC)(E(C^TC))^{-\frac{1}{2}} \\
&= E(C^TC))^{-\frac{1}{2}} E(C^TC)(E(C^TC))^{-\frac{1}{2}} \\
&= I_{m\times m}.
\end{aligned}$$

■

2.3 Data Value Distortion Metrics

We need to define some metrics to evaluate our proposed data distortion methods. The evaluation will be on two aspects: data distortion and data utility. Our data value distortion measurement will be used to evaluate dissimilarity between the original and the distorted datasets. It should indicate how closely the original value of an item can be estimated from the distorted data. We will use a few data distortion metrics to assess the level of data distortion which only depends on the original matrix A and its distorted counterpart $\widetilde{A}$. Two kinds of metrics are designed for data value distortion and data pattern distortion, respectively.

This section will discuss data value distortion metrics. In §3, the usefulness of these metrics will be examined.

2.3.1 Relative Error (RE)

After a data matrix is distorted, the values of its elements change. The value difference of the datasets is represented by the relative value difference in the Frobenius norm. Thus RE is the ratio of the Frobenius norm of the difference of $\widetilde{A}$ from A to the Frobenius norm of A:

$$\mathsf{RE} = \frac{\| A - \widetilde{A} \|_F}{\| A \|_F}. \tag{2.8}$$

For example, for the following dataset A_e, its distorted data matrix $\widetilde{A_e}$ is obtained by applying the SVD method with $k = 2$. Then the RE value computed for this distortion is 0.1884. The level of distortion should be greater if the value of RE is increased.

$$A_e = \begin{bmatrix} 1 & 2.5 & 5 & 0.3 \\ 2 & 3.9 & 2 & 1.1 \\ 4 & 1.8 & 8 & 0.5 \\ 1 & 3.3 & 6 & 1.2 \end{bmatrix}, \quad \widetilde{A_e} = \begin{bmatrix} 1.8093 & 2.2060 & 4.7910 & 0.6064 \\ 1.2923 & 1.5757 & 3.4219 & 0.4331 \\ 2.8661 & 3.4947 & 7.5896 & 0.9606 \\ 2.2176 & 2.7040 & 5.8724 & 0.7433 \end{bmatrix}.$$

2.3.2 Rank Position (RP)

After a data distortion process, the ranks of the magnitudes of the data elements changes, too. We use several metrics to measure the rank difference of the data elements.

We use RP to denote the average change of rank for all the elements within their respective attributes. After the elements of an attribute are distorted, the rank of the magnitude of each element changes. Assume that the dataset A has n data objects and m attributes. $Rank_j^i$ denotes the rank in the ascending order of the jth element in the attribute i, and $Rank_j^{i*}$ denotes the rank in ascending order of the distorted element A_{ji}. If two elements have the same value, we define the element with the smaller row index to have the higher rank. Then RP is defined as:

$$\text{RP} = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{\sum_{j=1}^{n} |Rank_j^i - Rank_j^{i*}|}{n} \right). \tag{2.9}$$

In the dataset A_e, the rank vector for the $1st$ attribute can be represented as $Rank^1 = [2\ 3\ 4\ 1]^T$. After the distortion, $Rank^{1^*} = [2\ 1\ 4\ 3]^T$. The total change of rank for this attribute is 4. The average change of rank of the $1st$ attribute is 1. We can calculate the total change of rank for the other attributes and get $\text{RP} = 0.8760$.

2.3.3 Rank Maintenance (RK)

RK represents the percentage of elements that keep their ranks of magnitude in each column after the distortion. It is computed as:

$$\text{RK} = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{\sum_{j=1}^{n} Rk_j^i}{n} \right), \tag{2.10}$$

where Rk_j^i indicates whether an element keeps its rank during the data distortion process:

$$Rk_j^i = \begin{cases} 1, & \text{if } Rank_j^i = Rank_j^{i^*}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

For example, the rank vector of $2nd$ attribute in A_e is $[2\ 4\ 1\ 3]^T$. After the distortion, it is $[2\ 1\ 4\ 3]^T$ in $\widetilde{A_e}$. Thus all the elements in the $2nd$ attribute keep their original rank. RK for this example is 0.5625.

2.3.4 Attribute Rank Change (CP)

One may infer the content of an attribute from its relative value difference compared with the other attributes. Thus it is desirable that the rank of the average value of each attribute varies after the data distortion. Here we use the metric CP to define the change of rank of the average value of the attributes:

$$\text{CP} = \frac{\sum_{i=1}^{m} |RAV_i - RAV_i{}^*|}{m}, \tag{2.12}$$

where RAV_i is the rank in the ascending order of the average value of the ith attribute, while ${RAV_i}^*$ denotes its rank in the ascending order after the distortion. For instance, the rank vector of all attributes in matrix A_e is: $[4\ 1\ 2\ 3]^T$. The rank vector for the distorted matrix $\widetilde{A_e}$ is: $[4\ 1\ 2\ 3]^T$. Then the total change of rank is 0, so CP is equal to 0.

2.3.5 Attribute Rank Maintenance (CK)

Similarly to RK, we define CK to measure the percentage of the attributes that keep their ranks of average value after the distortion. So it is calculated as:

$$\mathsf{CK} = \frac{\sum_{i=1}^{m} Ck^i}{m}, \tag{2.13}$$

where Ck^i is computed as:

$$Ck^i = \begin{cases} 1, & \text{if } RAV_i = {RAV_i}^*, \\ 0, & \text{otherwise.} \end{cases} \tag{2.14}$$

In the previous example, CK$= 1$.

2.3.6 Summary

For any data modification method, the higher the value of RP and CP, and the lower the value of RK and CK, the more the original data matrix A is distorted, which implies that the data distortion method is better in preserving privacy.

For instance, we apply the SVD-based method with a different reduced rank, $k = 1$ on A_e in the previous example, a modified data matrix $\widetilde{A_e^*}$ is obtained as:

$$\widetilde{A_e^*} = \begin{bmatrix} 1.8093 & 2.2060 & 4.7910 & 0.6064 \\ 1.2923 & 1.5757 & 3.4219 & 0.4331 \\ 2.8661 & 3.4947 & 7.5896 & 0.9606 \\ 2.2176 & 2.7040 & 5.8724 & 0.7433 \end{bmatrix}.$$

The comparison of data value distortion metrics between $\widetilde{A_e}$ and $\widetilde{A_e^*}$ is shown in Table 2.1. $\widetilde{A_e^*}$ distorts the element values more than $\widetilde{A_e}$ since it has a greater RE value. It changes the magnitude rank of data elements more than $\widetilde{A_e}$ too, because of greater RP value and smaller RK value. The fact that CP$= 0$ and CK$= 1$ indicates that both of these two modified datasets do not change the attribute rank.

Table 2.1: Data value distortion metrics

Modified Dataset	RE	RP	RK	CP	CK
$\widetilde{A_e}$	0.1540	0.5000	0.5625	0	1
$\widetilde{A_e^*}$	0.2891	1.0000	0.4375	0	1

2.4 Data Pattern Distortion Metrics

Data quality is an old problem that was largely a scientific issue, with roots in measurement error and survey uncertainty. But for today's world of massive electronic data sets and difficult policy decisions, data quality problems can create significant economic and political inefficiencies. They should always be embedded in a decision-theoretic context [6]. We begin with a definition

> Data quality is the capability of data to be used effectively, economically and rapidly to inform and evaluate decisions. Necessarily, data quality is multi-dimensional, going beyond record-level accuracy to include such factors as accessibility, relevance, timelines, metadata, documentation, user capabilities and expectations, cost and context-specific domain knowledge [6].

In our study, data pattern distortion metrics indicate the accuracy of data mining algorithms possibly achieved on distorted data. Therefore, data quality, in the dissertation, is measured by the following defined data pattern metrics. In §3, the usefulness of these metrics is examined.

2.4.1 Subject Distance Distortion Metrics

In subject spaces, the similarity of subjects is measured by between-pair distances. For distance-based data mining algorithms, each object that is mapped to the same class may be thought of as more similar or closer to the objects in that class than to the objects in other classes. Distance measure is mostly used to identify the “alikeness” of different objects in the data sets. K-nearest neighbors (KNN) classification and $\mathcal{K}$-means clustering are two popular data mining algorithms based on distances. Therefore, their mining accuracy on the distorted datasets depends on the level of maintenance of *dissimilarity* or *distance* before and after the data distortion.

The Dissimilarity Matrix P**.** We define a symmetric matrix $P \in \mathbb{R}_+^{n\times n}$ as a dissimilarity matrix that stores a collection of pair-wise distances between every pair of subjects in a data set,

$$P = \begin{vmatrix} 0 & \dots & \dots & \dots & \dots \\ p(2,1) & 0 & \dots & \dots & \dots \\ p(3,1) & p(3,2) & 0 & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p(n,1) & p(n,2) & \dots & \dots & 0 \end{vmatrix}_{n\times n} \tag{2.15}$$

where the diagonal elements are self-distances and they are equal to zero. Each element $p(i,j)$ corresponds to the distance or dissimilarity between subjects i and j. In general, $p(i,j)$ is a nonnegative value that is close to zero when the subjects i and j are very similar to each other, and becomes larger the more they differ. We use the most popular distance measure, the Euclidean distance, to calculate P,

$$\begin{aligned} P_{ij} &= \|A_i - A_j\|_F \\ &= \left(\mathbf{tr}((A_i - A_j)^T(A_i - A_j))\right)^{1/2} \\ &= \begin{cases} 0 & \text{if } i = j, \\ \left(\sum_{s=1}^{m}(A_{is} - A_{js})^2\right)^{1/2} & \text{if } i \neq j. \end{cases}, \end{aligned} \tag{2.16}$$

where A_i and A_j are m-dimensional data subjects. Euclidean distance $p(i,j)$ satisfies the following constraints:

- $p(i,j) \geq 0$;
- $p(i,i) = 0$: the distance of an object to itself is zero;
- $p(i,j) = p(j,i)$: distance is a symmetric function;

- $p(i,j) \leq p(i,k) + p(k,j)$: distance satisfies the triangular inequality.

An interesting observation on P is that it demonstrates block patterns if we arrange the subjects from the same cluster together [11]. The heat map of P of the IRIS data set, in Figure 2.1, shows 9 blocks since IRIS is partitioned into 3 classes. The darkness in the heat map shows the smaller within-class dissimilarity. The darkest part forms a straight line on the diagonal since the distance of one subject to itself is zero.

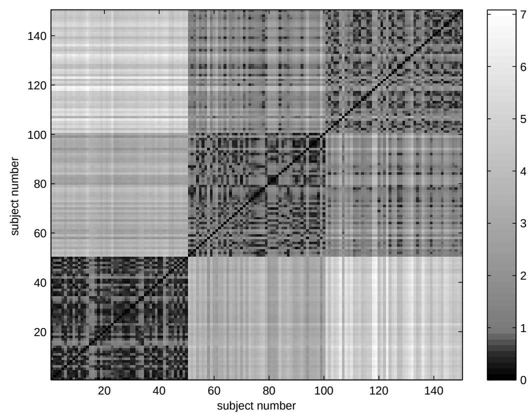


Figure 2.1: The dissimilarity matrix of the IRIS data set.

Pair-wise Distance Distortion (`DistVal`)

We define `DistVal` as the relative error of the difference between dissimilarity matrices of A and $\widetilde{A}$, in Frobenius norm as

$$\texttt{DistVal} = \frac{\| P - \widetilde{P} \|_F}{\| P \|_F}. \tag{2.17}$$

In our experiments, the redundant information in P is removed and only the lower triangular part of P is written column by column into a row vector of size $\frac{n \times (n-1)}{2}$, $pdist$. Therefore, (2.17) becomes

$$\texttt{DistVal} = \frac{\| pdist - \widetilde{pdist} \|_F}{\| pdist \|_F}. \tag{2.18}$$

Pair-wise Distance Maintenance Rate (`DistMaintain`).

We define `DistMaintain` as the percentage of distances that maintain their ranks in all the pair-wise distances after the distortion. It is computed as:

$$\texttt{DistMaintain} = \frac{\sum_{i=1}^{\frac{n\times(n-1)}{2}} Rpdist_i}{n \times (n-1)/2} \times 100\%, \tag{2.19}$$

where $Rpdist_i$ indicates whether a distance keeps its rank in all the pair-wise distances during the data distortion process:

$$Rpdist_i = \begin{cases} 1, & \text{if } PRank_i = \widetilde{PRank}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2.20}$$

$PRank_i$ is the rank of $p(i)$ in the $pdist \in \mathbb{R}^{1\times\frac{n\times(n-1)}{2}}$, and $\widetilde{PRank}_i$ denotes the rank of $p(i)$ in the $\widetilde{pdist} \in \mathbb{R}^{1\times\frac{n\times(n-1)}{2}}$. The larger the value of `DistMaintain` is, the better the pair-wise distance is kept in the distortion strategy. The distortion strategies with better maintenance of pair-wise distances are supposed to achieve higher accuracy in distance-based mining.

2.4.2 Attribute Correlation Distortion Metrics

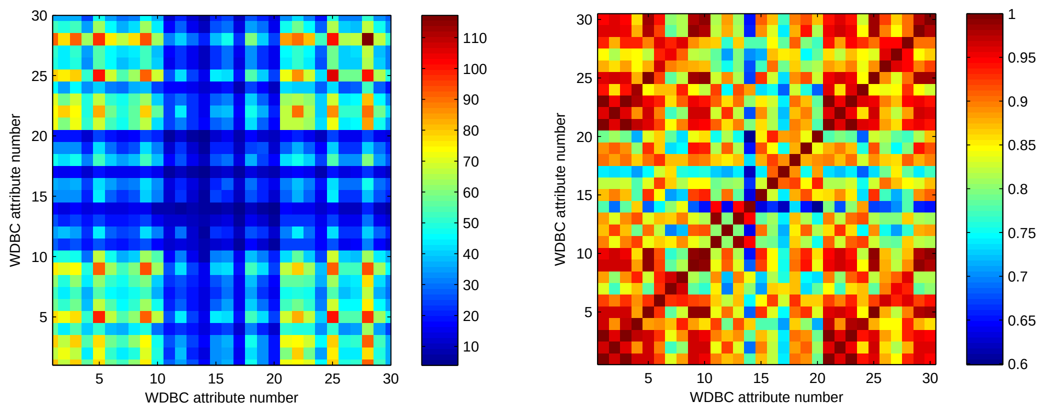
Attribute correlations affect the data mining results. With the zero mean, let s be the correlation of an attribute pair $(x,\ y)$, s is defined as a standard inner product $s =< x, y >= x^T y = \sum_k (x_k y_k)$. s can be used as the measure of how much two attributes vary together. If two attributes (x, y) tend to vary together, then s is positive. The zero value means an orthogonal relation, ***i.e.***, uncorrelated. Otherwise, s is negative if x and y vary oppositely.

The Correlation Matrix $\mathcal{S}$. We define a linear matrix $\mathcal{S} \in \mathbb{R}_+^{m\times m}$ where the correlation of an attribute pair $(A_{.i}, A_{.j})$, is defined as a standard inner product $S_{ij} =< A_{.i}, A_{.j} >= A_{.i}^T A_{.j} = \sum_k (A_{ki} A_{kj})$. All the pair-wise correlations are represented by S as

$$S = (S_{ij})_{i\in[1,m],\ j\in[1,m]} = A^T A. \tag{2.21}$$

By the above definition of S, S is a positive semidefinite symmetric matrix. Similar to P, S

also shows some pattern among attributes. Figure 2.2 shows two correlation matrices of the WDBC data set (refer to §2.7). In Figure 2.2(b), the correlation matrix is computed after each attribute is normalized to unit length by using unit-length normalization in §2.2.1. Therefore, the diagonal exhibits the darkest color which corresponds to the value of 1. In Figure 2.2(a), the correlation matrix is computed after each attribute is normalized by the range adjustment in §2.2.1. Even though two different normalizations are used, both of the figures display a similar pattern of a cross in the area covered by the middle 10 attributes ($11th$ to $20th$). It implies these 10 attributes have relatively lower correlations.



(a) $normal(A)^T \times normal(A)$.

(b) $normalc(A)^T \times normalc(A)$.

Figure 2.2: Correlation matrices of the WDBC data set.

Two metrics are designed to measure the difference of pair-wise attribute correlations after the data modification.

Correlation Distortion Metric (`CorrVal`)

We define `CorrVal` as the relative error of the value difference between S and $\widetilde{S}$, the correlation matrices of A and $\widetilde{A}$, in Frobenius norm as

$$\texttt{CorrVal} = \frac{\| S - \widetilde{S} \|_F}{\| S \|_F}. \tag{2.22}$$

Pair-wise Correlation Maintenance Rate (`CorrMaintain`).

We define `CorrMaintain` as the percentage of pair-wise correlations that maintain their ranks in all the pair-wise correlations after the distortion. It is computed as:

$$\texttt{CorrMaintain} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} Rank_{ij}}{m \times (m-1)/2} \times 100\%, \tag{2.23}$$

where $Rank_{ij}$ indicates whether a correlation keeps its rank in all the pair-wise correlations during the data distortion process:

$$Rank_{ij} = \begin{cases} 1, & \text{if } SRank_{ij} = \widetilde{SRank}_{ij}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.24}$$

$SRank_{ij}$ is the rank of S_{ij} in S, and $\widetilde{SRank}_{ij}$ denotes the rank of $\widetilde{S}_{ij}$ in $\widetilde{S}$. The larger the value of `CorrMaintain` is, the better the pair-wise correlation is kept in the distortion strategy. The distortion strategies with better maintenance of correlation are able to achieve higher mining accuracy.

2.4.3 Variance Preserving Rate (`VarP`)

For the SVD-based data modification methods, the amount of information preserved is quantified by the percentage of variance preserved in the distorted data. The metric of the variance preserving rate, denoted by `VarP`, is defined as a ratio of the sum of the preserved singular values to the sum of the total singular values in the original data set, formulated as

$$\texttt{VarP} = \frac{\sum_{i=1}^{k}\sigma_i}{\sum_{i=1}^{m}\sigma_i}, \tag{2.25}$$

where $\sigma_i(A) = \sqrt{\lambda_i(A^T A)}$, if both of singular values σ and eigenvalues λ are sorted by magnitude in the same order, usually, the descending order.

2.4.4 Summary

According to the definitions of these five pattern distortion metrics, the intuition on their relationship with data pattern distortion is that the higher the value of `DistMaintain`, `CorrMaintain` and `VarP`, and the lower those of `DistVal` and `CorrVal`, the more the original data pattern in A is maintained or the underlying information is less distorted. It should lead to better mining accuracy on $\widetilde{A}$. Next, the efficacy of the five metrics is examined on a small real data set, IRIS with 150 instances and 4 real attributes. (For a description of the IRIS data set, refer to §2.7).

From the experimental data in Table 2.2, an obvious trend is that with the increment of the rank k in the SVD, the relative error of data value (`RE`), dissimilarity matrix (`DistVal`) and correlation matrix (`CorrVal`) are decreasing; two maintenance percentages (`DistMaintain` and `CorrMaintain`) are increasing. The variance maintained becomes larger with the increment of k. Therefore, It experimentally turns out that these pattern distortion measures can practically evaluate the distortion level.

Table 2.2: Pattern distortion metrics of the rank-k SVD on the IRIS data set(150×4).

ThinSVD		*Data Pattern Distortion*				
rank	`RE`	`VarP`	`DistVal`	`Dist Maintain`	`CorrVal`	`Corr Maintain`
1	0.18593	0.80616	0.23399	0.02685	0.23318	0
2	0.04040	0.95507	0.10320	0.13423	0.02131	16.66667
3	0.01924	0.98421	0.05320	0.18792	0.02179	66.66667
4	0.00000	1	0.00000	72.49217	0	100

The singular values of IRIS are $[95.95, 17.72, 3.47, 1.88]$. Figure 2.3 shows a cumulative percentage line and singular value bars.

2.5 Experiments on Metrics

We conduct some experiments by using two data modification strategies, the thin SVD and noise-additive on one real data set YEAST (refer to §2.7.4) to examine the usefulness of the data value and data pattern metrics designed in §2.3 and §2.4. By observing Figure 2.4, when using the SVD to modify the YEAST data, for data value distortion metrics, it is found that `RE`, `RK` and `RP` show some nicely monotonically decreasing or increasing relationship with the decreasing of the ranks of approximation; while `CP` and `CK` do not show

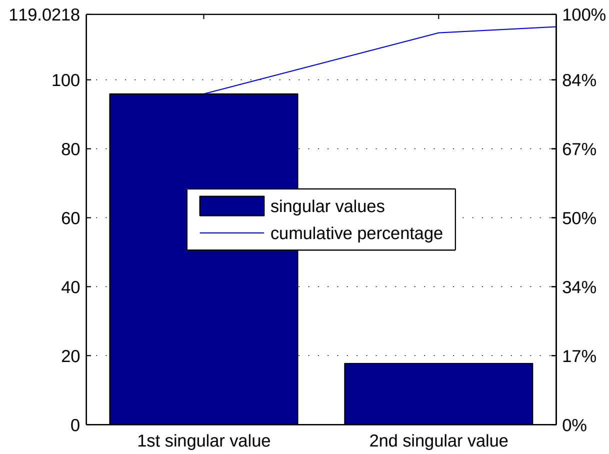


Figure 2.3: Cumulative percentage bar plot of singular values of IRIS.

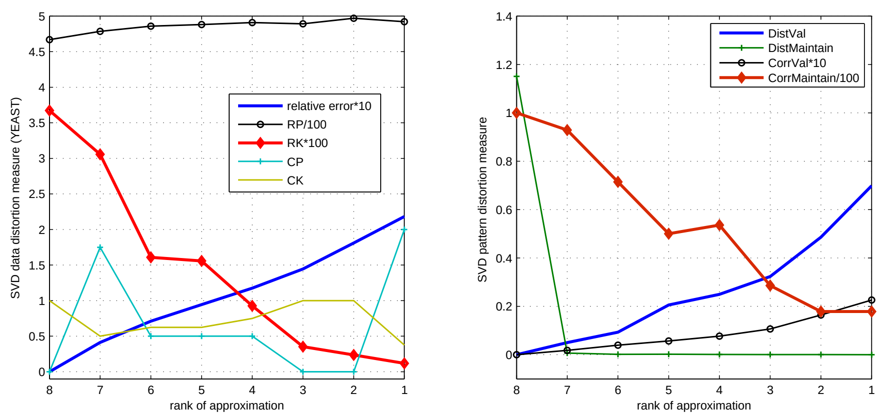


Figure 2.4: Distortion metrics of the rank-k SVD-based data distortion on YEAST.

clear trends. For data pattern distortion metrics, `DistVal` and `CorrVal` monotonically increase with the decrement of the ranks in the SVD; `CorrMaintain` almost monotonically decreases, while `DistMaintain` is almost zero for a rank range from 7 to 1.

Figure 2.5 shows the results of data value distortion metrics by adding two kinds of noise to the YEAST data, where RE and RK seem to be two suitable metrics for evaluating the value distortion by noise-additive methods. RE is almost linearly related to the magnitudes of the added noise, and RK is roughly negatively related to the magnitudes with frequent oscillations.

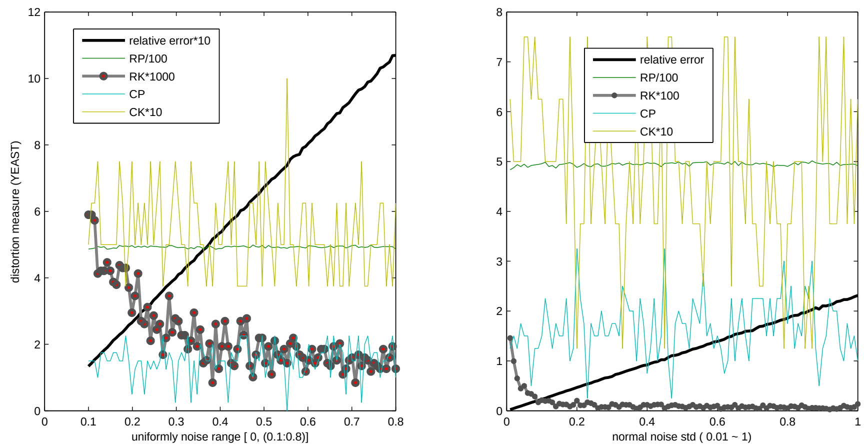


Figure 2.5: Distortion metrics of noise-additive data distortion on YEAST.

2.6 Mining Accuracy Metrics

By using data pattern distortion metrics, a relationship might be developed between the distortion level on characteristics of subjects or attributes, and the relatively accurate estimation on the final mining. This relationship provides possible recommendations on choosing data value distortion level in an attempt at achieving a balance between value protection and mining accuracy. Another way to assess the data pattern maintenance level is to compare the mining accuracy change after data modification. In our experiments, two popular mining techniques are used: $\mathcal{K}$-means clustering and the support vector machine (SVM) classification. It should be noted that the purpose here is to compare the accuracy difference after the data distortion, rather than to improve the mining accuracy.

For data clustering, ***Silhouette Value*** is a measure of how similar a subject is to subjects in its own cluster compared to subjects in other clusters. It ranges from -1 to $+1$. It is defined by MATLAB code as

$$\texttt{s(i)} = (\texttt{min}(\texttt{b}(\texttt{i},:),2) - \texttt{a}(\texttt{i}))./\texttt{max}(\texttt{a}(\texttt{i}),\texttt{min}(\texttt{b}(\texttt{i},:),2)), \tag{2.26}$$

where $a(i)$ is the average distance from the ith subject to the other subjects in its cluster, and $b(i,k)$ is the average distance from the ith subject to subjects in another cluster k.

For data classification, ***N-fold Cross-validation*** is used to calculate the classification accuracy. It can be achieved by the steps described in Algorithm 1 in Table 2.3. For

Table 2.3: Algorithm 1: N-fold cross-validation.

Algorithm 1 N-fold cross-validation.

```
Input:  a data set S, class truth C, a positive integer N, a
        classification algorithm L.
Output: classification accuracy ACC.
begin
    define an N-dimensional column vector, sum = zeros(N,1);
    partition the dataset into N subsets, S_1, S_2, ..., S_N;
    for i ← 1 to N do
        leave out one part of the data set, S_i, as the test data;
        train a prediction rule or model on the remaining (N − 1)
        subsets;
        sum(i) ← the classification accuracy on S_i;
    end
    take the average of the N accuracy values as the final mining
    accuracy. ACC ← mean(sum);
end
```

small datasets, the *leave-one-out* validation procedure is often used and N is the number of subjects. SVM classification is chosen as the classification accuracy metric by building a classifier on distorted dataset and applying N-fold cross-validation method to compute classification accuracy.

2.7 Four Real Data Sets

Four real data sets from UCI machine learning repository are used in our experiments [1]. Their names and dimension sizes are listed in Table 2.4.

Table 2.4: Four real data sets.

Name	Number of subjects	Number of attributes	Class number
IRIS	150	4	3
WDBC	569	30	2
YEAST	1484	8	10
WBC	699	9	2

2.7.1 Iris Plant Database (IRIS)

IRIS is a very simple data set with 150 instances in a 4-dimensional attribute space. The four attributes are sepal length, sepal width, petal length and petal width. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant: Iris Setosa, Iris Versicolour and Iris Virginica. Iris Setosa is linearly separable from the other two; the latter two are not linearly separable from each other. The misclassification rate for the Iris Setosa is 0%. The boxplots of four attributes grouped by three classes, in Figure 2.6, clearly demonstrate the $3rd$ or $4th$ attributes are highly related to the class labels; either one can accurately filter the Iris Setosa out. The reason is that there is no overlap of the value range of the $3rd$ and $4th$ attributes between the Iris Setosa and the other two classes.

2.7.2 Wisconsin Diagnostic Breast Cancer Database (WDBC)

WDBC is used for the purpose of diagnosis. Each of 569 instances has 30 real attributes. Two classes refer to two type of cancer: benign and malignant. 357 instances are in the group of benign and 212 are in the malignant group. It does not have missing values. The boxplot of the 30 attributes is shown in Figure 2.7. In the profile at UCI machine learning repository, the best known estimated accuracy is 97.5% using repeated 10-fold cross validations.

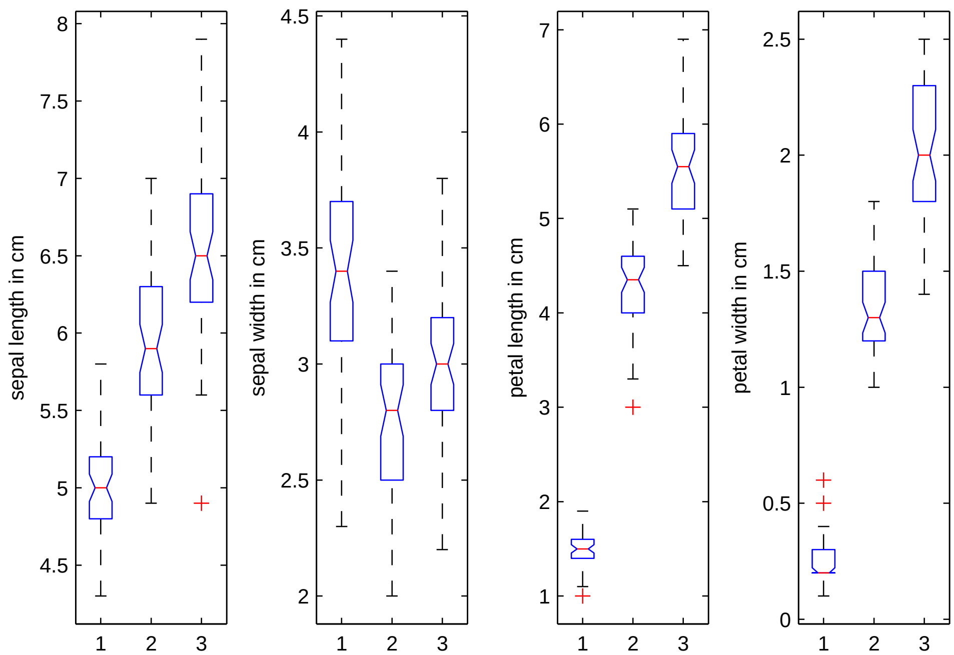


Figure 2.6: Boxplots of 4 attributes of the IRIS data set grouped by 3 classes.

2.7.3 Wisconsin Breast Cancer Database (WBC)

The original version is used here, which consists of 699 instances, 10 integer-valued attributes and one class attribute [1]. There are 16 missing attribute values for Bare Nuclei. Table 2.5 is a description of WBC original version. Some modifications on the original WBC dataset are performed. The missing values of Bare Nuclei are filled using the following rule:

$$\text{The missing value of Bare Nuclei } = \begin{cases} 1, & \text{if class label is benign;} \\ 8, & \text{if class label is malignant.} \end{cases}$$

The target WBC dataset is a matrix of size (699×10) with the 10th column representing the class label.

2.7.4 YEAST Database

The YEAST is a real-valued data set having 1484 instances and 8 attributes. It is used to predict the localization site of protein, which has 10 predications in Table 2.6. The boxplot

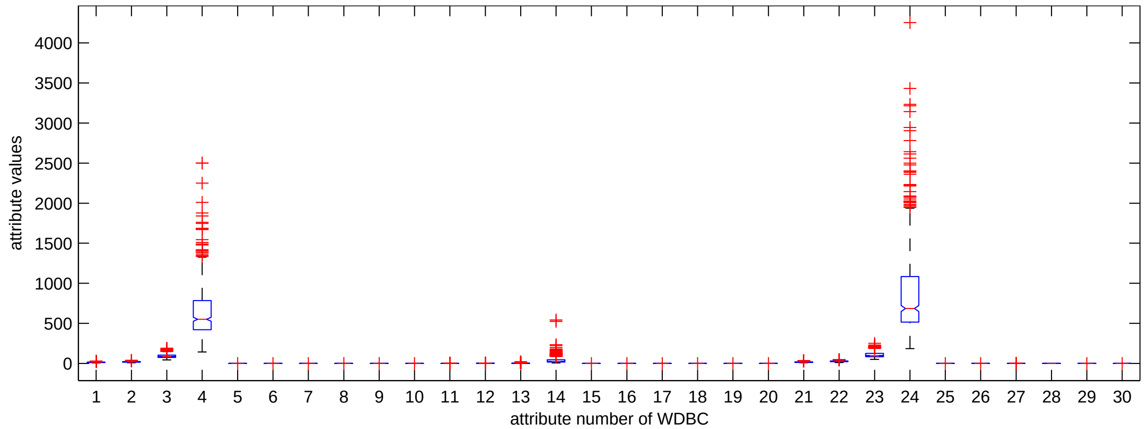


Figure 2.7: Boxplots of 30 attributes of the WDBC data set.

of each attribute grouped by 10 classes is in Figure 2.8.

Table 2.5: Attribute description of the WBC data set.

Number	*Attribute*	*Domain*
1	Sample Code Number	Id Number
2	Clump Thickness	1-10
3	Uniformity of Cell Size	1-10
4	Uniformity of Cell Shape	1-10
5	Marginal Adhesion	1-10
6	Single Epithelial Cell Size	1-10
7	Bare Nuclei	1-10
8	Bland Chromatin	1-10
9	Bare Nucleoli	1-10
10	Mitoses	1-10
11	Class	2 for benign, 4 for malignant
	Class distribution:	Benign: 458 (65.5%), Malignant: 241 (34.5%)

Table 2.6: Class distribution of the YEAST data set.

Number	1	2	3	4	5	6	7	8	9	10
Class Name	CYT	NUC	MIT	ME3	ME2	ME1	EXC	VAC	POX	ERL
Class Size	463	429	244	163	51	44	35	30	20	5

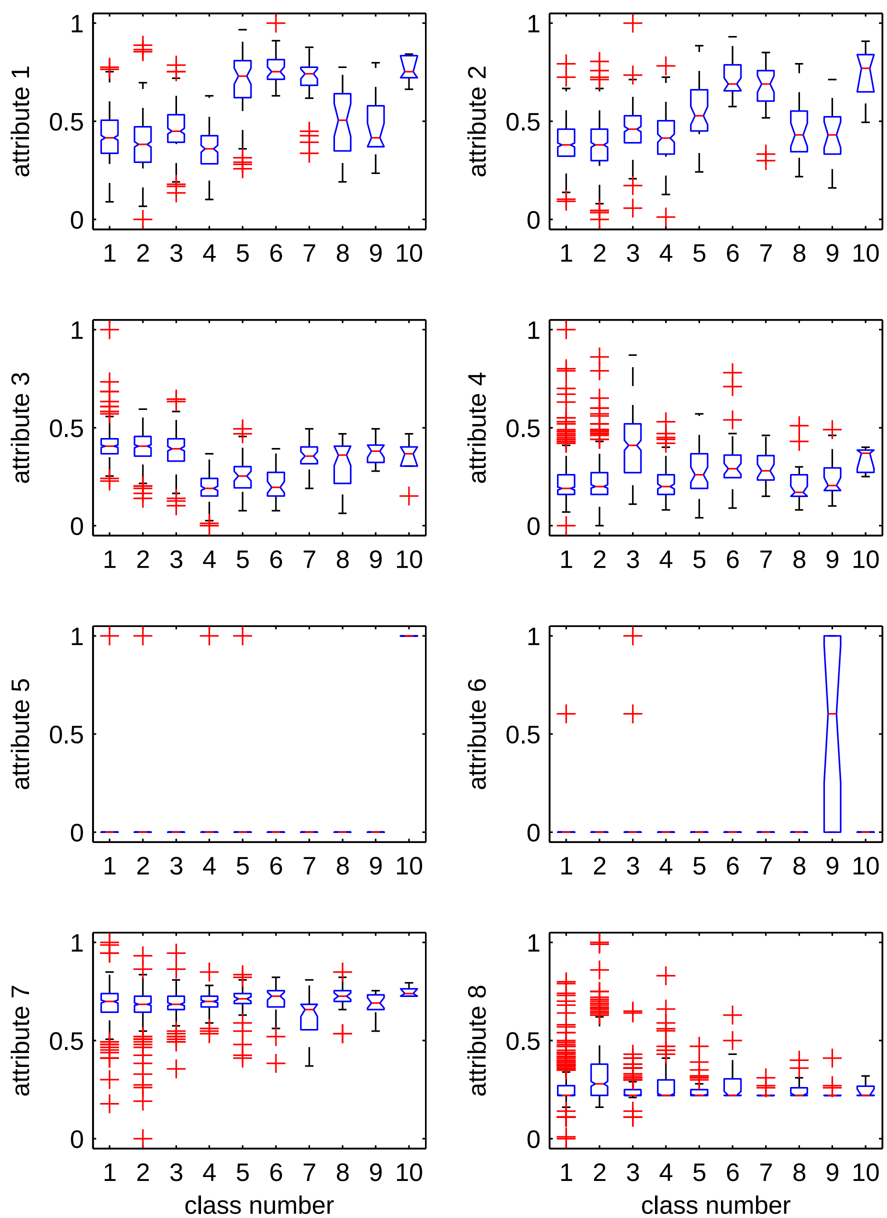


Figure 2.8: Boxplots of 8 attributes of the YEAST data set grouped by 10 classes.

Chapter 3

SVD-based Data Hiding Strategy

In abstract linear algebra terms, a matrix represents a linear transformation from one vector space, the domain, to another, the range. The singular value decomposition (SVD) implies that for any linear transformation, it is possible to choose an orthonormal basis for the domain and a possibly different orthonormal basis for the range.

The rank of a matrix is the number of linearly independent rows, which is the same as the number of linearly independent columns. The rank of a diagonal matrix is clearly the number of nonzero diagonal elements. Since orthogonal transformations preserve linear independence, the rank of any matrix is the number of nonzero singular values.

The Complete SVD. Referring to Definition 2. in § 2.1, the m columns of the data matrix A correspond to the attributes, and the n rows correspond to the subjects. Here, we assume the singular values are simple; *i.e.*, they are not repeated; and the rank of A is K, $K \leq \min\{n, m\}$. The *complete* singular value decomposition of a matrix of rank K can be written as

$$A = U \begin{bmatrix} \Sigma_{m \times m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{n \times m} V^T \tag{3.1}$$

or

$$U^T A V = \begin{bmatrix} \Sigma_{m \times m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{n \times m}. \tag{3.2}$$

where $A \in \mathbb{R}^{n \times m}, (n > m)$; $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{m \times m}$, U and V are orthonormal. $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_m)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m > 0$.

By (3.2), the ith singular value

$$\sigma_i = U_{.i}^T A V_{.i}. \tag{3.3}$$

By discarding zero entries in (3.1), we have

$$A = U_{.(1:m)} \Sigma_{m \times m} V^T = \sum_{i=1}^{m} \sigma_i U_{.i} V_{.i}^T, \tag{3.4}$$

where the matrix $U_{.(1:m)}$ is produced by removing the last $(n - m)$ columns from U.

The Compact SVD. The further simplification can be done. Setting

$$\begin{aligned} \Sigma_K &= diag(\sigma_1, \sigma_2, \ldots, \sigma_K), \\ \Sigma_{m-K} &= diag(\sigma_{K+1}, \sigma_{K+2}, \ldots, \sigma_m), \end{aligned}$$

then

$$\Sigma = \begin{bmatrix} \Sigma_K & \mathbf{0} \\ \mathbf{0} & \Sigma_{m-K} \end{bmatrix}. \tag{3.5}$$

Obviously, Σ_{m-K} is $\mathbf{0}_{m-K}$. Thus, (3.4) can be reduced to a *compact* representation as

$$A = U_{.(1:K)} \Sigma_K V_{.(1:K)}^T = \sum_{i=1}^{K} \sigma_i U_{.i} V_{.i}^T, \tag{3.6}$$

The SVD of A produces two orthonormal bases, one defined by the right singular vectors in V and the other by the left singular vectors in U. The right singular vectors, contained in V, span the row space of A and the left singular vectors, contained in U, span the column space of A. These three matrices, U, Σ, and V, reflect a transform of original relationship into linearly independent vectors.

Equivalently, the SVD decomposes A into a sum of rank-1 matrices generated by singular value *triplets*: $\sum_{i=1}^{m} \sigma_i U_{.i} V_{.i}^T$. The rank-$k$ *thin / truncated* SVD is generated if restricting this sum to the k triplets having the largest-magnitude singular values. That is the basis of our proposed SVD-based model.

The SVD equation for the ith subject in A can be represented as

$$A_i = \sum_{r=1}^{k} U_{ir} \Sigma_r V_r, \quad i = 1, 2, \ldots, n, \tag{3.7}$$

which is a linear combination of the right singular vectors V_r. The ith row of U, U_i, contains the coordinates of the ith subject A_i in the coordinate system (basis) of the scaled right singular vectors, $\Sigma_r V_r$. If $k < m$, the subjects may be reasonably well represented with fewer attributes using U_i rather than A_i. This property of the SVD is sometimes referred to as *dimensionality reduction*.

3.1 Theoretical Analysis of the SVD-Based Model

Due to the arrangement of the singular values in the matrix Σ (in a descending order), the SVD transformation has the property that the maximal variation among the objects is captured in the first singular value, as $\sigma_1 > \sigma_i$, for $i \geq 2$. Similarly much of the remaining variations is captured in the second dimension, and so on. Thus, a transformed matrix with a much lower dimension can be constructed to represent the original matrix faithfully. This property makes the SVD particularly interesting for our application of high accuracy data hiding.

It is possible, for Σ_K in (3.6), to retain only the first k, $k \ll K$, singular values by discarding other $(K - k)$ singular values. We term this reduced matrix Σ_k. Setting $\Sigma_k = diag(\sigma_1, \sigma_2, \ldots, \sigma_k)$, $\Sigma_{K-k} = diag(\sigma_{k+1}, \sigma_{k+2}, \ldots, \sigma_K)$, we have

$$\Sigma_K = \begin{bmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \Sigma_{K-k} \end{bmatrix}. \tag{3.8}$$

Then (3.6) can be written as

$$\begin{aligned} A &= U_{.(1:K)} \begin{bmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \Sigma_{K-k} \end{bmatrix} V^T_{.(1:K)} \\ &= U_{.(1:k)} \Sigma_k V^T_{.(1:k)} + U_{.(k+1:K)} \Sigma_{K-k} V^T_{.(k+1:K)} \\ &= \sum_{i=1}^{k} \sigma_i U_{.i} V^T_{.i} + \sum_{i=k+1}^{K} \sigma_i U_{.i} V^T_{.i}. \end{aligned} \tag{3.9}$$

Truncating the sum after first k triplets in (3.9), called the *truncated / thin* SVD in [33], the result is a rank-k approximation to the original matrix.

The Thin / Truncated SVD. Let $A^{(k)}$ be the rank-k approximation to A, and E_k be the error of this approximation, by (3.9), we know that

$$\begin{aligned} A &= A^{(k)} + E_k. \\ A^{(k)} &= U_{.(1:k)}\Sigma_k V^T_{.(1:k)} = \sum_{i=1}^{k} \sigma_i U_{.i} V^T_{.i}. \\ E_k &= U_{.(k+1:K)}\Sigma_{K-k} V^T_{.(k+1:K)} = \sum_{i=k+1}^{K} \sigma_i U_{.i} V^T_{.i}. \end{aligned} \tag{3.10}$$

Here, $U_{.(1:k)}$ and $V_{.(1:k)}$ represent the first k columns of U and V. A graphical depiction of the truncated SVD is shown in Figure 3.1.

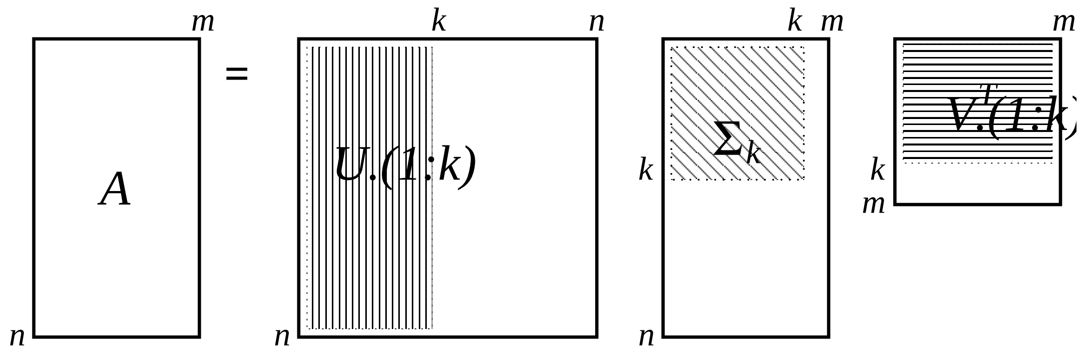


Figure 3.1: Graphical depiction of the singular value decomposition of a matrix A.

Then taking the Frobenius norm on E_k,

$$\begin{aligned} \|E_k\|_F^2 &= \|A - A^{(k)}\|_F^2 \\ &= \sum_{k+1}^{K} \sigma_i^2 \\ &= \sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots + \sigma_K^2. \end{aligned} \tag{3.11}$$

Therefore, the error, E_k in this approximation depends upon the magnitude of the neglected singular values. By the Schmidt (later Eckart-Young-Mirsky) theorem, the thin SVD is the optimal rank-k approximation of A under any unitarily invariant norm, including the Frobenius norm [56]. The proof is shown below.

Let $\widetilde{A} = A + E$ be a perturbation of A,

$$\widetilde{U}^T \widetilde{A} \widetilde{V} = \begin{bmatrix} \widetilde{\Sigma} \\ 0 \end{bmatrix}. \tag{3.12}$$

We use $\tilde{\sigma}_i$ and σ_{Ei} to denote the singular values of $\widetilde{A}$ and E.

The basic perturbation bounds for the SVD of a matrix are due to the following two theorems [70]. It is proven in the *Weyl* theorem that the singular values of a matrix are perfectly conditioned, and no singular value can move more than the norm of the perturbation,

$$|\tilde{\sigma}_i - \sigma_i| \leq \|E\|_2 = \sigma_{Emax}. \tag{3.13}$$

where σ_{Emax} is the greatest singular value of E. In the *Mirsky* theorem, it is proven that for any matrix B, produced by adding any perturbation, E, on A, there is a lower bound on the Frobenius norm of E,

$$\sqrt{\sum_i (\tilde{\sigma}_i - \sigma_i)^2} \leq \|E\|_F = \sqrt{\sum_i \sigma_{Ei}^2}. \tag{3.14}$$

By using the above two theorems, it has been proven that the distance between A and a rank-k approximation is minimized by the approximation $A^{(k)}$ in the sense of the Frobenius norm [29]. Let B be any matrix of rank not greater than k, and let the singular values of B be denoted by $\psi_1 \geq \ldots \psi_k > \psi_{k+1} = \psi_{k+2} = \cdots = \psi_m = 0$. By the Mirsky's theorem,

$$\begin{aligned} \|B - A\|_F^2 &\geq \sum_{i=1}^{m} |\psi_i - \sigma_i|^2 \\ &= \sum_{i=1}^{k} |\psi_i - \sigma_i|^2 + \sum_{k+1}^{m} |-\sigma_i|^2 \\ &\geq \sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots + \sigma_m^2, \end{aligned} \tag{3.15}$$

By (3.11), $\|B - A\|_F^2 \geq \|A - A^{(k)}\|_F^2$. Therefore, the matrix $A^{(k)}$ is a matrix of rank k that is nearest to A in the Frobenius norm [29].

Thin SVD for Data Hiding: The best rank-k approximation gives the additional interpretation of the thin SVD as a form of *noise suppression*, where A is presumed to be a low-rank data matrix containing attributes contaminated with additive Gaussian noise. Therefore, we may consider E_k in (3.11) as the additive noise in the original matrix A. Given the descending order of the singular values in Σ_K, in $A^{(k)}$, the first k most significant patterns are kept, and the $(K - k)$ less significant patterns are removed. Therefore, for extracting useful knowledge from data, it is pointed out that the low-rank approximation of the original space may be better than the original space itself due to the filtering out of the small singular values that represent noise [13].

Hence, using $A^{(k)}$ instead of A may yield better data mining accuracy. Simultaneously due to the value difference between A and $A^{(k)}$, the distorted data $A^{(k)}$ can preserve privacy, as it is difficult to figure out the exact values of A from those of $A^{(k)}$ without the knowledge of E_k. Hence, $A^{(k)}$ can be seen as both a distorted copy of A and a faithful representation of the original data. The significance of the truncated SVD for PPDM is reflected by the following three facts:

1. Value Difference: The data values are modified in $A^{(k)}$ and they are different from those in A.

2. Pattern Maintenance: The dominant data pattern in A is preserved in $A^{(k)}$.

3. Noise Removal: The noise represented by the small-magnitude singular values is filtered out in $A^{(k)}$.

The value difference can be utilized to protect data value disclosure. The pattern maintenance can be used to ensure the data mining accuracy and preserve data utility of the modified dataset. The noise removal may improve data mining accuracy.

3.2 Thin SVD-based Data Modification Method

If a certain value of k is determined by some privacy and accuracy metrics, $A^{(k)} =$`svds(A, k)`, can be directly used as the final modified dataset. We call it the basic or thin SVD-based data modification method as described in Algorithm 2 in Table 3.1.

3.3 Performance Comparison of Thin SVD, Noise-Additive and Random Projection

In this section, three series of experiments are conducted on the WDBC data set to examine three data distortion methods, the $\mathcal{K}$-means clustering accuracy and the classification accuracy of the support vector machine. The thin SVD, two noise-additive and four random

Table 3.1: Algorithm 2: Basic/thin SVD-based data modification method.

Algorithm 2 Basic/thin SVD-based data modification method.

```
Input:  a data set S with its vector-space model A, a learning
        algorithm L.
Output:  a modified data set Ã.
begin
    do SVD decomposition on A to compute U, Σ and V.
    r ← the number of nonzero diagonal elements of Σ.
    for k ← 1 to r − 1 do
        compute a modified data matrix:  A^(k) = U_.(1:k) Σ_k V_.(1:k);
        calculate data modification metrics on A^(k);
        examine the mining accuracy of A^(k);
    end
    choose one A^(k) as the final modified dataset Ã.
end
```

projection methods are used to distort the original data, respectively. $\mathcal{K}$-means clustering and the support vector machine classification are used here to examine the utilities of the distorted data. For the same database, in order to make a fair comparison, the same parameter configuration of data mining algorithms are used for all the generated distorted data versions from the original data set.

For a simple introduction of WDBC, please refer to §2.7.2. The normal noise matrix is generated for 100 times, where each entry is generated from a distribution, $\mathcal{N}(0, \sigma^2)$, where σ is some value from a linear space of $[0.2, 15, 100]$. The 100 upper limits of the uniformly noise matrix is drawn from a linear space of $[0.5, 20, 100]$. The four random projection matrices are generated for 100 times from an unknown distribution, $\mathcal{N}(0, \sigma_r^2)$, where σ_r is some value from a linear space of $[0.01, 10, 100]$.

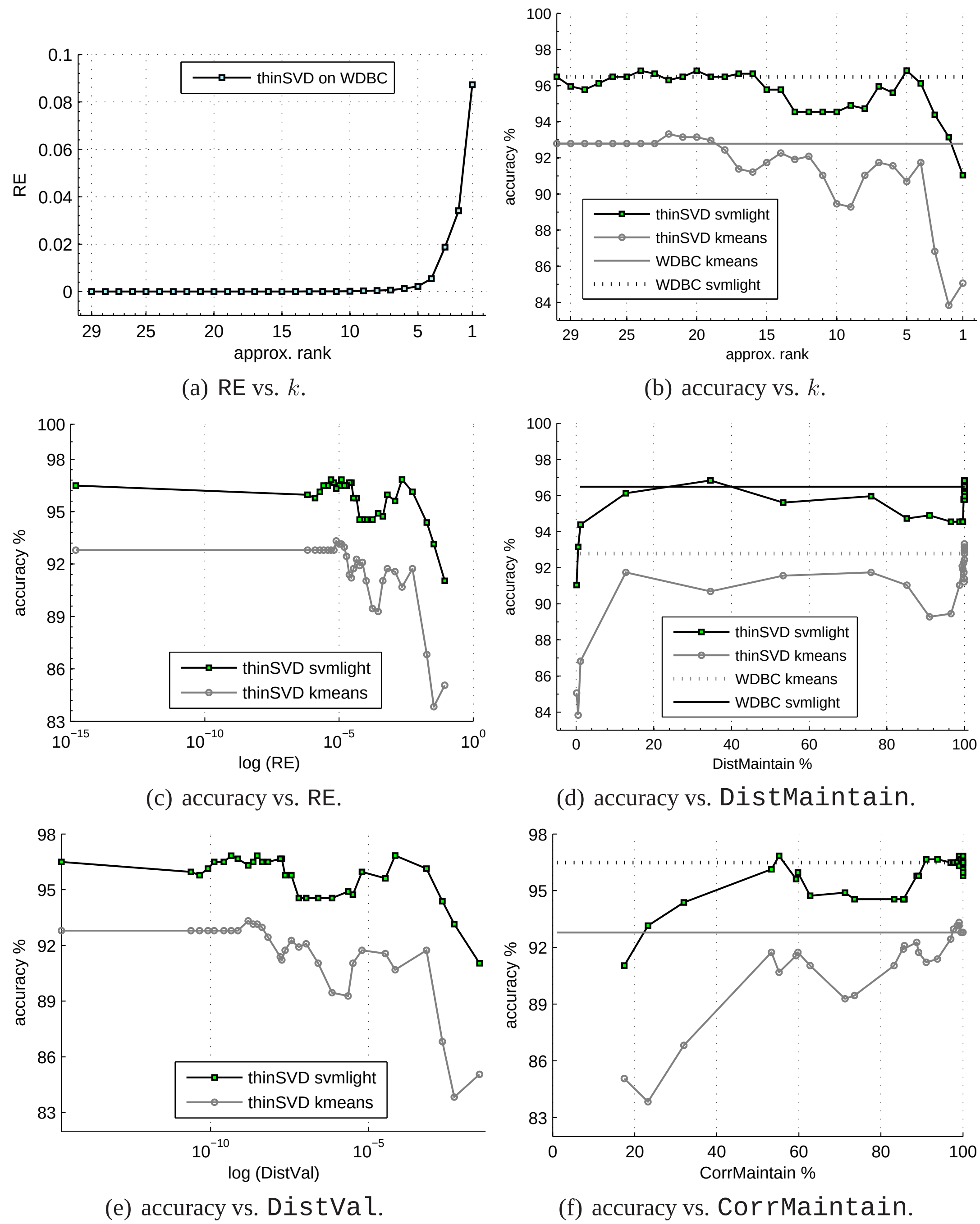
For the $\mathcal{K}$-means clustering, the initial starting cluster centers are fixed on the first 2 data points. For the SVMlight [40], the smallest value of each attribute is normalized to zero. Radial basis function is chosen as the kernel function and $\gamma = 1$. The original accuracies are 92.7944% for the $\mathcal{K}$-means clustering and 96.4912% for the SVMlight. The mean of accuracies is the average over all the 29 distorted data sets for the thin SVD-based method, the average over 30 samples for two noise-additive methods, an average over 10 samples for each of the four random projection methods.

3.3.1 Experimental Analysis of Thin SVD-based Data Modification

The 29 distorted data versions are generated by `svds(WDBC,k)`, where k is the rank of approximation from 1 to 29, which is the column size of the WDBC. On these rank-k distorted data sets, we examine data distortion and pattern distortion level. The experimental data is in Appendix A. The performance is shown in Figure 3.2.

(1). Relationship of RE vs. approximation rank k. Figure 3.2(a) shows the relative error RE as a function of the rank of approximation.

When k is 4, the order is 10^{-3}. When k is 7, the error is in the order of 10^{-4}. The lowest error is 6×10^{-7} when the approximation rank is 29. The shape-preserving data fitting by the black line in Figure 3.3 displays that $\log_{10}(\mathtt{RE})$ has a roughly linear relationship with the approximation rank of the thin SVD.



(a) RE vs. k. (b) accuracy vs. k.

(c) accuracy vs. RE. (d) accuracy vs. `DistMaintain`.

(e) accuracy vs. `DistVal`. (f) accuracy vs. `CorrMaintain`.

Figure 3.2: Performance evaluation of the thin SVD-based data distortion on WDBC.

(2). Relationship of mining accuracies vs. approximation rank k. The mining accuracies as a function of approximation rank are plotted in Figure 3.2(b), where the black line with green squares denotes the SVM classification accuracy and the gray line

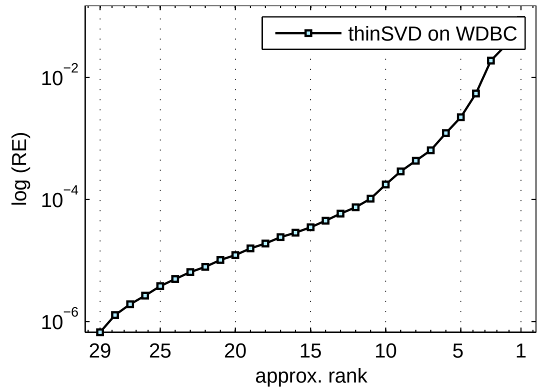
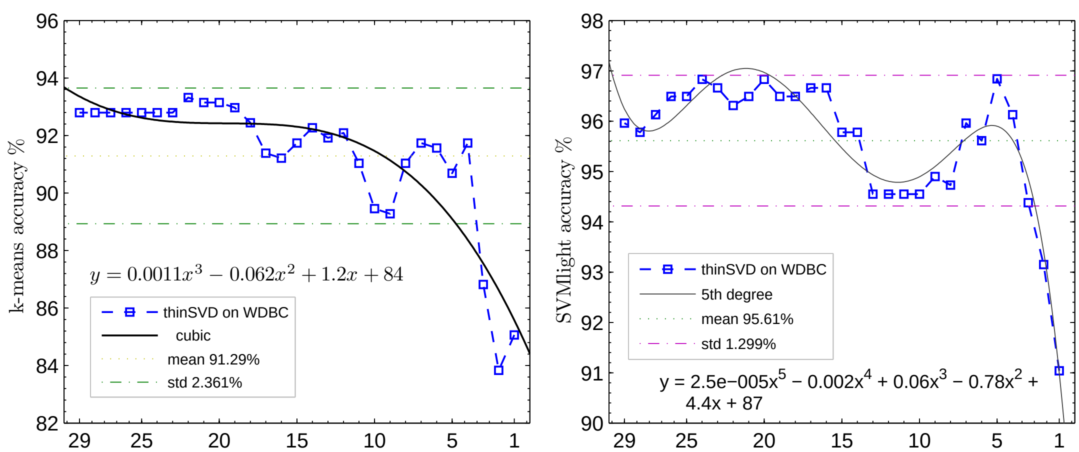


Figure 3.3: The log plot of RE as a function of approximation rank in the thin SVD-based data modification on WDBC.

with non-faced circles is the $\mathcal{K}$-means clustering accuracy. The individual plots of mining accuracy for the two methods are shown in Figure 3.4(a) and Figure 3.4(b). A general common pattern is displayed that the accuracy deteriorates with the decreasing rank in the thin SVD, and the distorted data versions mostly have lower accuracy but very comparable to the original accuracies.



(a) $\mathcal{K}$-means clustering accuracy vs. k.

(b) SVMlight classification accuracy vs. k.

Figure 3.4: The mining accuracy vs. approximation rank in WDBC.

Some distorted data versions perform quite well and achieve the same or better accuracies: 11 in SVMlight and 10 in $\mathcal{K}$-means. That means 38 percent of all the distorted data sets perform better on classification and 35 percent of them perform better on clustering.

A simple statistic analysis can be found in Table 3.2. It shows that the average accuracy of classification is 95.61 percent, which is 99.09 percent as good as the results over the original WDBC; the average accuracy of clustering is 91.29 percent, which is 98.38 percent as good as the results over the original WDBC.

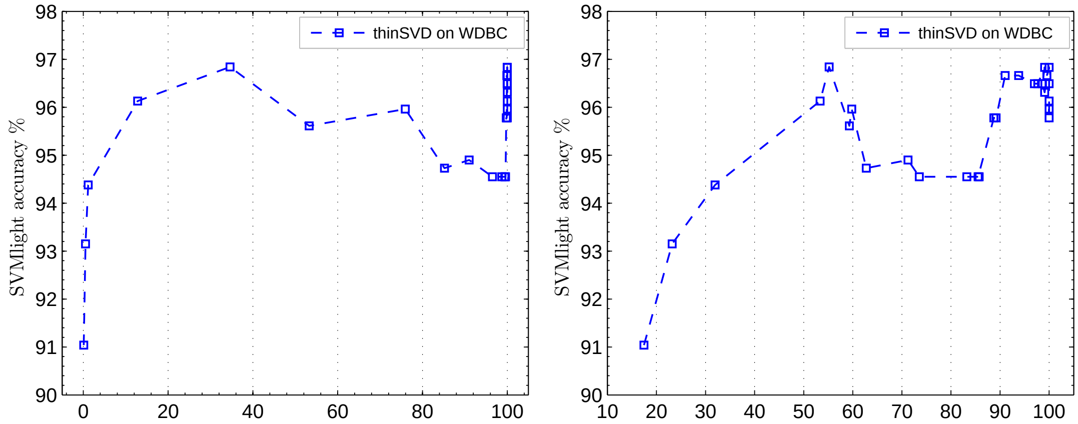
Table 3.2: Basic statistic analysis of the mining accuracies of the thin SVD-based data modification on WDBC.

Mining algorithms Name	RE Mean	Accuracy (%) Original	Max.	Min.	Mean	Std	Max.rel.err
$\mathcal{K}$-means clustering	0.0052	92.79	93.32	83.83	91.29	2.36	9.65
SVMlight classification		96.49	96.84	91.04	95.61	1.29	5.65

(3). Relationship of mining accuracies vs. `RE` and `DistVal`. The mining accuracies as a function of $log(\texttt{RE})$ and $log(\texttt{DistVal})$ are plotted in Figure 3.2(c) and Figure 3.2(e), where the black line with green squares denotes the $\mathcal{K}$-means clustering and the gray line with non-faced squares is the SVM classification accuracy. The leftmost point represents the original accuracy. Compared to Figure 3.2(b), it is found that the plots in these three figures demonstrate similar changing patterns. Generally, the accuracies are negatively related to `RE` and `DistVal`.

(4). Relationship of mining accuracies vs. `DistMaintain` and `CorrMaintain`. The mining accuracies as a function of `DistMaintain` and `CorrMaintain` are plotted in Figure 3.2(d) and Figure 3.2(e), where the black line with green squares denotes the $\mathcal{K}$-means clustering and the gray line with non-faced squares is the SVM classification accuracy. The leftmost point represents the original accuracy. Compared to Figure 3.2(b), it is found that the plots in these three figures demonstrate similar changing patterns. Generally, the accuracies are negatively related to `RE` and `DistVal`.

Figure 3.5(a) and Figure 3.5(b) show the SVMlight classification accuracy as a function of `DistMaintain` and `CorrMaintain`, respectively.



(a) accuracy vs. `DistMaintain`. (b) accuracy vs. `CorrMaintain`.

Figure 3.5: SVM classification accuracy vs. `DistMaintain` and `CorrMaintain` for the thin SVD-based data modification in WDBC.

3.3.2 Experimental Analysis of Noise-additive Data Modification

Two kinds of noise are added to the WDBC data. One is generated from uniform distribution with a range starting from 0 to some real-valued upper limit. The other is from some normal distribution with zero mean and some variance. The experiment is repeated for 100 times with the value of standard deviation σ taken from a linear space linspace(0.2,15,100), and the value of upper limit taken from a linear space linspace(0.5,20,100).

(1). Relationship of RE vs. noise magnitudes (σ and upper limit). Figure 3.6 shows the relative error RE as a function of the noise magnitudes.

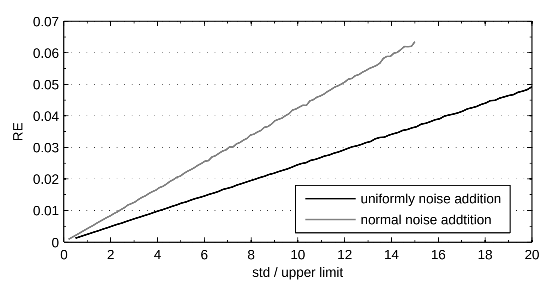


Figure 3.6: RE as a function of noise magnitude in noise-additive data distortion on WDBC.

The blue solid line represents the uniformly distributed noise and the green dash line is for the normal distributed noise. Obviously, both of them display a linear positive relationship between the RE and the standard deviation σ or the upper limit. The RE of the normal noise has a steeper rise than that of the uniformly noise with the same increment of σ and the upper limit.

(2). Relationship of `DistMaintain` and `CorrMaintain` with the noise magnitudes. The effects of the noise magnitudes on the `DistMaintain` and `CorrMaintain` are examined here and the results are shown in Figure 3.7. As in Figure 3.7(b) and Figure 3.7(e), the relationships between `DistMaintain` and the noise magnitudes are monotonically decreasing functions. For the `CorrMaintain` vs. the noise magnitudes, the plots demonstrate very rough and approximately decreasing functions as in Figure 3.7(c) and Figure 3.7(f). Therefore, in general, both of `DistMaintain` and `CorrMaintain` are negatively related to the magnitude of the added noise, and `DistMaintain` has a much smoother variation than `CorrMaintain`.

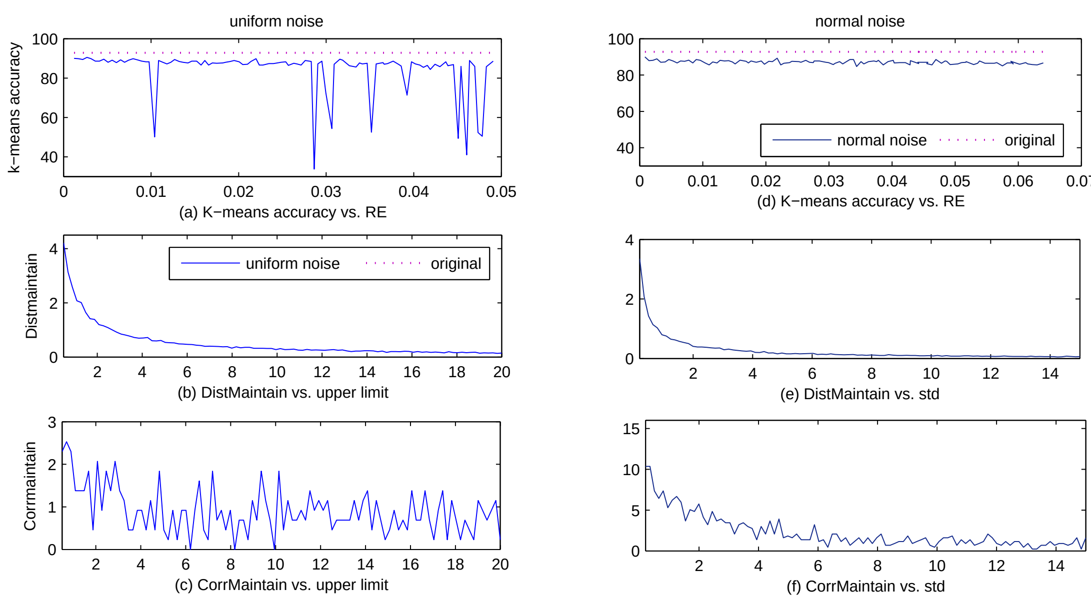


Figure 3.7: Performance evaluation of noise-additive data distortion on WDBC.

(3). Relationship of $\mathcal{K}$-means clustering accuracy and RE. Referring to Figure 3.7(a) and Figure 3.7(d), a reasonable result is shown that for noise-additive methods, the accuracy also decreases with the increasing of RE. However, the addition of noise degrades the clustering accuracy and it is found that all the distorted data versions have lower accuracies than the original one. A basic statistic analysis is shown in Table 3.3. The average accuracy of the uniform noise-additive method is 84.31%, which is 90.86 percent as good as the original accuracy. The average accuracy of the normal noise-additive method is 87.05%, which is 93.61 percent as good as the original accuracy. Furthermore, the accuracy of the uniform noise-additive method is not as stable as that of the normal noise-additive method and it has a larger $\sigma = 11.21$.

Table 3.3: Basic statistic analysis of $\mathcal{K}$-means accuracy of the noise-additive data modification on WDBC.

Noise Name	Mean (upper limit/std)	RE	$\mathcal{K}$-means Accuracy (%) Original	Max.	Min.	Mean	Std	Max.rel.err
Uniformly	10.25	0.0250	92.79	90.51	33.74	84.31	11.21	63.63
Normal	7.6	0.0321	92.79	89.98	84.71	87.05	0.94	9.65

3.3.3 Experimental Analysis of Random Projection Data Modification

The projection matrix, R, is created by randomly sampling from some distribution with zero mean and some variance σ_r^2. Computationally, it is a matrix multiplication. Two cases exist here, *left multiplication* and *right multiplication*. The size of R is $m \times m$ for the right multiplication and $n \times n$ for the left multiplication, since in our study, the dimensions of the original and the perturbed matrices are kept to be the same. For each case, there are two different R, nonorthonormal and orthonormal. Four short names as described in Table 3.4 are used here: Arp, $Arpo$, rpA and $rpoA$.

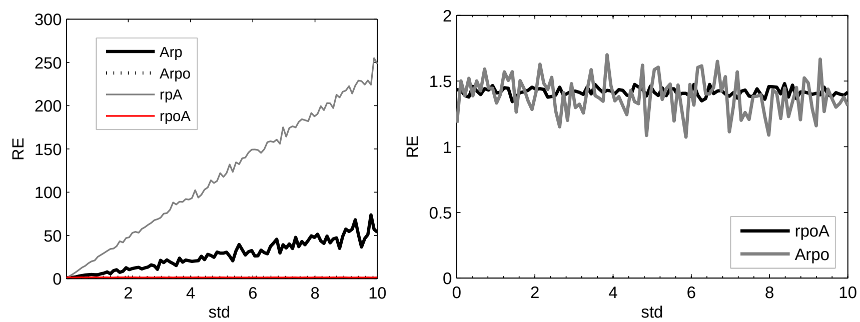
The 100 distorted data versions are generated by choosing the standard deviation σ_r from a linear space ranging from 0.01 to 10.

(1). Relationship of RE and σ_r. Referring to Figure 3.8, we can find the nonorthonormal projections bring out the large RE. The left multiplication method, rpA, distorts the data values more than the right multiplication method, Arp. Orthonormal projec-

Table 3.4: The notation of four random projection methods.

Method Name	Method
Arp	$\widetilde{A} = AR$, $R \in \mathbb{R}^{m\times m}$.
$Arpo$	$\widetilde{A} = AR$, $R \in \mathbb{R}^{m\times m}$, $RR^T = I$.
rpA	$\widetilde{A} = RA$, $R \in \mathbb{R}^{n\times n}$.
$rpoA$	$\widetilde{A} = RA$, $R \in \mathbb{R}^{n\times n}$, $R^TR = I$.

tions have a stable RE with an increasing σ_r, since the orthonormalization makes columns or rows unit length. Figure 3.8(b) shows that $rpoA$ has a smaller magnitude of RE than $Arpo$. A basic statistic analysis is in Table 3.5.



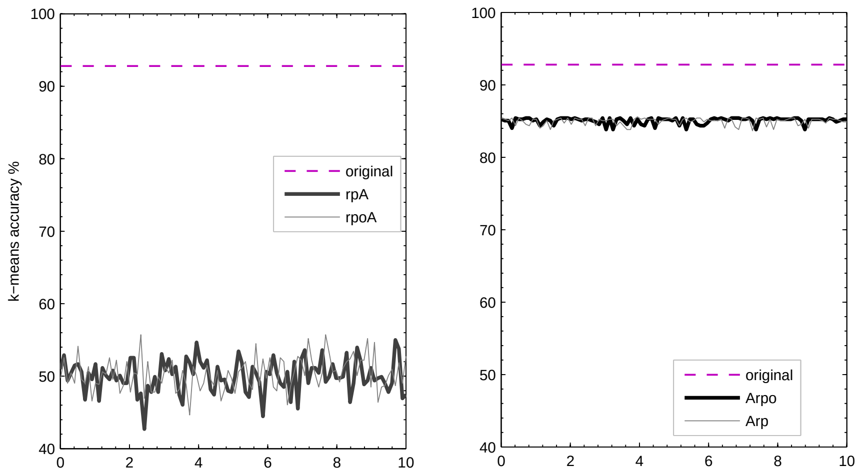
(a) RE vs. σ_r.　　(b) RE vs. σ_r.

Figure 3.8: RE as a function of σ_r in random projection data modification on WDBC.

(2). `DistMaintain` and `CorrMaintain` in four methods. $Arpo$ maintains the dissimilarity matrix and `DistMaintain` is always 100%. $rpoA$ maintains the correlation matrix and `CorrMaintain` is always 100%. The left multiplication methods have very low `DistMaintain` which is in the order of 10^{-3}, that might be the reason for their poor performance on the $\mathcal{K}$-means clustering.

(3). $\mathcal{K}$-means accuracies of four methods. All the four methods perform worse than the original data in $\mathcal{K}$-means clustering. It seems no obvious effect of σ_r on the accuracy, shown in Figure 3.9. Due to the fact that the left multiplication methods, rpA and $rpoA$, change the dissimilarity matrix almost as large as 100%, the experimental results show that their accuracies in $\mathcal{K}$-means clustering are very low with the average accuracies being

50.01% and 50.31%. The right multiplication methods, Arp and $Arpo$, have much better accuracies whose mean values are 84.97% and 85.06%. More detailed results can be found in Table 3.5.



(a) Left multiplication: K-means accuracy vs. σ_r. (b) Right multiplication: K-means accuracy vs. σ_r.

Figure 3.9: $\mathcal{K}$-means accuracy vs. σ_r in WDBC.

Table 3.5: Basic statistic analysis of random projection data modification on WDBC.

Methods	σ_r	$[0.01, 10]$				
	RE	Mean	Std	Max.	Min.	
Arp		27.4554	17.0362	73.6637	0.9721	
$Arpo$		1.3896	0.1386	1.6994	1.0927	
rpA		119.7492	70.5045	254.9169	1.0255	
$rpoA$		1.4157	0.0306	1.4801	1.3417	
	$\mathcal{K}$-means	Mean	Std	Max.	Min.	Max.rel.err.
Arp	original	84.9717	0.4866	85.5888	83.6555	9.84
$Arpo$	92.79%	85.0615	0.4391	85.4130	83.8313	9.65
rpA		50.0141	2.2544	55.0088	42.7065	53.98
$rpoA$		50.3146	2.2066	55.7118	44.6397	51.89
	SVMlight	Mean	Std	Max.	Min.	Max.rel.err.
Arp	original	94.2296	0.4847	95.0791	93.4974	3.45
$Arpo$	96.49%	94.1711	0.4423	94.9033	93.4974	3.45
rpA		52.7387	1.8256	56.0633	50.7909	47.55
$rpoA$		53.8079	2.0887	56.7663	49.7364	48.64

3.3.4 Summary

Based on the foregoing experimental results on the WDBC data, firstly, the average values of several metrics are combined in Table 3.7 so that a clear comparison can be observed. Secondly, one distorted data version is selected from each of the seven methods. The selection rule is to make the RE value of them as close as possible. The combination of the metrics of these seven data versions can be found in Table 3.6.

Table 3.6: Accuracy comparison of seven methods on WDBC.

Methods	Metrics			
	Parameter	RE	RP	RK
thinSVD	rank= 4	0.0054	171.5687	0.0800
uniformNoise	2.1850	0.0054	175.4101	0.0664
normalNoise	$\sigma = 1.2700$	0.0054	181.5627	0.0456
Arp	$\sigma_r = 0.1109$	0.9721	187.8826	0.0076
$Arpo$	$\sigma_r = 5.8627$	1.0727	188.3002	0.0060
rpA	$\sigma_r = 0.0100$	1.0255	188.9100	0.0019
$rpoA$	$\sigma_r = 1.4227$	1.3417	189.3051	0.0015
(-%-%)	DistVal	DistMaintain	CorrVal	CorrMaintain
thinSVD	0.0007	12.8134	0.0000	53.3333
uniformNoise	0.0009	1.1597	0.0059	0.9195
normalNoise	0.0019	0.6355	0.0003	5.7471
Arp	0.4036	0.1714	1.0186	0.0000
$Arpo$	0.0000	100.0000	1.2769	0.2299
rpA	0.8226	0.0012	0.9442	67.3563
$rpoA$	1.5605	0.0012	0.0000	100.0000
(%)	$\mathcal{K}$-means	SVMlight		
thinSVD	91.7399	96.1300		
uniformNoise	87.1705	92.8516		
normalNoise	87.6977	90.1200		
Arp	85.2373	95.0791		
$Arpo$	84.3585	93.6731		
rpA	50.9666	51.1424		
$rpoA$	52.5483	53.9543		

At this moment, some conclusions can be drawn from these experiments as follows:

1. The thin SVD-based method has the highest average accuracies both in SVMlight and $\mathcal{K}$-means. It is even possible for some distorted data to achieve a better performance on data mining than the original data. On the other hand, its data value distortion level is relatively lower than the other methods since there is no external noise introduced

into the original data.

2. The two left-multiplication-based methods have very poor performance on mining. On the other hand, the non-orthonomal left multiplication method can realize the greatest data value distortion among the seven methods. Therefore, if the maintenance of mining accuracy is considered valuable in real world applications, then these two methods can be removed from the candidate list.

3. For the maintenance of subject-pair-wise Euclidean distances, the orthonormal right-multiplicative random projection can keep the distances as good as 100%.

4. For the maintenance of attribute-pair-wise dot product, the orthonormal left-multiplicative random projection maintains the original dot product matrix.

5. For noise-additive methods, the normal-noise-based method has a more stable performance than the uniform-noise-based method.

6. Orthonormalization of projection matrices is capable of controlling the magnitude of the data value distortion level and making it independent of the magnitude of the external noise added into the original data.

7. Refer to the seven data versions of each of seven methods in Table 3.6, the random projection methods have the better data value distortion capability than the other three methods; the thin SVD-based method has the better accuracies than the other six methods.

8. A possible advantage of the thin SVD-based method over other methods, is that its accuracies are traceable from the approximation rank of the SVD, unlike the other 6 methods whose accuracies are unpredictable with the characteristic of randomization.

Table 3.7: A comparison of thin SVD, noise-additive and random projection data modification strategies on WDBC.

	Methods					
RE		Mean	Std	Max.	Min.	
	thin SVD	0.0052	0.0173	0.0872	0.0000	
	normalNoise	0.0321	0.0183	0.0635	0.0008	
	uniformNoise	0.0250	0.0140	0.0492	0.0012	
	Arp	27.4554	17.0362	73.6637	0.9721	
	$Arpo$	1.3896	0.1386	1.6994	1.0727	
	rpA	119.7492	70.5045	254.9169	1.0255	
	$rpoA$	1.4157	0.0306	1.4801	1.3417	
DistMaintain	(%)	Mean	Std	Max.	Min.	
	thin SVD	80.9701	34.8996	100.0000	0.0978	
	normalNoise	0.2573	0.4310	3.3850	0.0545	
	uniformNoise	0.5158	0.6251	4.1461	0.1355	
	Arp	0.1223	0.0787	0.4437	0.0316	
	$Arpo$	100	0	100	100	
	rpA	0.0005	0.0006	0.0025	0	
	$rpoA$	0.0005	0.0006	0.0019	0	
CorrMaintain	(%)	Mean	Std	Max.	Min.	
	thin SVD	79.8652	24.6604	100.000	17.4713	
	normalNoise	2.3103	2.3057	13.5632	0	
	uniformNoise	1.1061	0.6210	2.9885	0	
	Arp	0.2161	0.2441	1.1494	0	
	$Arpo$	0.1885	0.2050	0.9195	0	
	rpA	55.9747	9.0325	74.2529	29.8851	
	$rpoA$	100	0	100	100	
$\mathcal{K}$-means	(%)	Mean	Std	Max.	Min.	Max.rel.err.
	thin SVD	91.2914	2.3605	93.3216	83.8313	9.65
	normalNoise	87.0492	0.9419	89.9824	84.7100	8.71
	uniformNoise	84.3058	11.2116	90.5097	33.7434	63.63
original	Arp	84.9719	0.4866	85.5888	83.6555	9.84
92.79%	$Arpo$	85.0615	0.4391	85.4130	83.8313	9.65
	rpA	50.0141	2.2544	55.0088	42.7065	53.98
	$rpoA$	50.3146	2.2066	55.7118	44.6397	51.89
SVMlight	(%)	Mean	Std	Max.	Min.	Max.rel.err.
10-fold	thin SVD	95.61	1.29	96.84	91.04	5.65
rbf kernel	normalNoise	89.49		92.95	86.01	10.86
$\gamma = 1$	uniformNoise	91.27		94.29	88.16	8.63
original	Arp	94.23	0.48	95.07	93.49	3.1
96.49%	$Arpo$	94.17	0.44	94.90	93.49	3.1
	rpA	52.74	1.83	56.06	50.79	47.36
	$rpoA$	53.81	2.09	56.76	49.73	48.46

3.4 Sparsified Strategies

On the basis of the thin SVD-based data modification model, in order to do further distortion on the data values, sparsification is introduced to make a variant of the thin SVD. Three SVD sparsification strategies, which are *single threshold strategy (STS)*, *column threshold strategy (CTS)* and *exponential threshold strategy (ETS)*, have been proposed by Gao and Zhang for reducing the storage cost and enhancing the performance of the SVD in the area of information retrieval [32]. All these three strategies are used in our study to perform sparsification on $U_{.(1:k)}$ and $V_{.(1:k)}$ to further distort data values after the rank reduction by the thin SVD.

3.4.1 Three Sparsified methods

Let $\overline{U_{.(1:k)}}$ and $\overline{V_{.(1:k)}}$ denote the new matrices created after performing sparsification on $U_{.(1:k)}$ and $V_{.(1:k)}$ respectively, and the new version of the distorted matrix $A^{(k)}$ is

$$\overline{A}^{(k)} = \overline{U_{.(1:k)}} \Sigma_k \overline{V_{(1:k)}}^T. \tag{3.16}$$

Obviously the degree of perturbation of $\overline{A^{(k)}}$ is larger than that of $A^{(k)}$ and the protection on data privacy is improved.

- **Single Threshold Strategy (STS)**

  The basic idea of STS-based sparsification is that, given a certain threshold value $\epsilon > 0$, for any u_{ij} in U_k, if $|u_{ij}| < \epsilon$, we set $u_{ij} = 0$. The same operation is conducted on V_k^T. We use **s-SVD** to denote an SVD-based data modification method using STS sparsification strategy.

- **Column Threshold Strategy (CTS)**

  Given a scaling parameter $\epsilon > 0$, the threshold value for each column of U_k and V_k

is the product of the mean value of each column and ϵ.

$$T_j = \frac{\epsilon}{n} \sum_{i=1}^{n} |u_{ij}|, j = 1, 2, \ldots, m, \tag{3.17}$$

We use **c-SVD** to denote an SVD-based data modification method using CTS sparsification strategy.

- **Exponential Threshold Strategy (ETS)**

  The threshold value is determined by an exponential function:

$$T_j = \frac{\epsilon}{n} \sum_{i=1}^{n} |u_{ij}| e^{(\alpha j)^2}, j = 1, 2, \ldots, m \tag{3.18}$$

  where $\alpha > 0$ is a parameter, which should be on the order of $1/k$. It can be seen that a column with a larger index has a larger threshold value and more entries will be removed for this column [32]. We use **e-SVD** to denote an SVD-based data modification method using ETS sparsification strategy.

3.4.2 Experimental Evaluation

1. Magnifying data value distortion on WDBC. Several threshold values are examined and it turns out that it is appropriate that two different threshold values, ϵ_u and ϵ_v, are applied to sparsify $U_{.(1:k)}$ and $V_{.(1:k)}$, respectively. Here, $\epsilon_v = 0.02$ and ϵ_u changes from 0.02 to 0.06 with a step size of 0.002. Nine different approximation ranks of the thin SVD are tested, 1, 3, 4, 7, 20, 22, 23, 25, 27. The experimental data can be found in Appendices H1-H9. Obviously, all the data value distortion metrics are improved by the introduction of the sparsification.

In Figure 3.10, the lower nine plots are functions of RE with ϵ_u, and the upper nine plots are functions of $\mathcal{K}$-means accuracy with ϵ_u. We first note that the lower nine plots are almost completely overlapped, except that the plot for the rank of 1 has a little bit higher RE. That leads to *two possible implications*, for some data:

1. s-SVD may be able to make RE and the approximation rank independent of each

other. It might provide an answer for the choice of the approximation rank when doing thin SVD.

2. RE is dependent on the sparsification threshold values, which implies that adjustment on the sparsification level could control the data value distortion level in s-SVD.

The second point to note in Figure 3.10, is that the accuracies of $\mathcal{K}$-means clustering, although using different ranks in s-SVD, are approximately equal when ϵ_u is larger than 0.028. Further, the plots suggest a possible appropriate value for ϵ_u, and it is the peak point associated with $\epsilon_u = 0.036$, the best accuracy of 90.8612% and RE$= 0.4886$. Then the distorted data under different ranks are tested for the SVMlight classification accuracy. The lowest is 91.2127% at the rank of 4. The best is 92.4429% at the rank of 22.

If comparing this peak point to the average results for the thin SVD, the RE is increased by 9257.69%, the $\mathcal{K}$-means clustering accuracy is decreased by 0.47% and the SVMlight accuracy is decreased by 3.31%.

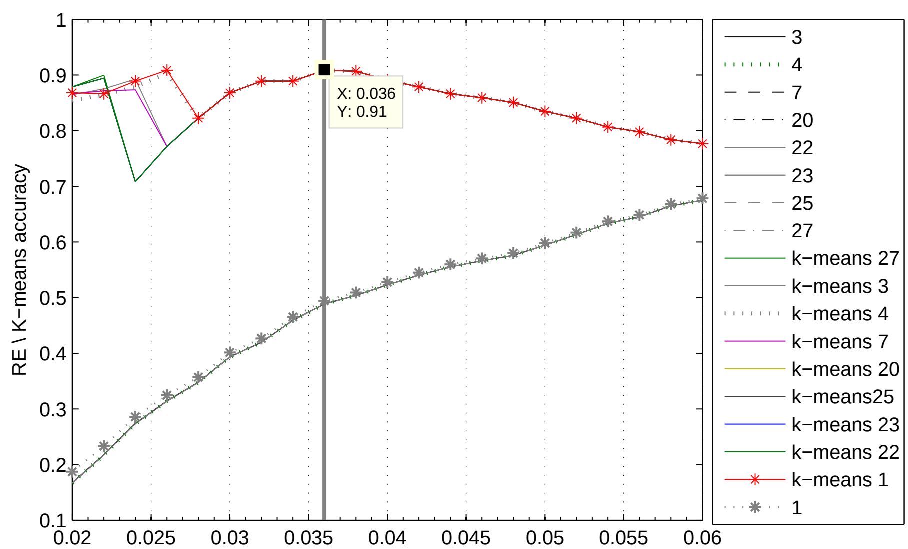


Figure 3.10: $\mathcal{K}$-means accuracy and RE as functions of threshold value ϵ_u by s-SVD on WDBC.

2. Comparison of sparsified SVD with thin SVD and noise-additive methods on WBC. A comparison is conducted on the WBC data set. In order to be fair in comparing the privacy metrics, parameters are set to make RE values as close as possible. The rank of thin SVD is 7. The results of performance evaluation on six methods are provided in Table 3.8.

Under the premise on the same level of value dissimilarity, the fact that CP value of uniform noise method and normal noise method is 0 and CK value is 1 indicates that both methods do not change any rank of the attributes. Experimental data in Table 3.8 supports the previous conclusions that SVD-based strategies achieve higher-level privacy protection than noise-additive methods. And sparsified-SVD-based methods are better than the thin SVD-based method on data distortion level without any significant degradation on classification accuracy.

Table 3.8: Comparison of three sparsified-SVD-based methods with other methods on WBC.

Methods	Data Value Distortion					Accuracy%
	RE	RP	RK	CP	CK	SVMlight
WBC						96.4
uniformNoise	0.1085	219.6993	0.0130	0	1	96.4
normalNoise	0.1098	224.8148	0.0084	0	1	96.3
thinSVD	0.1222	228.8972	0.0114	0.2222	0.7778	96.4
s-SVD	1.2662	228.1370	0.0013	3.3333	0	96.6
c-SVD	1.2702	230.1561	0.0021	3.3333	0	96.4
e-SVD	1.2704	228.0744	0.0014	3.3333	0	96.4

Among the three sparsification strategies, no significant difference exists on distortion level and data utility. Especially it shows that they have the same effect on changing rank of attributes with the same CP and CK values. It is obvious that sparsification increases data privacy level by making all the attributes change their rank in average value because the CK value is 0. As to the SVMlight classification accuracy, five methods achieve a level not worse than that attained with the original dataset, normal noise-additive method is slightly worse.

3.5 Sparsified SVD-based Structural Partition Schemes

Instead of conducting the thin SVD and the proposed sparsification strategies on the whole data matrix, structural matrix partition is used here to divide the original matrix into several submatrices, and we perform the sparsified SVD on one selected submatrix. Three kinds of matrix partition schemes are proposed here, which are denoted by P1, P2, and P3, respectively.

3.5.1 Three partition schemes

1. **Subject-based Partition Scheme (P1).** We denote the subject-based partition scheme by P1. Let us partition A as

$$A = \begin{bmatrix} A(1) \\ A(2) \end{bmatrix} \tag{3.19}$$

The whole dataset is divided into two groups, $A(1)$ and $A(2)$. We perform the sparsified SVD on $A(1)$ to get $B(1) =$s-SVD$(A(1))$. Then, the partially distorted dataset is

$$\widetilde{A} = \begin{bmatrix} B(1) \\ A(2) \end{bmatrix} \tag{3.20}$$

Here, all attribute values of the first group are distorted.

2. **Attribute-based Partition Scheme (P2).** We use P2 to denote the attribute-based partition scheme.

Let

$$A = [\ A(1) \quad A(2)\ ] \tag{3.21}$$

$A(1)$ contains the first part of the attribute items and $A(2)$ the second part. We perform the sparsified SVD on $A(1)$ to get $B(1)$ =s-SVD$(A(1))$. Then the new distorted matrix is

$$\widetilde{A} = [\ B(1) \quad A(2)\ ] \tag{3.22}$$

In this case, only one part of the attribute values is distorted by SSVD.

3. **Two-dimensional Partition Scheme (P3).** The two-dimensional partition scheme is denoted by P3.

   Let the partition be

$$A = \begin{bmatrix} A(1) & A(2) \\ A(3) & A(4) \end{bmatrix} \tag{3.23}$$

   We perform sparsified SVD on $A(1)$ to get $B(1) =$s-SVD$(A(1))$. Then, the selectively distorted matrix is

$$\widetilde{A} = \begin{bmatrix} B(1) & A(2) \\ A(3) & A(4) \end{bmatrix} \tag{3.24}$$

   Here, a part of the attribute values for one part of the subjects is selected for distortion operation.

The levels of the data value and pattern distortion are dependent on the partition scheme in use. Depending on specific goals of the various applications, one of the above three schemes can be chosen. The analysis of the proposed strategies will be performed in the next sections.

SVD computation incurs a significant computational cost for large scale data matrices. The cost of computing the SVD of a sparse matrix A using a Lanczos-type procedure can be expressed:

$$\texttt{Total cost} = I \times \texttt{cost}(A^T Ax) + k \times \texttt{cost}(Ax),$$

where I is the number of iterations required by a Lanczos-type procedure to approximate the eigensystem of $A^T A$, x is a vector and k is the number of computed singular values and their corresponding number of nonzero entries in the sparse matrix A. The dominant computational cost of the Lanczos method is related to the number and complexity of the matrix multiplications by A and A^T.

Computing SVD only on one part of the original matrix would lead to a reduction on the computational cost and an improvement on the efficiency of data mining algorithms by removing unnecessary data distortion. This is because that the matrix multiplication is now performed with respect to the submatrix $A(1)$, not to the full matrix A.

3.5.2 Experimental Evaluation

A synthetic dataset, called ORG, a $[2000 \times 100]$ matrix is generated to represent a dataset with 2000 subjects and 100 attributes. Its entries are randomly and independently generated from a uniform distribution on the interval $[1, 10]$. We classify all the subjects into two classes using the following rule:

$$\text{class label} = \begin{cases} 1 & \text{if } |\sin(\text{ORG}(i,1)) \text{ - ORG}(i,88)| * |\cos(\text{ORG}(i,45))| \\ & \text{*ORG}(i,78) > 15; \\ -1 & \text{otherwise.} \end{cases}$$

The class labels are $+1$ and -1. SVM classification is used to learn from the synthetic dataset and build a classifier model. The classification results are obtained by a 5-fold cross validation.

1. **Sensitivity of classification accuracy to threshold value ϵ in s-SVD.** Here we examine the influence of the threshold value, ϵ, in the STS of s-SVD. Figure 3.11 illustrates the classification accuracy under ϵ in the interval from 0 to 0.1. In the experiment, the approximation rank of the thin SVD is 40. With the increment of ϵ in s-SVD, it exhibits no observable trend in data utility for all three distortion schemes. This implies that the sparsification parameter ϵ does not affect the classification accuracy sensitively in this study.

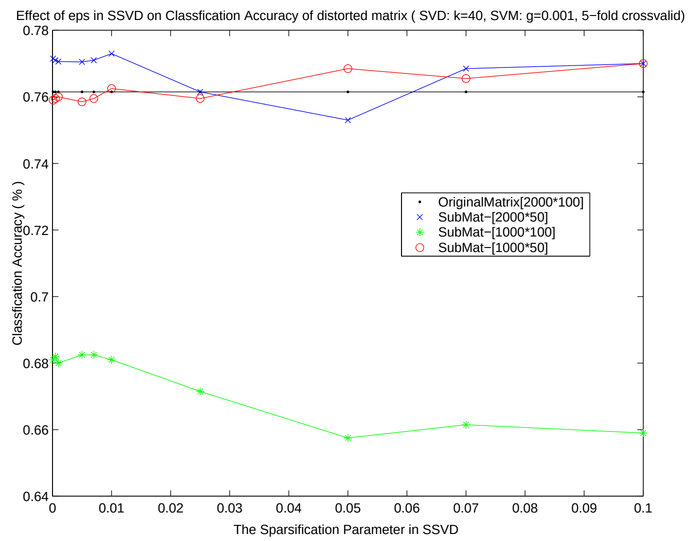


Figure 3.11: The effect of the threshold value ϵ in s-SVD on SVMlight accuracy

2. **Comparison of the five modification methods.** The five data modification methods, uniformly distributed noise (UD), normally distributed noise (ND), SVD, s-SVD, s-SVD with matrix partition, are implemented on ORG to compare the performances. Table 3.9 shows the comparison among these five data modification methods. The rank k in the SVD is 20. SVM classification is used to learn from ORG dataset and build a classifier model. The classification results are obtained by a 5-fold cross validation.

Table 3.9: Comparison of five modification methods on ORG.

Methods	*Level of Distortion*					*Accuracy%*
	RE	RP	RK	CP	CK	
ORG						76.15
UD	0.0760	664.0489	0.0062	0	1	76.20
ND	0.0758	665.1643	0.0043	0	1	75.80
SVD	0.3665	666.9214	0.0007	21.28	0.39	76.60
s-SVD	0.7464	664.0129	0.0005	36.42	0	76.50
s-SVD[P1]	0.5059	667.5759	0.0011	34.02	0.02	66.75
s-SVD[P2]	0.4866	332.7783	0.5002	35.48	0	77.35
s-SVD[P3]	0.3655	333.8874	0.5007	34.44	0	76.70

Based on the comparison results in Table 3.9, a conclusion can be made that, compared to the randomization-based data distortion methods such as UD and ND, thin SVD-based strategies achieve a higher level of distortion and can provide better protection on privacy. Sparsified SVD is better than thin SVD on three of the five metrics. The CK value for the s-SVD-based methods is near or equal to 0, which means all the attributes change their ranks in average value after performing certain data transformations.

Among the three proposed matrix partition strategies, for s-SVD[P1] and s-SVD[P2], the selected submatrices for sparsified SVD have the same size. s-SVD[P2] and s-SVD[P3] are comparable on the distortion level with the largest RK value and the lowest RP value. All these three methods greatly affect attribute ranks.

As to mining accuracy, the accuracies of the three new schemes are 66.75%, 77.35% and 76.7%. Naturally s-SVD[P1] is worst on data mining accuracy, due to its best preservation of privacy. s-SVD[P2] supplies the best data utility with a higher accuracy than the original dataset. From the above analysis, we can make a reasonable conclusion that, considering a trade-off between privacy preservation and data utility, the performance of s-SVD[P2] is the best among these three matrix partition strategies.

3. **Sensitivity of data value distortion to the choice of approximation rank of SVD.** To examine the change of data quality of the three partition schemes with the increasing rank of SVD, we conduct more experiments on ORG. Figure 3.12 illustrates the influence of rank of SVD on classification accuracy. P2 and P3 show similar graphs of accuracy. The accuracy tends to decrease with k till k is larger than a half of the number of attributes, 50 in our experiment. For any $k > 50$, the accuracy of P1 and P2 is equal to that of the original dataset. The highest accuracy is obtained with the rank of 1/10 of the number of attributes.

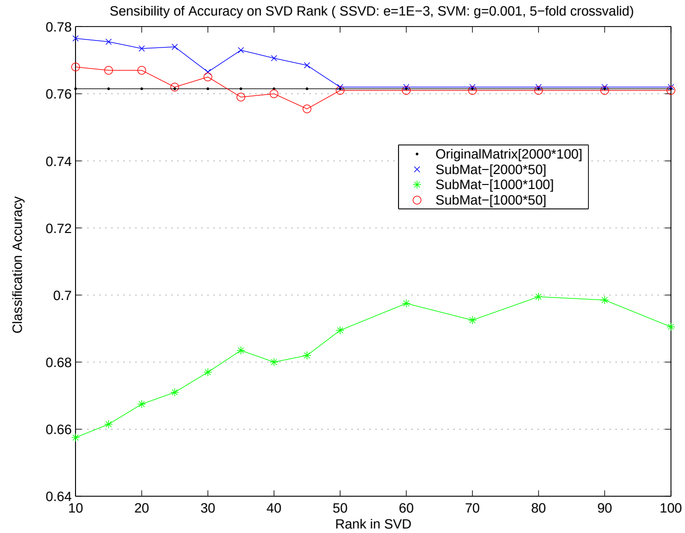


Figure 3.12: Accuracy by using s-SVD (s-SVD: $\epsilon = 1E-3$, SVM: $g = 0.001$, 5-fold cross validation).

P1 shows worse performance on data utility than P2 and P3 and its accuracy is lower than that of the original dataset. It also demonstrates a different trend of change. The accuracy of P1 increases with k when $k < 60$ and decreases with k for $k > 60$.

How to choose the rank of SVD is still unsolved and empirical tests are required. Our experiment implies one possible good choice of the rank of SVD for our distortion strategies if only considering data utility. If P1 scheme is used, 3/5 of the number of attributes is a good choice for k. For P2 and P3, we can choose $1/10$ of the number of attributes as the rank of SVD.

4. **Attribute size sensitivity in attribute-based partition.** The previous experiments on the synthetic dataset demonstrate that attribute-based partition scheme can provide a high mining accuracy with an acceptable level of data distortion. The further test on this partition scheme is implemented from the viewpoint of both data distortion and data utility. It shows an intuitive result that the level of distortion increases with the number of attributes in $A(1)$. Figure 3.13 exhibits a critical point with the highest accuracy when the number of columns in $A(1)$ is 70, which means

$A(1)$ contains 70 percent of the attributes.

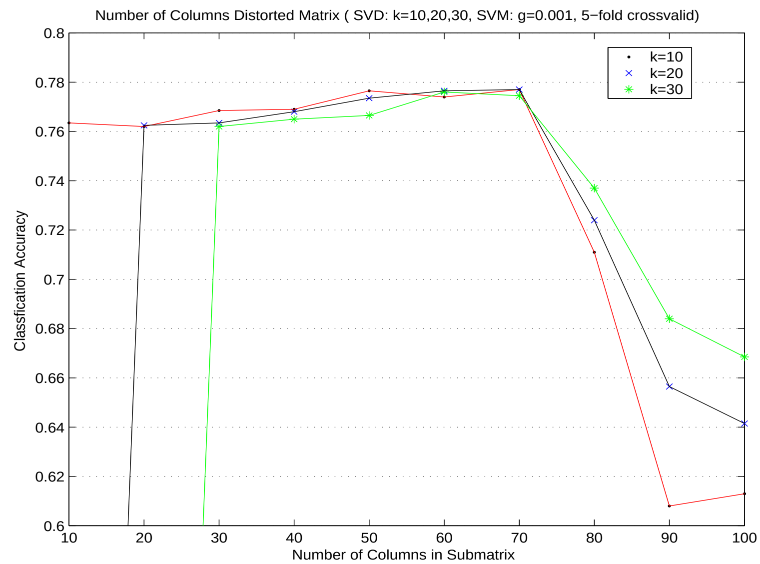


Figure 3.13: The effect of the number of attributes on accuracy of attribute-based partition.

5. **Computation Time.** The CPU time used to compute the SVD and partial SVD of the data set on a SunBlade 150 workstation is 46.12 seconds for s-SVD, 13.27 seconds for s-SVD[P1], 22.95 seconds for s-SVD[P2], and 5.07 seconds for s-SVD[P3].

6. **Comparison of three partition schemes on WBC.** We choose three target submatrices as 467 by 9 in P1, 699 by 6 in P2, and 600 by 7 in P3. Therefore, the number of entries in each submatrix is almost the same as 4200 in order to make our evaluation fair on three schemes. The rank k of SVD is 3. Table 3.10 summarizes a performance evaluation on three sparsified SVD methods.

   For data value distortion, P1 has the highest RE and RP with the smallest RK, which means that P1 supplies the best protection on elements. P3 has the best protection on average values of attributes with the highest CP and lowest CK. For the WBC data set, P2 does not perform very well on privacy protection.

   For data mining accuracy, all three schemes are better than the original dataset. P3 achieves the highest accuracy up to 97%. P2 is better than P1.

Table 3.10: Comparison of three partition schemes on WBC

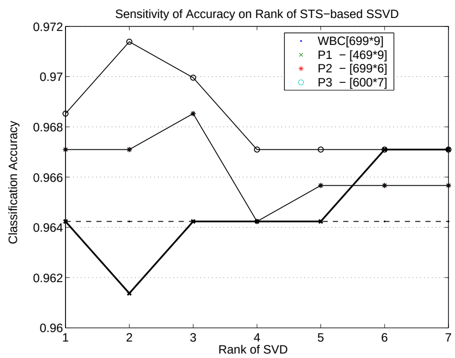
Methods		Level of Distortion					Accuracy%
		RE	RP	RK	CP	CK	
WBC		-	-	-	-	-	96.4
SVD		0.2846	238.1218	0.0052	1.5556	0.5556	96.6
s-SVD		1.2556	230.8523	0.0021	0.3283	0	96.7
P1	Ps-SVD	1.1443	214.4181	0.0296	1.7778	0.2222	96.4
	Pc-SVD	1.1468	212.7525	0.0299			96.6
	Pe-SVD	1.1470	213.7399	0.0297			96.6
P2	Ps-SVD	0.9632	152.7821	0.3351	2.8889	0.2222	96.9
	Pc-SVD	0.9738	150.6559	0.3357			96.7
	Pe-SVD	0.9632	152.6759	0.3352			96.9
P3	Ps-SVD	1.0492	180.2636	0.2287	3.1111	0	97.0
	Pc-SVD	1.0574	181.4964	0.2291			97.0
	Pe-SVD	1.0492	180.1682	0.2287			97.0

Parameters: $k = 3$; Submatrix size: P1:$[467 * 9]$ P2:$[699 * 6]$ P3:$[600 * 7]$

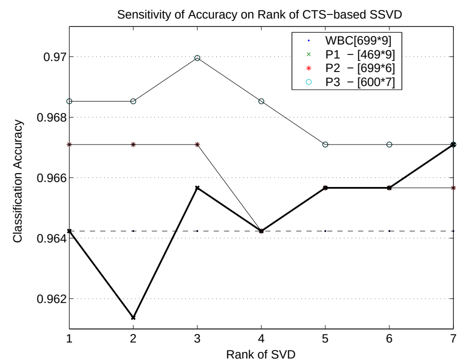
7. **Sensitivity of mining accuracy to the approximation rank of SVD.** As stated earlier, the optimal value of rank of SVD is dependent on specific applications and chosen mostly by empirical tests. But a general impact tendency of rank on data quality would provide good recommendations on rank determination. Figures 3.14(a), 3.14(b) and 3.14(c) indicate the existence of such a general tendency and a critical point, which is consistent with the result from Experiment 3 on the synthetic dataset.

   Data quality level in the descending order is P3, P2 and P1. P2 and P3 behave similarly on accuracy and the highest level accuracy can be obtained at some k less than or equal to $1/3$ of the number of attributes. After this peak point, accuracy decreases with k.

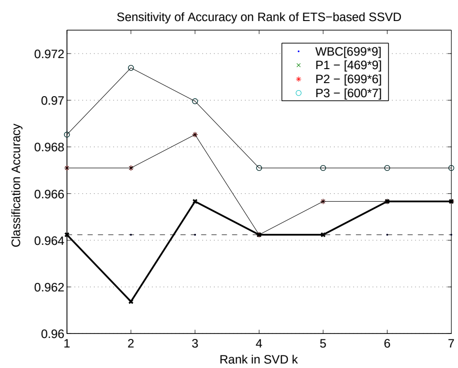
   P1 shows worse performance on data utility than P2 and P3, and when $k = 2$, its accuracy is lower than the original. It also demonstrates a different trend of change. Its accuracy increases with k when k is greater than a turning point which is close to a half of the number of attributes, 4 or 5 in WBC. No observable impact of different sparsification strategies on accuracy is exhibited in this experiment. Taking computational cost into consideration, STS is a better choice for P3.



(a) Sensitivity of accuracy to approximation rank of thin SVD by selective s-SVD



(b) Sensitivity of accuracy to approximation rank of thin SVD by selective c-SVD



(c) Sensitivity of accuracy to approximation rank of thin SVD by selective e-SVD

Figure 3.14: Sensitivity of mining accuracy to the approximation rank of thin SVD for selective sparsified SVD-based Methods.

3.6 Summary

The foregoing experimental evaluation reveals that the proposed hybrid approach provides better performance both on data distortion and data utility. Some important conclusions can be drawn from these experiments:

1. The overall performance of the SVD-based distortion approaches is better than the data perturbation approaches.

2. Most of the SVD-based approaches can achieve a higher accuracy on classification

than the original data.

3. Sparsified SVD-based approaches are better than SVD-based ones on data distortion level without any loss of data utility, along with a further improvement on reducing computational cost due to the SVD manipulation.

4. Three sparsification strategies have nearly identical effects on data distortion and utility level in our experiments. Compared to all the other methods in the study, all of the three exhibit much better privacy protection on average values of attributes. With respect to the computational cost, STS is a desirable choice.

5. For attribute-based partition and two-dimensional partition distortion strategies, the classification accuracy decreases with the increment of the rank of SVD after reaching a peak value at certain rank less than or equal to 1/3 of the number of attributes. This inherent property lends itself well for achieving a high accuracy with a significant reduction on computational cost due to the use of a small rank value.

6. The overall performance of the three structural partition strategies is as follows:

   - Object-based partition has the highest distortion level on elements of datasets.
   - Two-dimensional partition provides the most satisfactory protection on average values of attributes.
   - All three schemes provide a satisfactory level of data utility.
   - Attribute-based and two-dimensional based schemes display quite comparable classification accuracy. Object-based scheme has the lowest data utility level among the three.

Of course, which partition strategy to use in a particular application is dependent on the circumstances of that application such as the nature of the database. With respect to the specific requirements of data administrators and characteristics of target datasets, we believe that the above conclusions from our experiments would provide data miners with a good recommendation on finding a desirable solution with a reasonable compromise on privacy protection, utility of data and computational cost.

Chapter 4

NMF-based Data Hiding Strategy

The previous section shows that SVD is a good solution for data privacy protection with competitive data mining accuracy. However, a drawback is associated with the extraction of singular vectors of orthogonal decompositions. If the underlying data only consists of nonoverlapping, *i.e.*, orthogonal patterns, SVD performs very well. If patterns with similar strengths overlap, attributes contained in some of the previously discovered patterns are extracted from each pattern. In orthogonalizing the second vector with respect to the first vector, SVD introduces negative values into the second vector. Since most real-world databases have nonnegative attribute values, there is no easy interpretation of these negative values in the context of most data mining activities, and negative components contradict physical realities.

Considering the nonnegative-valued characteristic of most datasets, a nonorthogonal decomposition that does not introduce negative values into the vector components may be desirable. In this section, nonnegative matrix factorization (NMF) will be used to distort the original dataset A.

NMF is a matrix decomposition method to obtain a representation of data using non-negative constraints. These constraints can lead to a part-based representation because they allow only additive, not subtractive, combinations of the original data. This is in contrast to techniques for finding a reduced rank representation based on SVD. As an unsupervised learning method for uncovering latent features in high-dimensional data, NMF can be used to preserve natural data nonnegativity and avoid subtractive basis vector and encoding interactions. The overall performance of NMF on distortion level and data mining accuracy will be compared to our previously proposed SVD-based data hiding strategies and other existing popular data perturbation methods.

4.1 Nonnegative Matrix Factorization (NMF)

Given a nonnegative matrix $A \in \mathbb{R}_+^{n \times m}$ with $A_{ij} \geq 0$ and a pre-specified positive integer $K < \min\{n, m\}$, NMF finds two nonnegative matrices $H \in \mathbb{R}_+^{n \times K}$ with $H_{ik} \geq 0$ and $W \in \mathbb{R}_+^{K \times m}$ with $W_{kj} \geq 0$ so that $A \approx HW$ minimizes the objective function

$$f(H, W) = \frac{1}{2} \|A - HW\|_F^2 . \tag{4.1}$$

The usual way to find H and W is by the following least-squares optimization, which minimizes the difference between A and HW:

$$\min_{H,W} f(A, H, W) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - (HW)_{ij})^2 \tag{4.2}$$

$$\text{subject to} \quad \begin{array}{l} H_{ia} \geq 0, \\ W_{bj} \geq 0, \quad \forall\, i,\ a,\ b,\ j. \end{array}$$

In optimization, upper- and lower-bounding variables is referred to as imposing bound constraints. Hence (4.2) is a standard bound-constrained optimization problem. There are several methods to solve (4.2) in the literature. Algorithms designed to approximate A generally begin with initial estimates of the matrices H and W, followed by alternating iterations to improve these estimates.

In our NMF-based data hiding methods, let the original dataset T be encoded by a vector space data model A. Using some NMF algorithm, A can be decomposed into two nonnegative factor matrices. It can be stated as a transformation from A to $\widetilde{A}$ defined as follows: *Given a nonnegative data model $A(n \times m)$, find two nonnegative matrices $H \in \mathbb{R}_+^{n \times K}$ and $W \in \mathbb{R}_+^{K \times m}$ with K being the number of clusters in A, that minimizes $\mathcal{Q}$, where $\mathcal{Q}$ is an objective function defining the nearness between the matrices A and HW.*

The modified version of A is denoted as $\widetilde{A} = HW$.

The choice of the objective function $\mathcal{Q}$ affects the solution of $\widetilde{A}$. Here, the Euclidean distance or the Frobenius norm is chosen as they are popular in matrix computations,

$$\min_{H \in \mathbb{R}_+^{n \times K}, W \in \mathbb{R}_+^{K \times m}} \mathcal{Q} = \|A - HW\|_F^2 . \tag{4.3}$$

Now we do reduction on $\mathcal{Q}$

$$\begin{array}{rcl} \mathcal{Q} & = & \|A - HW\|_F^2 \\ & = & tr((A - HW)^T (A - HW)) \\ & = & tr(A^T A - A^T HW - W^T H^T A + W^T H^T HW) \\ & = & tr(A^T A) - 2tr(A^T HW) + tr(W^T H^T HW) \end{array} . \tag{4.4}$$

4.2 Algorithms of Nonnegative Matrix Factorization

NMF can be viewed as a minimization problem with bound constraints. There are several algorithms to compute submatrices H and W. In our study, since we use the transposed form of the general NMF, these algorithms are modified according to our definition of matrix model of the data set in which the rows of A represent the data points. Therefore, H here is equal to H^T in general NMFs and W here is equal to W^T as well. We also modify the formal description of the following algorithms, however, the basic idea of these algorithms is maintained.

In (4.4), the objective function is

$$\mathcal{Q} = tr(A^T A) - 2tr(A^T HW) + tr(W^T H^T HW).$$

Taking the gradients of $\mathcal{Q}$ consists of two parts which are the partial derivatives with respect to H and W, respectively:

$$\begin{array}{rcl} \nabla_H \mathcal{Q} & = & \dfrac{\partial \mathcal{Q}}{\partial H} \\ & = & -2\dfrac{\partial tr(A^T HW)}{\partial H} + \dfrac{\partial tr(W^T H^T HW)}{\partial H} \\ & = & -2AW^T + 2HWW^T, \end{array} \tag{4.5}$$

$$\begin{aligned}\nabla_W \mathcal{Q} &= \frac{\partial \mathcal{Q}}{\partial W} \\ &= -2\frac{\partial tr(A^T HW)}{\partial W} + \frac{\partial tr(W^T H^T HW)}{\partial W} \\ &= -2H^T A + 2H^T HW.\end{aligned} \tag{4.6}$$

The optimal solution $(\mathbf{H}, \mathbf{W})$ makes $\partial \mathcal{Q}/\partial H = 0$ and $\partial \mathcal{Q}/\partial W = 0$. Hence,

$$A\mathbf{W}^T \oslash \mathbf{HWW}^T = I, \tag{4.7}$$

$$\mathbf{H}^T A \oslash \mathbf{H}^T\mathbf{HW} = I, \tag{4.8}$$

where $\oslash$ denotes element-wise division, I denotes an identity matrix.

4.2.1 Multiplicative Update Algorithm

The most popular approach is multiplicative updates proposed by Lee and Seung [47][48]. At each iteration of this method, the elements of H and W are multiplied by certain factors which are from (4.7) and (4.8). The update rules are

$$H_{ij} \leftarrow H_{ij}\frac{[AW^T]_{ij}}{[HWW^T]_{ij}}, \tag{4.9}$$

$$W_{ij} \leftarrow W_{ij}\frac{[H^T A]_{ij}}{[H^T HW]_{ij}}. \tag{4.10}$$

In the matrix notation, the above updates become:

$$H \leftarrow H \odot AW^T \oslash HWW^T, \tag{4.11}$$

$$W \leftarrow W \odot H^T A \oslash H^T HW, \tag{4.12}$$

where $\odot$ denotes element-wise multiplication. The nonnegativity of H and W is maintained in the updates. Lee and Seung proved that under the above update rules the Frobenius norm of $(A - HW)$ is monotonically non-increasing [48].

Table 4.1: Algorithm 3: Multiplicative update algorithm (transposed version: $\mathbf{A} = \mathbf{HW}$).

Algorithm 3 Multiplicative Update (Transposed Version: $\mathbf{A} = \mathbf{HW}$)

Input: $A \in \mathbb{R}_{+}^{n\times m}$, $0 < K \ll \min(n, m)$, and maxIter.
Output: $H \in \mathbb{R}_{+}^{n\times K}$, $W \in \mathbb{R}_{+}^{K\times m}$.

```
Randomly create the initial estimates for H and W:
   H_ij^(0) ⇐ nonnegative value, 1 ≤ i ≤ n, 1 ≤ j ≤ K.
   W_ij^(0) ⇐ nonnegative value, 1 ≤ i ≤ K, 1 ≤ j ≤ m.
Scale rows of W to unit length.
for p = 1 to maxIter do
   for k = 1 to K do
      H_ik ← H_ik [AW^T]_ik / ([HWW^T]_ik + 10^-9),  1 ≤ i ≤ n;
      W_kj ← W_kj [H^T A]_kj / ([H^T HW]_kj + 10^-9),  1 ≤ j ≤ m;
      Scale rows of W to unit length
   end
   if converge then
      Output H^(p) and W^(p);
      break
   end
end
```

where the updates are

$$H_{ik} \leftarrow H_{ik}\frac{[AW^T]_{ik}}{[HWW^T]_{ik} + 10^{-9}}, \quad 1 \le i \le n;$$

$$W_{kj} \leftarrow W_{kj}\frac{[H^TA]_{kj}}{[H^THW]_{kj} + 10^{-9}}, \quad 1 \le j \le m;$$

After updating a column of H, we update the corresponding row of W. A small positive value is added into the denominators of the updates to prevent a division by zero, for which we use 10^{-9}. At each iteration, the rows of W are normalized to sum to one. It is simple to implement. The nonnegativity of W and H is guaranteed in the iterations, since at each iteration, the entries of the two matrices are multiplied by nonnegative factors. For the zero entries in the initial estimates, there is no update and they remain zero. That brings out one drawback of the multiplicative algorithm, which is that once an entry in W or H becomes 0, it must remain 0. This locking of 0 entries is restrictive, meaning that once the algorithm starts heading down a path towards a fixed point, even if it is a poor fixed point, it must continue in that direction [12].

As far as the computational cost, each iteration requires six $O(nmK)$ matrix-matrix multiplications and six $O(n^2)$ element-wise operations. Due to the fact that the multiplicative update is the first well-known NMF algorithm, it has become a baseline against which the newer algorithms are compared. This algorithm is notoriously slow to converge [12]. It requires many more iterations than alternatives methods described below.

4.2.2 Alternating Nonnegative Least-squares Using Projected Gradients

One class of NMF algorithms is the *alternating least-squares* (ALS) methods. We refer to this class of algorithms simply by ALS. A least-squares step is followed by another least-squares step in an alternating way. ALS algorithms were first used by Paatero in [63]. ALS algorithms exploit the fact that, while the optimization problem of (4.3) is not convex in both W or H, it is convex in either W and H. Thus, given one matrix, another matrix can be found with the simple least-squares computations. An elementary ALS algorithm follows in Algorithm 4 of Table 4.2. The least-squares computation might generate negative entries in W and H. A simple strategy is used to set all negative entries to 0, or set all entries, which are less than a predefined positive number ϵ, to ϵ. This strategy adds sparsity and additional flexibility not available in the multiplicative update method.

Depending on the implementation as in Algorithm 4, ALS algorithm can be very fast. It requires slightly less work than an SVD implementation. Improvements to the basic ALS algorithm include incorporation of sparsity and nonnegativity constraints such as those described later.

Table 4.2: Algorithm 4: Basic ALS algorithm for NMF (transpose version: $\mathbf{A} = \mathbf{HW}$).

Algorithm 4 Basic ALS algorithm for NMF (Transpose Version: $\mathbf{A} = \mathbf{HW}$)

Input: $A \in \mathbb{R}_+^{n\times m}$, $0 < K \ll \min(n,m)$, maxIter, and a very small positive number ϵ.

Output: $H \in \mathbb{R}_+^{n\times K}$, $W \in \mathbb{R}_+^{K\times m}$.

begin
- randomly create an initial estimate for H:
  - $H_{ij}^{(0)} \Leftarrow$ nonnegative value, $1 \le i \le n, 1 \le j \le K$.
- **for** *$p = 0$ to maxIter* **do**
  - # solve $W^{(p+1)}$ in matrix equation $H^T HW = H^T A$ with $H^{(p)}$
    - $W^{(p+1)} \longleftarrow arg \min_{W\in\mathbb{R}_+^{K\times m}} (H^T HW - H^T A)$.
  - scale rows of W to unit length.
  - # set all entries W_{kj} in W to $\max(\epsilon, W_{kj})$.
  - **for** *each entry in W, W_{kj}* **do**
    - $W_{kj} \longleftarrow \max(\epsilon, W_{kj})$
  - **end**
  - # solve $H^{(p+1)}$ in matrix equation $HWW^T = AW^T$ with $W^{(p+1)}$
    - $H^{(p+1)} \longleftarrow arg \min_{H\in\mathbb{R}_+^{n\times K}} (HWW^T - AW^T)$.
  - # set all entries H_{ik} in H to $\max(\epsilon, H_{ik})$.
  - **for** *each entry in H, H_{ik}* **do**
    - $H_{ik} \longleftarrow \max(\epsilon, H_{ik})$
  - **end**
  - **if** *converge* **then**
    - output $H^{(p+1)}$ and $W^{(p+1)}$;
    - break
  - **end**
- **end**

end

Lin [51] proposed a method for NMF by applying a projected gradient method to solve the nonnegative least-squares problem. Consider the following standard form of bound-constrained optimization problems:

$$\min_{x\in\mathcal{R}^n} \quad f(x)$$
$$\text{subject to} \quad l_i \le x_i \le u_i, i = 1, \dots, n,$$

where $f(x) : \mathcal{R}^n \rightarrow \mathcal{R}$ is a continuously differentiable function, and l_i and u_i are lower and upper bounds, respectively. Assume k is the index of iterations. Projected gradient methods update the current solution $x^{(k)}$ to $x^{(k+1)}$ by the following rule:

$$x^{(k+1)} = P[x^{(k)} - \alpha_k \nabla f(x^{(k)})],$$

where

$$P[x_i] = \begin{cases} x_i & \text{if } l_i \le x_i \le u_i, \\ u_i & \text{if } x_i \ge u_i, \\ l_i & \text{if } x_i \le l_i, \end{cases}$$

maps a point back to the bounded feasible region. Variants of projected gradient methods differ on selecting the step size α_k. The most used condition in projected gradient methods is

$$f(x^{(k+1)}) - f(x^{(k)}) \le \sigma \nabla f(x^{(k)})(x^{(k+1)} - x^{(k)}), \tag{4.13}$$

which ensures the sufficient decrease of the function value per iteration. An improved projected gradient method as Algorithm 5 is described in Table 4.3. The common choice of σ is 0.01, and we consider $\beta = 0.1$ here.

Searching $\alpha_k = \beta^{t_k}$ is the most time consuming operation in Algorithm 5, so one should check as few step sizes as possible. Since $\alpha_{(k-1)}$ and α_k may be similar, $\alpha_{(k-1)}$ is used as the initial guess and then either increases or decreases it in order to find the largest β^{t_k} satisfying the condition (4.13).

This method leads to faster convergence than the popular multiplicative update method, and the overall computational cost is

$$\text{iter} \times (O(nmk) + \text{subIter} \times O(tmk^2 + tnk^2)),$$

where subIter is the number of sub-problem iterations, k is the rank of NMF.

Table 4.3: Algorithm 5: An improved projected gradient method.

Algorithm 5 An improved projected gradient method.

```
Input:  0 < β < 1, 0 < σ < 1, f, ∇f.
begin
    initialize any feasible x^(1) and set α_0 = 1
    for k = 1, 2, ..., do
        assign α_k ← α_{k-1}
        x^(k+1) = P[x^(k) − α_k ∇f(x^(k))]
        if f(x^(k+1)) − f(x^(k)) ≤ σ∇f(x^(k))(x^(k+1) − x^(k)) then
            while f(x^(k+1)) − f(x^(k)) ≤ σ∇f(x^(k))(x^(k+1) − x^(k))||x(α_k/β) ≠ x(α_k) do
                α_k ← α_k/β
            end
        end
        else
            while f(x^(k+1)) − f(x^(k)) ≥ σ∇f(x^(k))(x^(k+1) − x^(k)) do
                α_k ← α_k β
            end
        end
        Set x^(k+1) = P[x^(k) − α_k ∇f(x^(k))].
    end
end
```

4.2.3 Incorporating Additional Constraints

The objective function $\mathcal{Q}$ in (4.3) is sometimes extended to include auxiliary constraints on H and/or W. This is often done to compensate for uncertainties in the data, to enforce desired characteristics in the computed solution, or to impose prior knowledge about the application at hand. *Penalty terms* are usually used to enforce auxiliary constraints, extending the objective function $\mathcal{Q}$ as follows:

$$\min_{H\in\mathbb{R}_+^{n\times K}, W\in\mathbb{R}_+^{K\times m}} \mathcal{Q} = \|A - HW\|_F^2 + \alpha J_1(H) + \beta J_2(W). \tag{4.14}$$

Here α and β are small positive regularization parameters that balance the trade-off between the approximation error and the constraints. The functions $J_1(H)$ and $J_2(W)$ are nonnegative penalty terms to enforce certain constraints.

The regularization terms $J_1(H)$ and $J_2(W)$ can be defined in many ways. The usual constraints are *sparsity* and *smoothness*. Let us assume the following definition for L_p-

norm of a given matrix $C \in \mathcal{R}^{n\times m}$: [18]

$$L_p(C) = \left(\sum_{i=1}^{n}\sum_{j=1}^{m}|C_{ij}|^p\right)^{\frac{1}{p}}.$$

1. **Sparse solution.** The notion of sparsity refers sometimes to a representation where only a few attributes are effectively used to represent data. Measures for sparsity include, the L_p norms for $0 < p \leq 1$, and Hoyer's measure.

   If L_1 norm is used, then penalty terms can be defined as follows [18]:

$$J_1(H) = \sum_{i=1}^{n}\sum_{k=1}^{K} H_{ik}, \tag{4.15}$$

$$\nabla_H J_1(H) = 1. \tag{4.16}$$

$$J_2(W) = \sum_{k=1}^{K}\sum_{j=1}^{m} W_{kj}, \tag{4.17}$$

$$\nabla_W J_2(W) = 1. \tag{4.18}$$

   By Hoyer's measure, sparseness$(x) = \frac{\sqrt{n}-\|x\|_1/\|x\|_2}{\sqrt{n}-1}$, where n is the number of elements. The penalty term could be

$$J_1(H) = (\omega\|\text{vec}(H)\|_2 - \|\text{vec}(H)\|_1)^2, \tag{4.19}$$

   where $\omega = \sqrt{nK} - (\sqrt{nK} - 1)\gamma$ and vec(.) is the vec operator that transforms a matrix into a vector by stacking its columns. The desired sparseness in H is specified by setting γ to a value from 0 to 1 [12].

2. **Smooth solution.** Smoothness constraints are often enforced to regularize the computed solutions in the presence of noise in the data. The L_2 norm penalizes the solu-

tions of large Frobenius norm [18], thus

$$\begin{aligned} J_1(H) &= \sum_{i=1}^{n}\sum_{k=1}^{K} H_{ik}^2 = \mathbf{tr}(HH^T), && (4.20)\\ \nabla_H J_1(H) &= 2H. && (4.21)\\ J_2(W) &= \sum_{k=1}^{K}\sum_{j=1}^{m} W_{kj}^2 = \mathbf{tr}(WW^T), && (4.22)\\ \nabla_W J_2(W) &= 2W. && (4.23) \end{aligned}$$

4.3 NMF-based Data Modification Method

In this section, we describe a basic data factorization scheme that performs nonnegative matrix factorization (NMF) on the original data set, which is the essential part for our NMF-based data modification.

4.3.1 Basic Data Factorization Scheme

Alternating nonnegative least squares using projected gradients for NMF is used in our implementation. It generally begins with initial estimates of the matrices H and W, followed by alternating iterations to improve these estimates. After performing basic data factorization on A, the modified data set is $\widetilde{A} = HW$, where

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix}, \qquad W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_K \end{bmatrix}.$$

$$H_i = (h_{i1}\, h_{i2}\, \ldots\, h_{is}\, \ldots\, h_{iK}), \qquad i = 1, 2, \ldots, n.$$

$$W_j = (w_{j1}\, w_{j2}\, \ldots\, w_{jt}\, \ldots\, w_{jm}), \qquad j = 1, 2, \ldots, K.$$

4.3.2 Data Hiding Scheme

Based on the basic data factorization scheme, data hiding can be easily fulfilled with some simple preprocessing procedure on the original data matrix A. The nonnegative property of A needs to be validated by checking the nonnegativity of all entries. Most real-world data sets have nonnegative entries. If A has negative entries, its values can be shifted column-wise and then normalized. After this process, $\widetilde{A}$ can be generated with the basic data factorization scheme. The algorithm is illustrated as Algorithm 6 in Table 4.4.

Table 4.4: Algorithm 6: Data hiding scheme.

Algorithm 6 Data hiding scheme.

Input: a data set $A \in \mathbb{R}^{n \times m}$, the number of classes K, $0 < K \ll \min(n, m)$.

Output: a modified version $\widetilde{A}$, two factor matrices: $H \in \mathbb{R}_+^{n \times K}$ and $W \in \mathbb{R}_+^{K \times m}$.

begin
    NonNeg = 1;
    foreach A_{ij} **do**
        if $A_{ij} < 0$ **then**
            NonNeg = 0.
        end
    end
    if *NonNeg == 0* **then**
        do nonnegativity normalization on A;
        Compute H and W;
        Calculate $\widetilde{A} = HW$.
    end
end

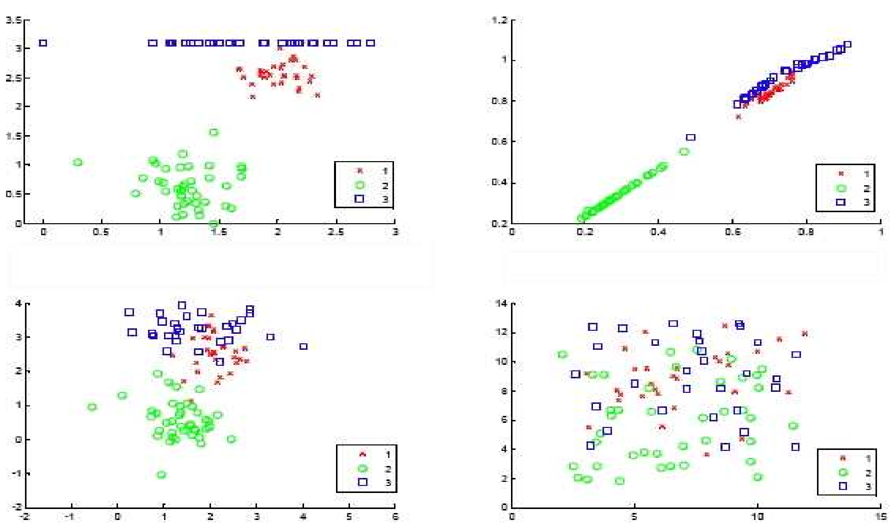


Figure 4.1: A 2D synthetic dataset with 3 classes and its modified version from NMF are in the upper two subfigures. The bottom two subfigures show modified data using the two noise-additive methods.

The performance of this scheme is illustrated in Figures 4.1 and 4.2. Figure 4.1 shows the data distributions of a dataset and its modified versions from NMF and two noise-additive methods. The dataset is synthetically created from three bivariate Gaussian distributions and normalized to a nonnegative matrix. It has 100 subjects, each of which has 2 features. Three classes are depicted with three different symbols. The modified version in the upper right subfigure is calculated from an NMF operation with $K = 3$. The lower two subfigures show modified datasets generated from adding normally distributed noise and uniformly distributed noise, respectively. It is clearly observable that the data distributions from NMF and the addition of uniformly distributed noise (lower right) are distorted more than the one from the addition of normally distributed noise (lower left).

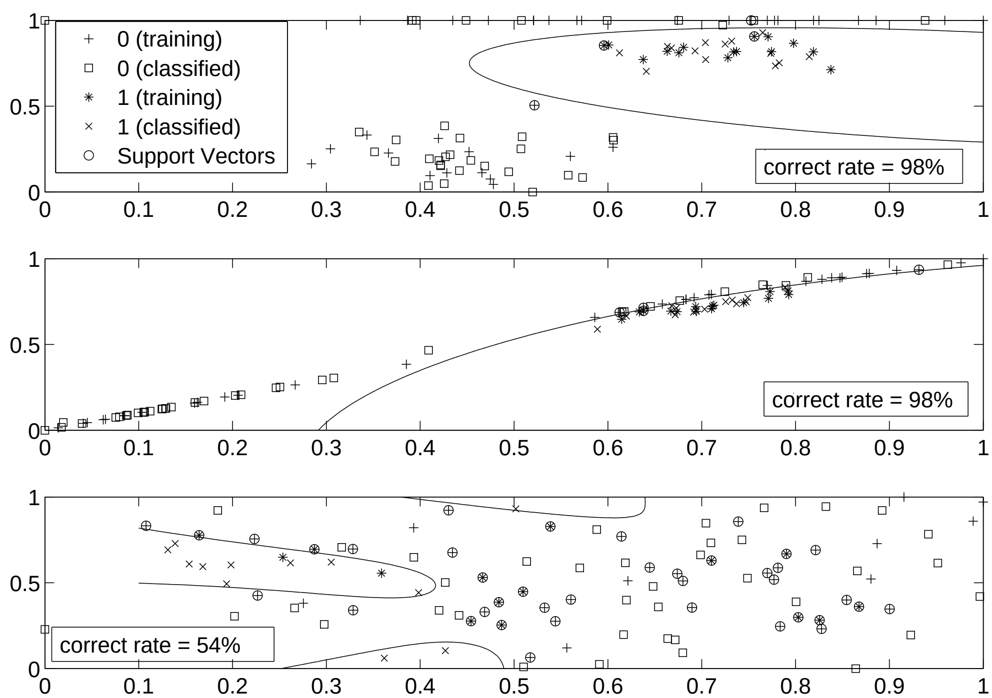


Figure 4.2: Binary SVM classification on the original data (top), the modified data by NMF (middle) and the modified data by adding uniformly distributed noise (bottom).

Minimizing the impact of data distortion on mining results is another requirement for good data hiding schemes. Our basic data modification scheme using NMF can maintain data patterns better than some classical noise-additive methods. The synthetic dataset in Figure 4.1 is used as an example to demonstrate this claim. In Figure 4.2, three scatter plots are used to illustrate the execution of a binary Support Vector Machine (SVM) classification on the synthetic data, the modified version using NMF and the modified version using the addition of uniformly distributed noise. A binary SVM classifier is trained to separate class 1 from class 2 and class 3. Using the same training set and testing set, the modified version from NMF has the same correct classification rate as that of the original data which is 98%. The addition of uniformly distributed noise deteriorates the classification accuracy and its correct classification rate is reduced to only 54%.

4.3.3 NMF-based Data Modification

Our proposed NMF method for data distortion consists of three parts: *initialization*, *data factorization* and *further distortion*. Each part includes several steps detailed in Algorithm 7 in Table 4.5.

Table 4.5: Algorithm 7: NMF-based data modification.

Algorithm 7 NMF-based data modification.

Input: a data set $A \in \mathcal{R}^{n\times m}$, a learning algorithm L, an NMF algorithm NMFAlgorithm, error and stopping condition tol, 0<K$\ll \min\{n, m\}$.

Output: the final distorted dataset $\widetilde{A}$.

begin

Initialization:

1. preprocess the original data set A
2. examine its nonnegative property and do normalization if necessary
3. set up the objective function for NMF

Factorization:

for $K = 2$ *to* m **do**
- 4. randomly generate initial estimates of two nonnegative matrices: $(H^{(0)}_{n\times K}, W^{(0)}_{K\times m})$
- 5. compute $(H_{n\times K}, W_{K\times m})$ = NMFAlgorithm $(H^{(0)}_{n\times K}, W^{(0)}_{K\times m})$
- 6. approximation $\widetilde{A}^{(K)} = H_{n\times K}W_{K\times m}$
- 7. save $\widetilde{A}^{(K)}$

end

Further Distortion & Comparison:

for $K = 2$ *to* m **do**
- 8. evaluate data distortion level of $\widetilde{A}^{(K)}$
- 9. compute mining accuracies
- 10. or do further distortion:
- **for** $r = K$ *to* 2 **do**
  - 11. $\widetilde{A}^{(r)} = H_{n\times r}W_{r\times m}$
  - 12. evaluate data distortion level of $\widetilde{A}^{(r)}$
  - 13. compute mining accuracies
- **end**

end

Publication:

14. choose one $\widetilde{A}^{(K)}$ or $\widetilde{A}^{(r)} \longrightarrow B$ with satisfactory data distortion level and accuracies
15. Publish the final modified data set, B

end

4.4 Experiments and Results

4.4.1 Comparison of Two Iterative NMF Algorithms

Two NMF algorithms are implemented on the WBC dataset to compare their performances. One is *multiplicative update* in Algorithm 3, denoted by **NMFM**. The other is *alternating projected gradients* for each sub-problem, denoted by **NMF**. The problem size $(n, K, m) = (699, 7, 9)$. All tests share the same initial estimate of $(H^{(0)}_{699\times 7}, W^{(0)}_{7\times 9})$. The tolerance is set to be 10^{-3}, 10^{-4}, 10^{-5} and 10^{-6} in order to examine convergence speed. We also impose a time limit of 4000 seconds and a maximal number of 50000 iterations on each method. Table 4.6 shows that when the tolerance is 10^{-5}, NMFM often exceeds the iteration limit of 50000. Obviously NMF is superior to NMFM. The data in the following experiments are collected by using NMF algorithm only. Some notes for Table 4.6: NMF: alternating projected gradients method. NMFM: multiplicative updating method. initial objective value: 276.2; initial gradient norm: 7609.7; dimension:7. When tolerance is more stringent than 10^{-5}, the number of iterations of NMFM exceeds the prescribed limit.

Table 4.6: Performance comparison of two NMF algorithms

Tolerance	# of Iter.		Time (seconds)		Final Gradient Norm		Objective Values	
-	NMF	NMFM	NMF	NMFM	NMF	NMFM	NMF	NMFM
1e-3	17	3060	0.8	2.6	1.04	7.11	41.4	41.5
1e-4	94	20000	3.6	23.1	0.09	1.54	41.3	41.4
1e-5	386	50000	9.8	49.7	0.01	0.84	41.4	41.5
1e-6	2382	-	63.3	-	0.001	-	41.4	-

4.4.2 Performance of NMF Algorithm Using Projected Gradients

An initial random guess on W and H is the first step in the beginning of iteration. Different starting values lead to different initial gradient norms. Therefore, the result and iteration time are dependent on the initial guess. The computational costs are roughly examined on dimension value from 9 to 2 under the tolerance$= 10^{-4}$. The result is shown in Table 4.7.

Table 4.7: Performance of NMF algorithm using projected gradients

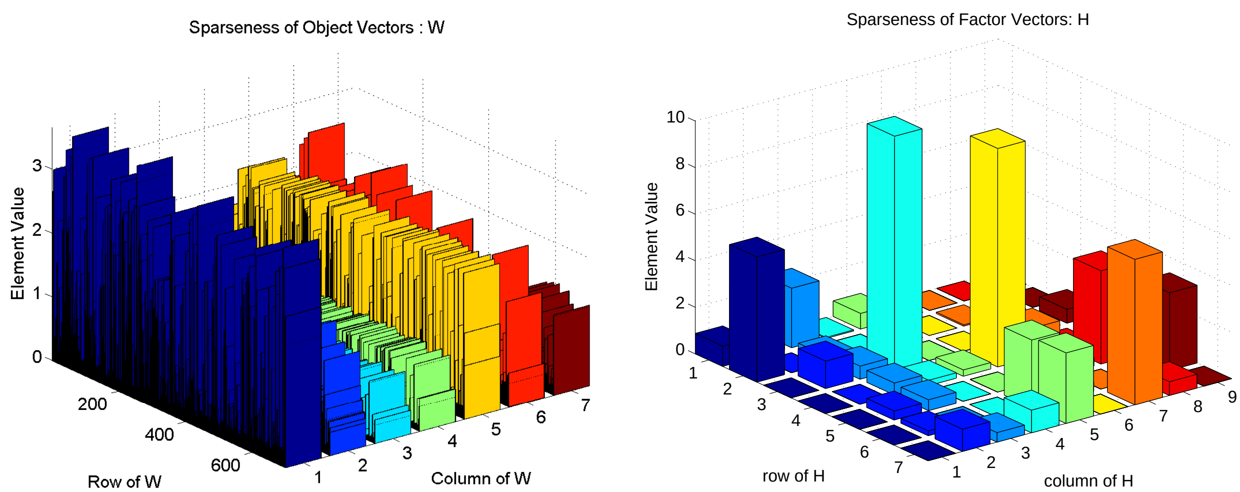
dimension	Initial Gradient Norm	# of Iteration	Run Time (seconds)
9	16525	83	12.41
8	11584	94	7.44
7	10648	80	7.38
6	7499	109	8.84
5	4816	117	7.85
4	5196	128	9.20
3	3265	76	4.65
2	4312	20	0.52

4.4.3 Sparseness Level of W and H

NMF factorization yields two submatrices with higher sparseness than those obtained by the SVD. In the following experiment, the sparseness of a nonzero vector x of length n is defined as

$$\text{sparseness}(x) = \frac{\sqrt{n} - \|x\|_1/\|x\|_2}{\sqrt{n} - 1} \tag{4.24}$$

To measure the sparseness of a matrix, we stack columns of the matrix to form a vector. The maximal sparseness of x is 1 if x contains $n-1$ zeros, and it reaches zero if the absolute values of all coefficients of x coincide.



(a) Sparseness of basis vectors $W = 0.34$.

(b) Sparseness of factor vectors $H = 0.64$.

Figure 4.3: Sparseness levels of basis and factor vectors created by NMF algorithm on the WBC data with $K = 7$ and tolerance$= 10^{-4}$.

Figures 4.3(a) and 4.3(b) illustrate the bar plots of W and H created by NMF algorithm on the WBC data with $K = 7$ and tolerance$= 10^{-4}$. The sparseness of W and H are 0.34 and 0.64 respectively. More than 50% of entries in H are zeros.

The algorithms to compute H and W used in our method make factor vectors sparser in preference to basis vectors. When the basis vectors tend to be sparse, implicitly this suggests that the basis will involve only some of the original attributes. While that basis vectors are denser than the factor vectors implies the subjects are combinations of all of bases.

4.4.4 Comparison of NMF-based Data Hiding Strategies with SVD, UD and ND on WBC

The ten distortion methods, SVD-based, NMF-based, uniformly distributed noise (UD), normally distributed noise (ND), sparsified SVD-based, and sparsified NMF-based are implemented on the WBC data to compare their performances.

In order to be fair in comparing the data distortion metrics, parameters are set to such values as to make RE values of UD, ND, SVD and NMF as close as possible. The rank of SVD is 7. The dimension size in NMF is 7 and final dimension is also 7. The results of performance evaluation on the ten methods are provided in Table 4.8.

Under the premise on the same level of value difference, the fact that CP value of UD and ND is 0 and CK value is 1 indicate that additive noise methods are worse than matrix-decomposition-based methods.

Table 4.8: Comparison of different modification strategies on WBC

Methods	Level of Distortion					Accuracy
	RE	RP	RK	CP	CK	(SVM)%
WBC	-	-	-	-	-	96.4
UD	0.1085	219.6993	0.0130	0	1	96.4
ND	0.1098	224.8148	0.0084	0	1	96.3
SVD	0.1222	228.8972	0.0114	0.2222	0.7778	96.4
NMF	0.1228	228.4295	0.0100	0.2222	0.7778	96.7
s-SVD	1.2662	228.1370	0.0013	3.3333	0	96.6
c-SVD	1.2702	230.1561	0.0021	3.3333	0	96.4
e-SVD	1.2704	228.0744	0.0014	3.3333	0	96.4
s-NMF	0.1228	228.4362	0.0076	0.2222	0.7778	96.4
c-NMF	0.1297	226.5042	0.0081	0.2222	0.7778	96.5
e-NMF	0.1234	228.2035	0.0089	1.1111	0.5556	96.5

Experimental data in Table 4.8 supports the following conclusions

1. NMF-based distortion strategy achieves a comparable performance with SVD-based strategy. In particular, NMF achieves the highest classification accuracy.

2. No improvement on performance of NMF is obtained by applying sparsification strategies. It is reasonable under the condition that NMF is a sparse factorization and the two factors, W and H, have a deep level of sparseness. Thus, further sparsification does not provide any improvement.

3. Sparsified SVD performs best on privacy level without any degradation on data mining accuracy. It is obvious that sparsification has a strong effect on data privacy level of SVD by making all the attributes change their rank in average value because CK value is 0.

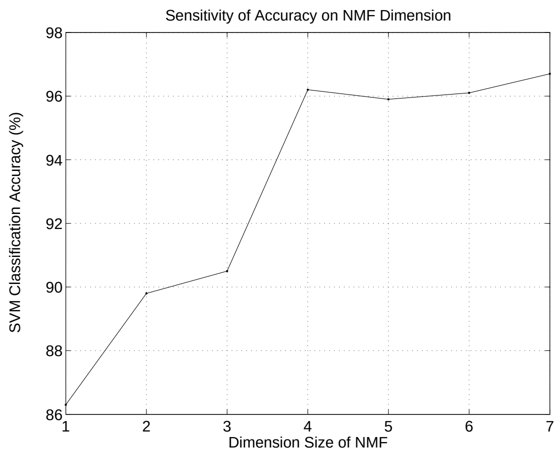
4. As to the mining accuracy, all the ten methods achieve a level comparable to or better than the original dataset.

4.4.5 Sensitivity of Performance on Dimension of NMF

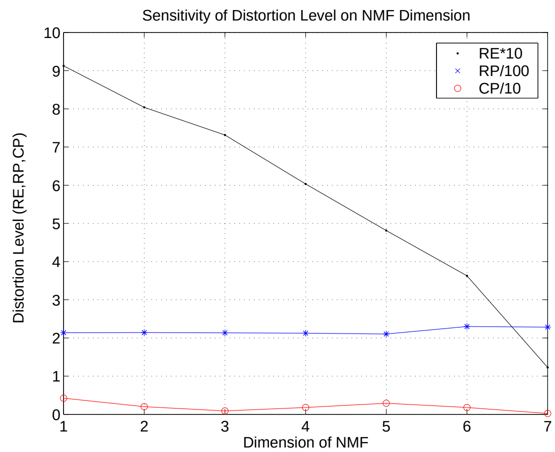
To examine the effect of dimension size on data distortion level and data utility level in the NMF approximation, we conduct an the experiment on the WBC data and Figure 4.4(a) illustrates the influence of dimension size on distortion level and classification accuracy. Here W and H are solved under dimension of $K = 7$. Then the final compressed approximation of the WBC data is computed by setting up r from 6 to 1.

Dimension size is a key element both for dimension reduction and distortion level. The smaller the dimension size is, the higher the privacy level of the method is. However, clearly, dimension size is negatively related to data utility level. Figure 4.4(a), Figure 4.4(b) and Figure 4.4(c) illustrate the above relationship.

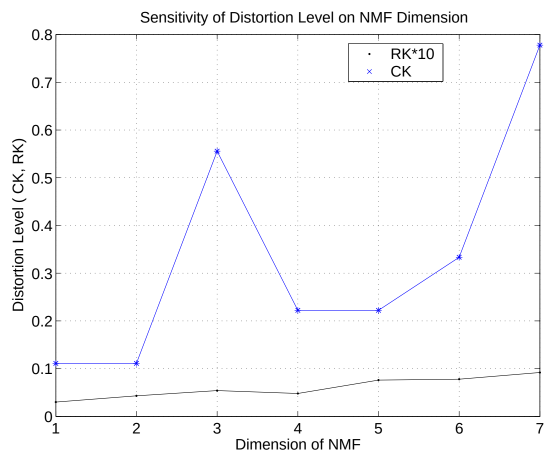
How to choose a dimension size in the proposed method is an empirical problem. For the WBC data, our experiments imply one possible good choice for our distortion method both considering data utility and data privacy. When the initial dimension size is 7, we can choose 4 as a reasonable size.



(a) Sensitivity of accuracy on NMF dimension



(b) Sensitivity of distortion level on NMF dimension



(c) Sensitivity of distortion level on NMF dimension

Figure 4.4: Sensitivity of performance of NMF-based method on NMF dimension.

4.5 Summary

Experimental results indicate that by a careful choice of iterative parameter settings, two sparse nonnegative factors can be computed by some efficient iterative algorithms. Alternating least-squares using projected gradients in computing NMF converges faster than multiplicative update methods. The two matrices are not unique because they are dependent on initial estimates at the beginning of the iterative procedure. This dependency provides our method both with uncertainty and flexibility. For nonnegative-valued datasets, our proposed method provides a possibility of simultaneously achieving satisfactory privacy, accuracy and efficiency. In our experiments, with the same level of data distortion as other data distortion methods, the NMF method demonstrates the highest classification accuracy. In particular, we foresee that using iterative factorization of the original data set can fulfill all three goals can reach an above-average point.

For the first time, we have considered high accuracy privacy preserving of nonnegative-valued datasets using NMF. The important properties of the NMF, nonnegativity and sparseness, make it not only a good dimension reduction technique but also an efficient privacy preserving tool. The promising performance of the proposed method with respect to data privacy and data utility further inspires our future work emphasizing matrix decomposition techniques.

Chapter 5

Simultaneous Pattern and Data Hiding

We have discussed the data value hiding (DVH) in the previous chapters. Two matrix-decomposition-based models have been proposed to achieve a balance between data privacy and data utility or information loss, where the attribute values are modified so that disclosure risk on sensitive attributes is reduced and the influence of data distortion on the mining results is small. In this chapter, the second category of PPDM, data pattern hiding (DPH), is considered too.

Clifton *et al.* [20, 19] propose some possible approaches to pattern hiding, including limiting access to the data, fuzzyfying data, eliminating unnecessary groupings and augmenting the data. Compared to a rich literature on attribute-value hiding, the published work on pattern hiding is mainly limited to association rule hiding and classification rule hiding [10, 25, 78, 84].

To our knowledge, no effort has been made on realizing both attribute-value hiding and data pattern hiding by using one transformed version of the original data. The challenge is that data transformation might lead to some undesirable side effect on the outcome of the data mining process. It follows that two different modified data versions may be required to fulfill these two disparate ends.

Experimental results of our previous work in §4 show that NMF can be used to distort sensitive data sets and it outperforms some traditional noise-additive methods in data hiding [82]. It provides a feasible platform to achieve both data hiding and pattern hiding.

NMF decomposes a nonnegative matrix A into two nonnegative matrices, $A \approx HW$. It was shown in [26] that when the Frobenius norm is used as a divergence metric, NMF is equivalent to a relaxed form of K-means clustering. Basis matrix W contains K cluster centroids and factor matrix H is a cluster membership indicator. Based on this relationship, we make an attempt on construction of one modified version of the original data set for attribute-value distortion and data pattern protection. Accordingly, the protection of privacy is simplified with enhanced performance. Four schemes are introduced for $\mathcal{K}$-means clustering with some assumptions on numerical attribute values and that the data patterns are limited to the pre-specified memberships or associations of data subjects. Under the constraint of zero side effects on pattern protection, an optimal solution can be produced for some data sets in our experiments to hide memberships or associations of data subjects.

In this Chapter, we make attempt on developing a simultaneous data and pattern hiding strategy for data mining activities using $\mathcal{K}$-means clustering. We consider an unclassified or unlabeled dataset T consisting of n subjects or data points, each of which has m attributes. Data clustering methods can be used to find the cluster properties of the data under a prior assumption of the number of clusters K. T is partitioned into K subsets which are referred to as clusters or classes. Each data subject is a member of a particular cluster or subset. Vector space data model is used to represent T by a matrix A (defined in §5.2).

In order to realize the dual privacy protection in one modified data set, a novel strategy composed of four schemes is proposed. The strategy is based on NMF. One scheme is proposed to achieve general attribute value hiding by way of the basic NMF. The other three schemes are designed for hiding specified cluster properties of data subjects. The basic idea is an underlying correlation of factor vectors, which are computed by the NMF, with cluster properties, which are produced by $\mathcal{K}$-means clustering [26]. In the three schemes, slight alterations are made through three kinds of factor swapping. A detailed performance evaluation is also carried out in order to demonstrate the efficacy of the proposed strategy on the DVH and DPH. Under the constraint of zero side effects on pattern protection, our implementations can compute some optimal solutions that can protect attribute values and user-specified confidential cluster properties, while nonconfidential patterns are maintained. Accordingly, the protection of privacy is simplified with enhanced performance.

5.1 Problem Description

Our approach targets the simultaneous realization of the DVH and the DPH in a centralized database with a numerical attribute set from some continuous real domain. A modified/distorted data set is computed to reduce a disclosure risk of data values and limit the influence of data distortion on data cluster properties in $\mathcal{K}$-means clustering. In the same modified data set is reflected the realization of the DPH on the variations of selected cluster properties. The influence of data distortion on nonconfidential cluster properties should be limited. This may be the first work to formally introduce the dual data hiding. It is imperative therefore to define terms and common expressions used in the paper.

Definition 5.1.1. Data Model T. Given a data set T consisting of n independent subjects in an m-dimensional feature space. If we denote the i^{th} subject of T as T_i, then

1. $T = \{T_i\}_{i=1}^{n}$
2. $T_i = \{t_{i1}, t_{i2}, \ldots, t_{ij}, \ldots, t_{im}\}, 1 \leq i \leq n, 1 \leq j \leq m.$

Definition 5.1.2. Vector Space Data Model A. Given a data model T, T can be represented by a matrix A, $A \in \mathbb{R}^{n \times m}$, with the rows corresponding to the n subjects and the columns to the m attributes. If the ith row is denoted by A_i, then A_i represents A_i. The j^{th} attribute is represented by the j^{th} column of A, denoted by $A_{.j}$. In the paper, we use A to denote a data set.

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}, \text{ or } A = [\ A_{.1}\ A_{.2}\ \ldots\ A_{.m}\ ].$$

Definition 5.1.3. Data Cluster C. Given a data set A, the number of clusters K and a learning algorithm I, C_1, C_2,..., C_K are K subsets of A, created by I; c_1, c_2,..., c_K are K cluster centroids, such that:

1. $A = \bigcup_{i=1}^{K} C_i$,
2. $\forall p, q \in \{1, 2, \ldots, K\}, p \neq q, C_p \cap C_q = \Phi$,
3. $|C_i|$ = the number of data subjects in C_i,
4. $c_i = \frac{1}{|C_i|}(\sum_{A_j \in C_i} A_j)$.

Definition 5.1.4. Data Modification. Given two data sets A and $\widetilde{A}$ with the matrix models A and $\widetilde{A}$, and a modification scheme M, a sequence of modifications is a function Ψ to transform A into $\widetilde{A}$, where F indicates the attributes or data patterns to be modified.

$$\Psi : (A, F, M) \longrightarrow \widetilde{A}.$$

Definition 5.1.5. Data Value Hiding (DVH). Given a data model A, the attributes to be modified F and a learning algorithm I, a data distortion scheme M is selected to execute data modification and compute $\widetilde{A}$: $\Omega : (\mathrm{A}, F, M) \rightarrow \widetilde{A}$. The values of F is considered to be hidden in $\widetilde{A}$ if the following conditions are satisfied:

1. In $\widetilde{A}$, the original values of F is controlled without unauthorized access.
2. The mutual information between confidential attributes and their counterparts in $\widetilde{A}$ is limited to a user-defined threshold level.

Definition 5.1.6. Data Pattern Hiding (DPH). Given a data model A, user-defined confidential pattern P and a learning algorithm I, a data distortion method M is selected to execute data modification and compute $\widetilde{A}$: $\Psi : (\mathrm{A}, F, M) \rightarrow \widetilde{A}$. Two sets of learning results PO and $\widetilde{PO}$ are created by performing I on A and $\widetilde{A}$, respectively. P will be considered to be hidden in $\widetilde{A}$ if $P \subseteq PO$, and $P \nsubseteq \widetilde{PO}$.

Definition 5.1.7. Pairwise Association R. Given a data set A, let A^2 denote $A \times A$, the set of all possible unordered pairs of subjects of A, an association R is a binary relation over a function $\Psi : (A^2, I, C) \to \{true, false\}$. For all unordered pair $(x, y) \in A^2$, there exist $p, q, 1 \leq p, q \leq K$, $A_x \in C_p, A_y \in C_q$.

1. If $p = q$, that is, A_x and A_y are in the same cluster, then $xRy = true$: $p = q \to xRy = true$;

2. if $p \neq q$, that is, A_x and A_y are not in the same cluster, then $xRy = false$: $p \neq q \to xRy = false$.

Lemma 5.1.1. *R is an equivalence relation.*

Proof. First, R is reflexive as $\forall A_i \in A, A_i R A_i$. Second, it is symmetric, as for all $i, j, 1 \leq i \leq n, 1 \leq j \leq n, A_i R A_j$ means that A_i and A_j are in the same cluster which implies $A_j R A_i$. Third, it is transitive, as whenever A_i is in the same cluster as A_j and A_j is the same cluster as A_t, then A_i is in the same cluster as A_t, therefore $A_i R A_t$. ■

Definition 5.1.8. Confidential Association Hiding. Let $\widetilde{A}$ be the data set after applying a sequence of modifications on A and an unordered pair $(A_i, A_j) \in A^2$. $A_i R A_j$ is hidden if the following conditions are satisfied:

1. $l = A_i R A_j$ in A,

2. $g = \widetilde{A_i} R \widetilde{A_j}$ in $\widetilde{A}$,

3. $g \neq l$

Our purpose here is to do general value hiding as Definition 5.1.5 and user-defined confidential association hiding as Definition 5.1.8. Particularly, we need to approximate A with $\widetilde{A}$ in which the original values of sensitive attributes are generally distorted, and prespecified confidential cluster properties are protected from being extracted by $\mathcal{K}$-means clustering. Either memberships of subjects are shifted into a new cluster different than their original ones, or pairwise associations of subject pairs are negated. In the meantime, the negative side effects of data modification on nonconfidential memberships are limited.

5.2 NMF and $\mathcal{K}$-means Clustering

Clustering algorithms group a set of subjects into clusters. They are divided into two groups: hard clustering and soft clustering. Hard clustering assigns one subject to exactly one cluster. Soft clustering computes a distribution of a subject over all clusters, and a subject has fractional membership in several clusters [15]. $\mathcal{K}$-means clustering is a hard clustering algorithm. Subject A_i is assigned to cluster C_k if it is closest to the centroid, c_k, by some distance measure. Variation on its distances to the K centroids might incur a shift of A_i from its old cluster to a new cluster. In [26], it shows there is some connection between $\mathcal{K}$-means clustering and NMF. Based on their relationship, a DVH approach is proposed.

All the pairwise distances between the rows of A can create a symmetric matrix $P \in \mathbb{R}_+^{n\times n}$ that stores a collection of pairwise distances between each pair of subjects in A,

$$P = \begin{vmatrix} 0 & \dots & \dots & \dots & \dots \\ p_{21} & 0 & \dots & \dots & \dots \\ p_{31} & p_{32} & 0 & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \dots & \dots & 0 \end{vmatrix}_{n\times n} \tag{5.1}$$

where the diagonal elements are self-distances and they are equal to zero. Each element p_{ij} corresponds to the distance or dissimilarity between subjects i and j. In general, p_{ij} is a nonnegative value that is close to zero when the subjects i and j are very similar to each other, and becomes larger the more they differ. We use the most popular distance measure, the Euclidean distance, to calculate P,

$$\begin{aligned} P_{ij} &= \|A_i - A_j\|_F \\ &= \left(tr((A_i - A_j)^T(A_i - A_j))\right)^{1/2} \\ &= \begin{cases} 0 & \text{if } i = j, \\ \left(\sum_{s=1}^{m}(A_{is} - A_{js})^2\right)^{1/2} & \text{if } i \neq j. \end{cases} \end{aligned} \tag{5.2}$$

where A_i and A_j are m-dimensional data subjects.

Inner product of each row can produce the inner product similarity matrix $S = AA^T$. An interesting observation on P and S is that they demonstrate block patterns if we arrange the subjects from the same cluster together [11]. The heat maps of P and S of IRIS data set [2] in Fig. 5.1(b) and 5.1(c) show 9 blocks each since IRIS is partitioned into 3 clusters. The darkness in dissimilarity matrix of IRIS in Fig. 5.1(b) shows the smaller within-cluster dissimilarity. The solution of the clustering should either maximize within-cluster similarity or minimize within-cluster dissimilarity.

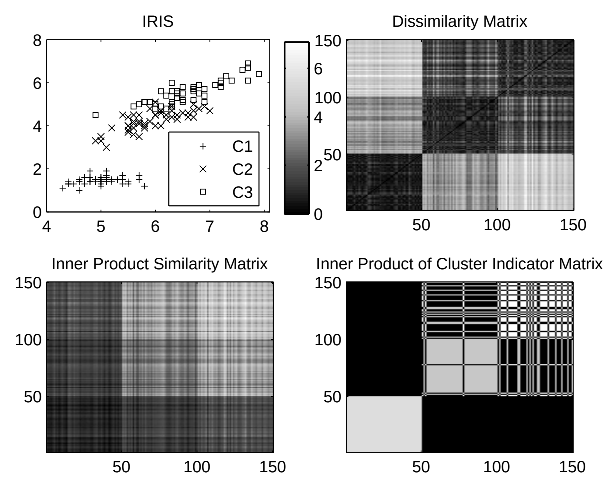


Figure 5.1: Cluster Distribution and Property Matrices of IRIS. (a) data distribution. (b) dissimilarity matrix of IRIS: P. (c) similarity matrix: S. (d) DD^T, D: cluster indicator matrix.

The clustering solution can be represented by a nonnegative *cluster indicator matrix* $D \in \mathbb{R}_+^{n\times K}$ as in [26], $D = (D_{.1}D_{.2}\ldots D_{.K})$. $|C_k|$ is the size of the k^{th} cluster. For the hard membership,

$$D_{ik} = \begin{cases} 0 & \text{if } A_i \notin C_k, \\ \frac{1}{\sqrt{|C_k|}} & \text{if } A_i \in C_k. \end{cases} \tag{5.3}$$

Each $D_{.k}$ is normalized to unit length so that $D^T D = I$.

One can easily see that the elements of D are between 0 and 1 and the sum of the elements in each row of D is equal to 1. The significance of D_{ik} is that it denotes the membership of A_i or for the soft clustering, it reflects the degree to which A_i associates with cluster C_k. Especially, the centroids $\{c_1, c_2, \dots, c_K\}$ can be represented as

$$\left(c_1\sqrt{|C_1|}, c_2\sqrt{|C_2|}, \dots, c_K\sqrt{|C_K|}\right) = D^T A. \tag{5.4}$$

We use $\widetilde{C}$ to denote $D^T A$.

For the k^{th} cluster, the sum of all the members in C_k can be represented in terms of the k^{th} row of $D^T A$ as

$$\sum_{A_i \in C_k} A_i = \sqrt{|C_k|}(D^T A)_k = \sqrt{|C_k|}(\widetilde{C})_k\ . \tag{5.5}$$

Now if we use D as a representation of the clustering solution, then the objective function for seeking a D given A can be encoded with a *symmetric convex coding* (SCC) model J that is built on S [11].

$$\min_{D \in \mathbb{R}_+^{n \times K}, B \in \mathbb{R}_+^{K \times K}, D^T D = I, B^T = B} J = \|S - DBD^T\|^2. \tag{5.6}$$

Here, S is defined as

$$S = (S_{ij})_{i \in [1,n], j \in [1,n]} = AA^T.$$

In [11], it is shown that the minimization of the objective function J in (5.6) is equivalent to

$$\max_{B^T = B, B \in \mathbb{R}_+^{K \times K}} tr(BB). \tag{5.7}$$

The proof process in [11] is as follows:

$$\begin{aligned} J &= \|S - DBD^T\|^2 \\ &= tr[(S - DBD^T)^T(S - DBD^T)] \\ &= tr(S^TS - S^TDBD^T - DB^TD^TS + DB^TD^TDBD^T) \\ &= tr(S^TS) - 2tr(DBD^TS) + tr(DB^TBD^T) \\ &= tr(S^TS) - 2tr(D^TSDB) + tr(BB) \end{aligned} \tag{5.8}$$

The above deduction uses the property of trace $tr(XY) = tr(YX)$, $S^T = S$, $B^T = B$ and $D^TD = I$. Then taking $\partial J/\partial B$ in (5.8)

$$\begin{aligned} \frac{\partial J}{\partial B} &= -2\frac{\partial tr(D^TSDB)}{\partial B} + \frac{\partial tr(B^TB)}{\partial B} \\ &= -2D^TSD + 2B = 0, \end{aligned} \tag{5.9}$$

and setting it to zero, we obtain

$$B = D^TSD = D^TAA^TD = \widetilde{C}\widetilde{C}^T. \tag{5.10}$$

Now (5.8) becomes $tr(S^TS) - tr(BB)$. Since $tr(S^TS)$ is a constant, the minimization of J is reduced to the maximization of $tr(BB)$. The proof is completed.

Hence, the K by K symmetric matrix B can be viewed as a cluster-similarity matrix such that its diagonal element B_{ii} denotes within-cluster similarity of C_i and its off-diagonal element B_{ij} denotes the similarity between the i^{th} cluster C_i and the j^{th} cluster C_j.

5.2.1 $\mathcal{K}$-means Clustering

Next, we examine the $\mathcal{K}$-means clustering. In the $\mathcal{K}$-means clustering, the objective function $\mathcal{L}$, using Euclidean distance, is used to minimize within-cluster dissimilarities.

$$\min_{C_k{}_{k=1}^K} \mathcal{L} = \sum_{k=1}^{K} \sum_{A_i \in C_k} \left\|(A_i - c_k)^T\right\|_F^2 . \tag{5.11}$$

In [26], it shows that the minimization (5.11) is equivalent to the maximization

$$\max_{D^TD=I, D\in\mathbb{R}_+^{n\times K}} \mathcal{L}(D) = tr(D^TSD). \tag{5.12}$$

In order to understand this equivalence, the proof in [26] is presented here:

$$\begin{aligned}
\mathcal{L} &= \sum_{k=1}^{K} \sum_{A_i \in C_k} \left\| (A_i - c_k)^T \right\|_F^2 \\
&= \sum_{k=1}^{K} \sum_{A_i \in C_k} \left[(A_i - c_k)(A_i - c_k)^T \right] \\
&= \sum_{k=1}^{K} \sum_{A_i \in C_k} A_i A_i^T - 2 \sum_{k=1}^{K} \sum_{A_i \in C_k} A_i c_k^T + \sum_{k=1}^{K} \sum_{A_i \in C_k} c_k c_k^T
\end{aligned} \tag{5.13}$$

We simplify the three terms in (5.13) as follows:

$$\sum_{k=1}^{K} \sum_{A_i \in C_k} A_i A_i^T = \|A\|_F^2 = tr(AA^T). \tag{5.14}$$

$$\begin{aligned}
\sum_{k=1}^{K} \sum_{A_i \in C_k} A_i c_k^T &= \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{A_i \in C_k} \left(A_i \sum_{A_i \in C_k} A_i^T \right) \\
&= \sum_{k=1}^{K} \left(\frac{1}{|C_k|} \sum_{A_i \in C_k} A_i \sum_{A_i \in C_k} A_i^T \right).
\end{aligned} \tag{5.15}$$

$$\begin{aligned}
\sum_{k=1}^{K} \sum_{A_i \in C_k} c_k c_k^T &= \sum_{k=1}^{K} |C_k| c_k c_k^T \\
&= \sum_{k=1}^{K} \left(\frac{1}{|C_k|} \sum_{A_i \in C_k} A_i \sum_{A_i \in C_k} A_i^T \right).
\end{aligned} \tag{5.16}$$

By substituting (5.5) into (5.15) and (5.16), the second and third terms of $\mathcal{L}$ in (5.13) become

$$\begin{aligned}
-\sum_{k=1}^{K} (D^T A)_k (D^T A)_k^T &= -tr((D^T A)(D^T A)^T) \\
&= -tr(D^T A A^T D).
\end{aligned} \tag{5.17}$$

Now $\mathcal{L}$ in (5.13) becomes

$$\begin{aligned}
\mathcal{L} &= tr(AA^T) - tr(D^T A A^T D) \\
&= tr(S) - tr(D^T S D).
\end{aligned} \tag{5.18}$$

Since $tr(S)$ is a constant, $\min \mathcal{L}$ becomes $\max \mathcal{L}(D) = tr(D^T S D)$. The proof is completed.

Considering (5.10), $\max tr(D^T S D)$ is equivalent to $\max tr(B)$ that is to maximize the sum of the diagonal elements of B which represent the within-cluster similarities.

5.2.2 Nonnegative Matrix Factorization (NMF)

Choosing one NMF algorithm, a data set A can be decomposed into two nonnegative factor matrices. The transformation from A to $\widetilde{A}$ can be defined as follows: *Given a nonnegative data model $A \in \mathbb{R}_+^{n\times m}$, find two nonnegative matrices $H \in \mathbb{R}_+^{n\times K}$ and $W \in \mathbb{R}_+^{K\times m}$ with K being the number of clusters in A, that minimize $\mathcal{Q}$, where $\mathcal{Q}$ is an objective function defining the nearness between the matrices A and HW. The modified version of A is denoted as $\widetilde{A} = HW$.*

The choice of the objective function $\mathcal{Q}$ affects the solution of $\widetilde{A}$. Here, the Euclidean distance or the Frobenius norm is chosen as they are popular in matrix computations,

$$\min_{H\in\mathbb{R}_+^{n\times K}, W\in\mathbb{R}_+^{K\times m}} \mathcal{Q} = \|A - HW\|_F^2 \, . \tag{5.19}$$

In [26], a proof on the equivalence between $\mathcal{K}$-means clustering and NMF is presented, which starts from doing some manipulations on $\mathcal{Q}$

$$\begin{aligned}
\mathcal{Q} &= \|A - HW\|_F^2 \\
&= tr((A - HW)^T(A - HW)) \\
&= tr(A^TA - A^THW - W^TH^TA + W^TH^THW) \\
&= tr(A^TA) - 2tr(A^THW) + tr(W^TH^THW)
\end{aligned} \tag{5.20}$$

Let $\partial Q/\partial W = 0$,

$$\begin{aligned}
\frac{\partial \mathcal{Q}}{\partial W} &= -2\frac{\partial tr(A^THW)}{\partial W} + \frac{\partial tr(W^TH^THW)}{\partial W} \\
&= -2H^TA + 2H^THW \\
&= 0.
\end{aligned}$$

We obtain $H^TA = H^THW$. If we apply the orthogonality restriction into NMF such that $H^TH = I$, then $W = H^TA$. Substituting W with H^TA in (5.20), we have

$$\begin{aligned}
\mathcal{Q} &= tr(A^TA) - tr(W^TW) \\
&= tr(A^TA) - tr(A^THH^TA).
\end{aligned} \tag{5.21}$$

Comparing (5.21) with (5.18), if $H = D$, then $\mathcal{Q} = \mathcal{L}$, *i.e.*, under the restriction of orthogonality on H, the NMF is identical to the $\mathcal{K}$-means clustering,

$$\begin{aligned} & \min_{H \in \mathbb{R}_+^{n \times K}, H^T H = I, W \in \mathbb{R}_+^{K \times m}} \|A - HW\|_F^2 \\ \equiv \quad & \min_{C_{k_{k=1}^K}} \sum_{k=1}^{K} \sum_{A_i \in C_k} \left\|(A_i - c_k)^T\right\|_F^2 . \end{aligned} \tag{5.22}$$

Therefore, H is cluster indicator matrix of A and $H^T A$ is cluster centroid matrix, which is W. Without the orthogonal restriction on H, the standard NMF is a kind of soft clustering where one subject might belong to several clusters with different weights. The membership of the i^{th} subject can therefore be assigned according to the largest weight in H_i.

5.2.3 NMF-based Clustering

The equivalence between NMF and $\mathcal{K}$-means clustering is the basis of NMF-based clustering [26]. Given the number of clusters K, H and W can be computed.

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix}, \qquad W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_k \end{bmatrix}.$$

$$H_i = (h_{i1}\, h_{i2}\ \ldots\ h_{is}\ \ldots\ h_{ik}), \qquad i = 1, 2, \ldots, n.$$

$$W_j = (w_{j1}\, w_{j2}\ \ldots\ w_{jt}\ \ldots\ w_{jm}), \qquad j = 1, 2, \ldots, K.$$

H represents the cluster indicator matrix D, and W represents the cluster center matrix C. Each row of W is a basis vector to represent one of the K clusters. Each row of H is a factor vector of one of n subjects. Each of the subjects can be approximately represented by an additive combination of the K basis vectors.

$$A_i \approx \sum_{j=1}^{K} h_{ij} W_j$$

Each element h_{ij} indicates to which degree the subject i belongs to the cluster C_j, while each element w_{ij} represents the weight of contribution of attribute j to the cluster C_i.

If the subject i belongs to the cluster C_x, then h_{ix} will take on a larger value than the rest of the elements in H_i. NMF can be viewed as a kind of unsupervised learning that the

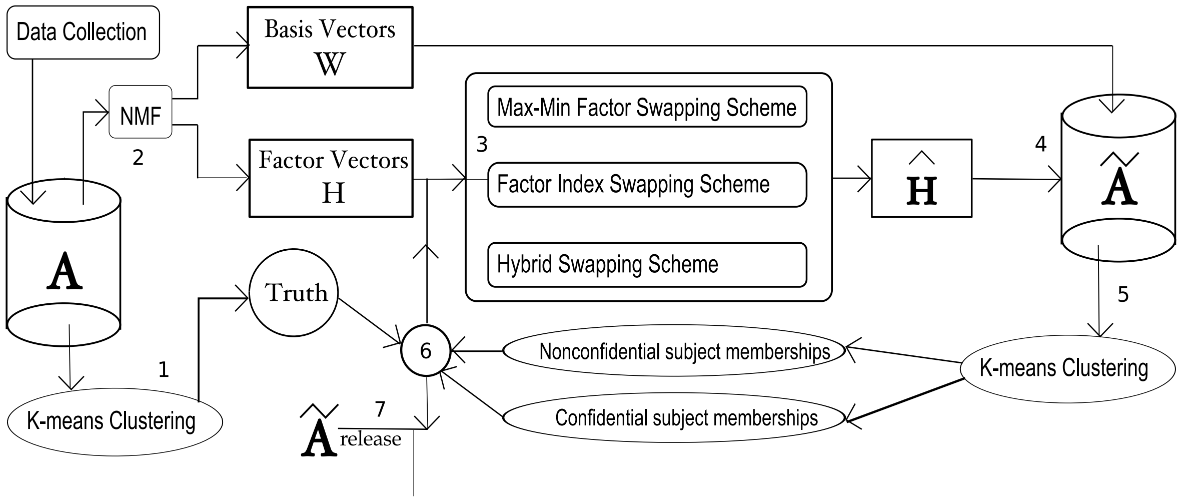


Figure 5.2: The process of dual privacy protection.

membership of the subjects can be determined by H [85]. The NMF-based clustering rule is described as: the subject A_x is placed in the cluster C_p if h_{xp} is the largest element in its factor vector H_x, *i.e.*,

$$A_x \in C_p, \text{if } p = \underset{j}{\operatorname{argmax}}\{h_{xj}\}.$$

This rule implies that any modification on factor vectors may change the memberships of the corresponding subjects. Based on this insight, we design three factor swapping schemes (described in §5) based on modifying H_x and H_y to change the membership of a single subject x or a pairwise relationship xRy.

5.3 Proposed Approach

In this section, we describe the proposed dual privacy preserving approach consisting of one data hiding scheme and three pattern hiding schemes. All schemes are based on a basic factorization scheme via an NMF on the original data set.

Fig. 5.2 is a process diagram of the dual privacy protection. A is created on T after data collection. The steps are:

1. $\mathcal{K}$-means clustering is run on A and its result of subject memberships is used as *truth*.

2. NMF of A generates two submatrices H and W.

3. Factor swapping schemes are used to transform the factor vectors H to $\hat{H}$.

4. $\hat{H}$ is combined with W to form a modified data set $\widetilde{A}$.

5. $\mathcal{K}$-means clustering is run on the modified data $\widetilde{A}$.

6. Confidential subject memberships are examined. Step 3 and 4 are repeated until confidential memberships and relationships are hidden.

7. One $\widetilde{A}$ is outputted for release.

Four schemes in the diagram will be elaborated in the following part of this section, as well as how they are adopted to hide values and memberships.

5.3.1 Basic Factorization Scheme

Given a prespecified K, the original data set A is decomposed into H and W. A standard way to find H and W is by the following least-squares optimization, which minimizes the difference between A and HW:

$$\min_{H\in\mathbb{R}_+^{n\times K},W\in\mathbb{R}_+^{K\times m}} \mathcal{Q} = \sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij} - (HW)_{ij})^2. \tag{5.23}$$

NMF algorithms generally begin by initial estimates of the matrices H and W, followed by alternating iterations to improve these estimates. Projected gradient method proposed by Lin [51] is used in our implementation to directly minimize (5.23).

The full factorization of A amounts to the two nonnegative matrices H and W as well as a residual U, such that: $A = HW + U$. The elements of the residual matrix can either be negative or positive. $\widetilde{A}$ is taken as HW, an approximate of A. Therefore, value difference caused by removing U can hide original values. The non-uniqueness of H and W is advantageous to prevention of privacy breach.

The transformation of A to $\widetilde{A}$ leads to value distortion and the relative information loss. To be more precise in talking about the amount of privacy protection, we need some scalar measures. In information theory, the mutual information between an attribute X of A and its distorted counterpart $\widetilde{X}$ in $\widetilde{A}$, denoted by $I(X;\widetilde{X})$, measures how much information $\widetilde{X}$ tell us about X, that is, how much information went through the transformation from A into $\widetilde{A}$. $I(X;\widetilde{X})$ is the reduction in uncertainty about X due to the knowledge of $\widetilde{X}$

$$\begin{aligned} I(X;\widetilde{X}) &= \mathcal{D}(p(x,\widetilde{x})||p_1(x)p_2(\widetilde{x})) \\ &= \sum_x \sum_{\widetilde{x}} p(x,\widetilde{x}) \log \frac{p(x,\widetilde{x})}{p_1(x)p_2(\widetilde{x})} \\ &= H_e(X) - H_e(X|\widetilde{X}), \end{aligned} \tag{5.24}$$

where $p(x,\widetilde{x})$ is the joint probability distribution of finding values x and $\widetilde{x}$, and $p_1(x)$ and $p_2(\widetilde{x})$ are the marginal probability distribution functions of X and $\widetilde{X}$. $H_e(X)$ is the Shannon entropy or self-information of X, which is defined as

$$H_e(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i), \tag{5.25}$$

where entropy is measured in bits when b is 2. When X and $\widetilde{X}$ are independent, $I(X;\widetilde{X})$=0, which implies $\widetilde{X}$ can provide no inference on X [22]. Therefore, both the privacy risk and data utility loss can be measured as the mutual information of attributes. In the intuitive sense, a smaller $I(X;\widetilde{X})$ may lessen the disclosure risk of the original attribute X; and on the other hand, it causes more information loss and more damaged data utility.

We define a ***mutual information row vector*** $M \in \mathbb{R}^{1\times m}$. Each element represents mutual information between one attribute and its distorted counterpart:

$$M = (I(A_{.j};\widetilde{A}_{.j}))_j. \tag{5.26}$$

The relative privacy risks among the attributes can be somehow quantified in M.

Another measure, ***entropy distortion*** or ***self-information distortion***, ED is defined as a nonnegative number:

$$ED = \frac{||E-\widetilde{E}||_F}{||E||_F}, \tag{5.27}$$

where E is a row vector whose elements are attribute entropies defined in (5.25). $||\cdot||_F$ is the Frobenius norm. ED shows how much the total distortion on the self-information of all the attributes.

5.3.2 Pattern Hiding Strategies

Given a data set A with K clusters, H and W are computed. The *max-min factor swapping scheme*, the *factor index swapping scheme* and the *hybrid modification scheme* are described as follows.

Scheme 1: Max-Min Factor Swapping Scheme. Let x be the index of the selected subject in A. The factor vector of A_x is $H_x = (h_{x1}\ \ h_{x2} \dots h_{xj} \dots h_{xK})$. The largest factor is swapped with the smallest factor in H_x.

Let

$$Id_{max} = \operatorname*{argmax}_{j}\{h_{xj}\}, \quad max = h_{x(IdY_{max})},$$
$$Id_{min} = \operatorname*{argmin}_{j}\{h_{xj}\}, \quad min = h_{x(IdY_{min})}.$$

then

$$h_{x(Id_{max})} \leftarrow min, \quad h_{x(Id_{min})} \leftarrow max.$$

Scheme 2: Factor Index Swapping Scheme. Given $(x, y) \in A^2$, x and y are the indices of one selected subject pair in A. The factor vectors of A_x and A_y are

$$H_x = (h_{x1}\ \ h_{x2} \dots h_{xj} \dots h_{xK}),$$
$$H_y = (h_{y1}\ \ h_{y2} \dots h_{yj} \dots h_{yK}).$$

Let

$$IdX_{max} = \operatorname*{argmax}_{j}\{h_{xj}\},$$
$$IdY_{max} = \operatorname*{argmax}_{j}\{h_{yj}\},$$
$$max_y = h_{y(IdY_{max})}.$$

- If A_x and A_y do not have the same index of the maximum factors, *i.e.*, $IdX_{max} \neq IdY_{max}$, we swap the maximum factor of A_y with the factor in the same index as the maximum factor of A_x,

$$h_{y(IdY_{max})} \leftarrow h_{y(IdX_{max})}$$

$$h_{y(IdX_{max})} \leftarrow max_y$$

- If A_x and A_y have the same index of the maximum factors, ***i.e.***, $IdX_{max} = IdY_{max}$, we swap the maximum factor of A_y with any factor not in the same index as the maximum factor of A_x. There exists $t, 1 \leq t \leq k, t \neq IdX_{max}$,

$$h_{y(IdY_{max})} \leftarrow h(y,t)$$

$$h_{yt} \leftarrow max_y$$

Scheme 3: Hybrid Swapping Scheme. Given $(x, y) \in A^2$, assume that the factor vectors of A_x and A_y are

$$H_x = (h_{x1}\ \ h_{x2} \ldots h_{xj} \ldots h_{xk}),$$

$$H_y = (h_{y1}\ \ h_{y2} \ldots h_{yj} \ldots h_{yk}).$$

Let

$$IdX_{max} = \operatorname*{argmax}_{\boldsymbol{j}}\{h_{xj}\}, \quad max_x = h(x, IdX_{max}),$$

$$IdX_{min} = \operatorname*{argmax}_{\boldsymbol{j}}\{h_{xj}\}, \quad min_x = h(x, IdX_{min}).$$

The factor vector of A_y is modified based on A_x by substituting its maximum and minimum factors for those in the same indices of A_y, then swapping them, ***i.e.***,

$$h_{y(IdX_{max})} \leftarrow min_x$$

$$h_{y(IdX_{min})} \leftarrow max_x$$

5.3.3 Single Membership Hiding

To hide the membership of one subject, we can make a shift of the subject from its source cluster to any other cluster. Since the hiding process is built on the basic data factorization, the non-uniquesness of the NMF solution may lead to unpredictable results. Different factor matrices H and W may cause a different shift of the subject even though the same hiding scheme is adopted. In order to improve the predictability on results and take advantage of the flexibility of NMF, we make use of the iterations in NMF to find an optimal

Algorithm 8 Single membership hiding.

```
Input:  a data set T with its vector space model A, cluster
        truth C, the index x of the confidential subject, the old
        membership of A_x, the new membership of A_x.
Output:  Ã, H, W, Ĥ (one distorted version of H)
begin
    Label ⟵ the old membership of A_x;
    while  Label ≠the new membership of A_x or sideEffect ≠ 0  do
        conduct basic factorization scheme to generate H and W;
        conduct Scheme 1 on factor vector H_x to produce Ĥ;
        compute Ã ← Ĥ * W;
        run clustering procedure on Ã to get new cluster labels;
        Label ← the new cluster label of A_x;
        check other subjects' membership shifts;
        update sideEffect.
    end
    output Ĥ, W and Ã.
end
```

Figure 5.3: Algorithm 1: Single membership hiding.

factorization that fulfills the requirement on value hiding and membership hiding simultaneously. Algorithm 8 in Fig. 5.3 is a single membership hiding scheme. The algorithm repeatedly executes basic factorization of A and *max-min factor swapping scheme* on the factor vector of confidential subject until the while condition meets, and a solution is found with zero side effects on the nonconfidential memberships.

Measuring the ***undesirable side effects*** associated with the pattern hiding schemes is a necessary part of the evaluation. An optimal hiding solution should be the one where only the user-specified pattern is hidden and all the rest of the patterns are kept intact, *i.e.*, there are no extra changes or nonzero side effects. Because in our experiments, the initial centroids are fixed for all the executions of the $\mathcal{K}$-means algorithm, we can quantify the side effects as a rate of the number of shifting subjects among the number of nonconfidential subjects. Here, only the shift of memberships is taken into consideration.

For example, when hiding the membership of one subject in IRIS, if 5 other subjects are shifted to clusters different from their original ones, the side effect rate can be calculated as

Algorithm 9 Single-pair relationship changing.

```
Input:  a data set T with its vector space model A, cluster truth
        C, a pair (x,y) with a confidential relationship: (xRy)_old.
Output: Ã, H and W, Ĥ (one distorted version of H)
begin
  pairTruth ⟵ (xRy)_old;
  pairNOT ⟵ (xRy)_old;
  while pairNOT == pairTruth or sideEffect ≠ 0 do
    conduct basic factorization scheme to generate H and W;
    modify the factor vectors:
       H_x or H_y by Scheme 2 or Scheme 3 to
       produce Ĥ;
    compute Ã ← Ĥ * W;
    run clustering procedure on Ã to get new cluster labels;
    pairNOT ← (xRy)_new;
    check other subjects' membership shifts;
    update sideEffect.
  end
  output Ĥ, W and Ã.
end
```

Figure 5.4: Algorithm 2: Single-pair relationship changing.

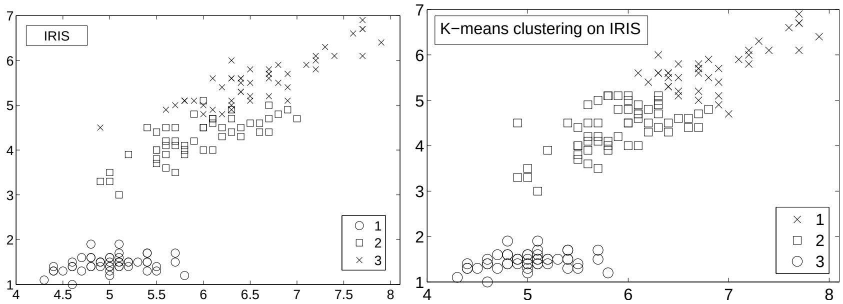
$5/149$, that is 3.36%. Obviously, the lower the side effect rate, the better the data usability following a hiding scheme.

5.3.4 Single-pair Relationship Changing

By Definition 3.7, the relationship xRy represents whether subject x and subject y belong to the same group. This relationship change on xRy is binary: from true to false or from false to true. If xRy is negated in the learning result from $\widetilde{A}$, then we consider it as a successful hiding. The membership shifts of x and y are not limited and either one or both can be shifted. However, changes on other subjects' memberships are not expected. The side effects should be avoided or limited.

Given a user-specified subject pair (x, y) in A, with the confidential relationship, the problem can be formulated as

$$\Psi : (A, (x, y), Scheme) \rightarrow \widetilde{A}.$$



(a) Cluster distribution of IRIS. (b) $\mathcal{K}$-means clustering on IRIS.

Figure 5.5: IRIS dataset and cluster distribution.

Fig. 5.4 is the proposed procedure to change a single-pair relationship. Considering multiple pair-wise relationship hiding, we can rewrite the iteration condition in the algorithm to change the pair relationships one by one.

5.4 Performance Evaluation

We conduct experiments on the IRIS data set to evaluate the performance of the proposed four schemes. IRIS contains 3 clusters of 50 subjects each, where each cluster refers to a type of iris plant and each subject has 4 attributes. As Fig. 5.5(a) shows, one cluster in cross marks is linearly separable from the other two in circle and square marks; the latter two clusters are not linearly separable from each other. Even though the experiments are mainly designed for an evaluation of three DPH schemes, the DVH by the basic factorization scheme is also examined. $\mathcal{K}$-means clustering is used as a learning tool. For the number of clusters K, we simply use the known number of the clusters. Note that how to choose the optimal number of clusters is a nontrivial model selection problem and beyond the scope of this study [11].

To achieve a fair comparison of results, during the $\mathcal{K}$-means clustering in all the experiments, the initial cluster centroids are fixed as the first three data subjects in the IRIS. First, the $\mathcal{K}$-means algorithm is run on IRIS to produce 3 clusters denoted by C_1, C_2, C_3, 3 centroids denoted by c_1, c_2, c_3 and the corresponding cluster labels. The cluster distribution created from the $\mathcal{K}$-means algorithm is shown in Figure 5.5(b). C_3 marked in circle contains

50 subjects. C_2 marked in square and C_1 in cross contain 61 and 39 subjects, respectively. 17 subjects are incorrectly grouped and the correct classification rate is 88.7%. This cluster distribution defined as C_1, C_2, C_3 in Fig. 5.5(b) is considered as the ***truth*** for estimating clustering accuracy in the subsequent experiments. The following is a description of the truth. To make it clear, the indices are used.

$C_1 = \{101 - 150\} - \{102, 107, 114, 115, 120, 122, 124,$
$127, 128, 134, 139, 143, 147, 150\} + \{51, 53, 78\}.$
$C_2 = \{51 - 100\} - \{51, 53, 78\} + \{102, 107, 114, 115,$
$120, 122, 124, 127, 128, 134, 139, 143, 147, 150\}.$
$C_3 = \{1 - 50\}.$

The three cluster centroids are

$$c_1 = [\ 6.8538 \quad 3.0769 \quad 5.7154 \quad 2.0538\ ],$$

$$c_2 = [\ 5.8836 \quad 2.7410 \quad 4.3885 \quad 1.4344\ ],$$

$$c_2 = [\ 5.0060 \quad 3.4180 \quad 1.4640 \quad 0.2440\ ].$$

Then a series of experiments are conducted to evaluate the proposed methods. The experiments abide by a common procedure, shown as Fig. 5.2 from the basic data modification to a modified version. The released version is an optimal solution for both data hiding and pattern hiding. As far as learning accuracy and the validation of pattern hiding are concerned, a comparison is made between the clustering truth and the clustering result from a modified data set.

The computation of H and W by NMF is implemented by an algorithm in [51]. In our experiments, the tolerance for a relative stopping condition is 10^{-4}. The time limit is 6000 seconds and the number of iterations is limited to 3000.

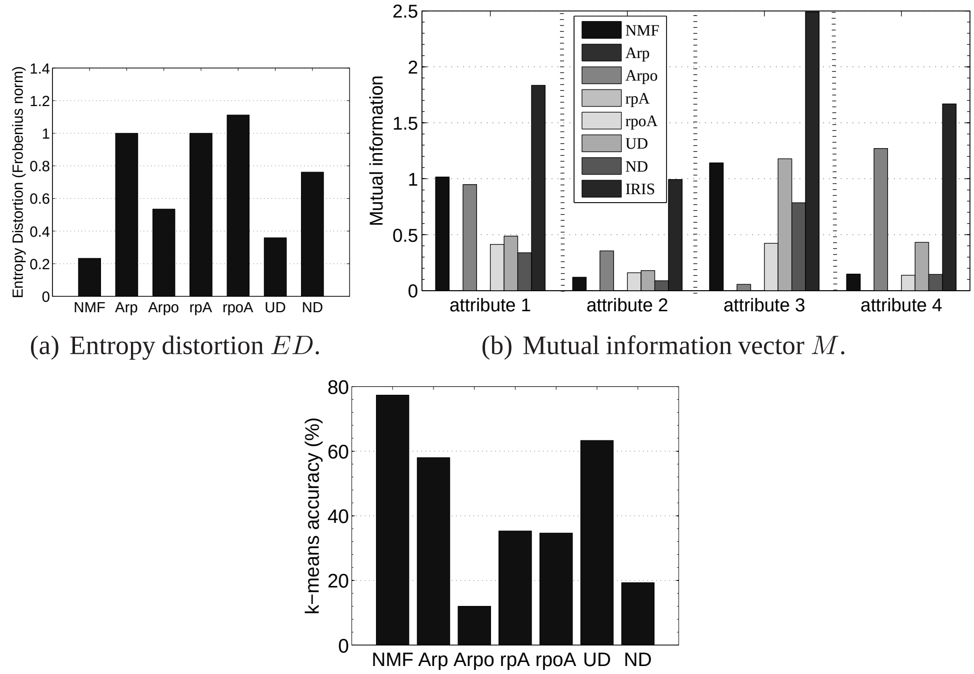
Table 5.1: The notations of seven methods.

Method Name	Method	Citation
NMF	$\widetilde{A} = HW$, $H \in \mathbb{R}^{n\times K}$,$W \in \mathbb{R}^{K\times m}$.	[81]
Arp	$\widetilde{A} = AR$, $R \in \mathbb{R}^{m\times m}$.	[43]
$Arpo$	$\widetilde{A} = AR$, $R \in \mathbb{R}^{m\times m}$, $RR^T = I$.	
rpA	$\widetilde{A} = RA$, $R \in \mathbb{R}^{n\times n}$.	
$rpoA$	$\widetilde{A} = RA$, $R \in \mathbb{R}^{n\times n}$, $R^T R = I$.	
UD	$\widetilde{A} = A + U, U \in \mathbb{R}^{n\times m}$.	[16, 30]
ND	$\widetilde{A} = A + N$, $N \in \mathbb{R}^{n\times m}$.	[45, 7]

5.4.1 Effectiveness of Basic factorization Scheme

As can be seen in many fields, there are many different methods for the same objective. We begin with a comparison of our proposed basic factorization scheme with six external perturbation methods shown in Table 5.1. NMF is used here to denote the basic factorization scheme. One noise-additive method denoted by ND is to add normally distributed noise that is generated with a mean $\mu = 0$ and a standard deviation $\sigma = 2$, to the original IRIS data set. Another noise-additive method is denoted by UD that adds uniformly distributed noise generated from the interval [0, 3] to IRIS. Four multiplicative perturbation methods use a projection matrix, R, is created by randomly sampling from some distribution with $\mu = 0$ and some variance $\sigma_r^2 = 1e - 4$. R is of size $m \times m$ for the right multiplication and $n \times n$ for the left multiplication, since in our study, the dimensions of the original and the distorted matrices are supposed to be the same. For each case, R can be either nonorthonormal or orthonormal. The shorthands are Arp, $Arpo$, rpA and $rpoA$ as described in Table 5.1.

Experiments: Seven modified data sets are computed on the IRIS data. Mutual information vector M and self-information distortion ED, as defined in (5.26) and (5.27), are calculated and listed in Table 5.2. $\mathcal{K}$-means clustering accuracies are estimated on the truth created in the beginning of this section. We list all the accuracies in Table 5.2. All external perturbation methods here have the property of randomness. NMF solution is not unique



(a) Entropy distortion ED.

(b) Mutual information vector M.

(c) $\mathcal{K}$-means clustering accuracy.

Figure 5.6: Comparison of seven data value hiding methods.

with random initial values. Therefore, the result here is considered as a demonstration of effectiveness on IRIS data set of different methods on privacy protection and information maintenance. The following analysis is made on the results of this particular experiment with previously specified parameters, even though some observations can be extended to some general senses.

Discussions: ED, M and $\mathcal{K}$-means accuracy of seven methods are shown in Fig. 5.6(a), 5.6(b) and 5.6(c), respectively. In Fig. 5.6(b), the four columns of M are grouped into four groups, each of which has eight bars. The first seven bars are the mutual information between the original attribute and its distorted value in seven data sets. From left to right in each group, the seven methods are placed in the same order as in Table 5.1. The rightmost bar in each group represents the mutual information between an original attribute and itself, which we know is the maximum value of the group. We can compare the privacy risk of each attribute under seven methods by following the common sense: the shorter bar with smaller mutual information means the distorted attribute will disclose less information on its original value than the taller bar with larger mutual information. Our proposed NMF-based scheme demonstrates a better protection on attribute 2 and 4. We also see this scheme performs well in the clustering. It seems that in this experimental environment, Arp and rpA cause lest information disclosure due to mutual information of all four attributes are zero. However, their $\mathcal{K}$-means clustering accuracies is relatively lower. They appears not to be a good candidate for privacy preserving applications emphasizing the data utility.

Similarly, entropy distortion ED in Fig. 5.6(a) and at the sixth column of Table 5.2, can be viewed as a measure of how much information are not carried into the distorted data. By an unit-valued ED, Arp and rpA preserve no entropy of original values. Correspondingly, their $\mathcal{K}$-means clustering accuracies here are lower than some others. Roughly speaking, ED is a one-dimensional measure of information loss, compared to the mutual information. Next we will turn to the evaluation of the proposed DPH schemes.

5.4.2 Membership Hiding Using Scheme 1

In this experiment, Scheme 1 is evaluated in hiding the membership of the 50^{th} subject. In the truth as defined earlier, the membership of the 50^{th} subject is C_3. A shift to C_2 or C_1 will hide its original membership. An optimal solution with minimal side effects can be obtained through the NMF iterations. First, the subject is designed to be shifted to C_2. One optimal W for this case is computed as:

$$W^* = \begin{bmatrix} 2.4284 & 1.5910 & 0.5626 & 0 \\ 2.0386 & 0.1599 & 2.1913 & 0.5940 \\ 0.6671 & 1.6579 & 0.2504 & 0.5813 \end{bmatrix}.$$

The factor vector of the 50^{th} subject is

$$H_{50} = [\ 1.8918 \quad 0.1394 \quad 0.1679\ ].$$

After swapping its maximum and minimum factor elements by using Scheme 1, the new factor vector is

$$\widehat{H}_{50} = [\ 0.1394 \quad 1.8918 \quad 0.1679\ ].$$

Leaving all the other factor vectors unchanged in $\widehat{H}$, an optimal modified version $\widetilde{A}$ is constructed as the product of $\widehat{H}$ and W^*. When the $\mathcal{K}$-means clustering is run on $\widetilde{A}$, the result is a clean shift of the 50^{th} subject from C_3 to C_2 without any additional membership changes in the rest of subjects. That means the side effect rate is 0%. Therefore, an optimal release data set can be taken as $\widetilde{A}^* = \widehat{H}W^*$.

Next, we will make anther shift of T_{50} to C_1. An optimal W generated from the NMF iterations and the corresponding factor vector of T_{50} are as follows:

$$W^{**} = \begin{bmatrix} 1.4285 & 1.1208 & 0.2422 & 0.0210 \\ 1.6549 & 0 & 1.3761 & 0.1504 \\ 1.6739 & 1.2329 & 1.6303 & 0.8675 \end{bmatrix},$$

$$H_{50} = [\ 2.9082 \quad 0.4674 \quad 0.0392\ ].$$

By executing Scheme 1, we have

$$\widehat{H}_{50} = [\ 0.0392 \quad 0.4674 \quad 2.9082\ ].$$

Accordingly, $\widetilde{A}^* = \widehat{H}W^{**}$ is an optimal solution for a shift of the 50^{th} subject from C_3 to C_1. This solution does not bring any other shifts so that the rest of the subjects remain in their original groups. The side effect is 0%.

We also conduct experiments on shifting subjects from C_2 to C_1 or C_3 and from C_1 to C_2 or C_3. For the 80^{th} subject, one optimal W and the distorted 80^{th} factor vector for the shift from C_2 to C_1 are

$$W^* = \begin{bmatrix} 2.7044 & 0 & 1.7202 & 0 \\ 1.3825 & 0.6344 & 1.4931 & 0.6260 \\ 1.1411 & 0.9137 & 0.1900 & 0.0175 \end{bmatrix},$$

$$\widehat{H}_{80} = [\ 1.8403 \quad 1.4754 \quad 0.5700\ ].$$

For the shift of A_{80} from C_2 to C_3, one optimal solution is

$$W^* = \begin{bmatrix} 0.0284 & 3.2999 & 0 & 0.9529 \\ 1.6979 & 0.9374 & 0.4465 & 0 \\ 1.0185 & 0 & 1.9840 & 0.8098 \end{bmatrix},$$

$$\widehat{H}_{80} = [\ 2.6486 \quad 0.0360 \quad 1.1725\ ].$$

For the 130^{th} subject, one optimal solution for the shift from C_1 to C_2 is

$$W^* = \begin{bmatrix} 2.0319 & 0.7374 & 0.8519 & 0 \\ 0.8570 & 1.0289 & 0 & 0.0619 \\ 0.1169 & 0 & 3.4679 & 2.1375 \end{bmatrix},$$

$$\widehat{H}_{130} = [\ 0.4487 \quad 3.3424 \quad 0.8188\ ].$$

For the shift of A_{130} from C_1 to C_3, we have

$$W^* = \begin{bmatrix} 0.1830 & 5.2784 & 0 & 0.8378 \\ 0.9032 & 0 & 3.1744 & 1.4457 \\ 2.6576 & 1.1713 & 0.7117 & 0 \end{bmatrix},$$

$$\widehat{H}_{130} = [\ 2.2949 \quad 1.2681 \quad 0.0492\ ].$$

These experimental results show that by using the iteration procedure described in Algorithm 8 in Fig. 5.3, an optimal solution without any side effects can be computed for membership hiding in IRIS. The experimental result demonstrates that Scheme 1 is an effective way to hide confidential memberships. We note that an optimal solution is not unique.

5.4.3 Relationship Change Using Scheme 2

Given a user-specified pair with the confidential relationship, (x, y) in the IRIS, using Scheme 2 to change xRy, the problem is $\Psi : (\text{IRIS}, (x, y), \text{Scheme 2}) \rightarrow \widetilde{A}^*$, where $\widetilde{A}^*$ is an optimal solution without any side effects on other subject memberships.

Test 1: $\Psi : (\text{IRIS}, (50, 80), \text{Scheme 2}) \rightarrow \widetilde{A}^*$. $50R80$ is $false$ in the clustering truth of IRIS. We need to find an $\widetilde{A}^*$ to change the relationship to $true$. Scheme 2 is carried out to produce an optimal factorization where the basis matrix is

$$W^* = \begin{bmatrix} 0.1261 & 3.3805 & 0 & 0.8557 \\ 1.7367 & 0.9309 & 0.4587 & 0 \\ 1.4324 & 0 & 2.8763 & 1.1859 \end{bmatrix}.$$

The corresponding factor vectors are

$$H_{50} = [\ 0.1948 \quad 2.8354 \quad 0.0336\ ],$$

$$H_{80} = [\ 0.0496 \quad 2.6134 \quad 0.8012\ ].$$

We may notice that the second elements of both vectors have the largest values, and they should be in the same cluster as the NMF-based clustering rule suggests. The truth here is that they are in the different clusters. Since our aim is to change their relationship, it does not matter what the NMF-based clustering rule suggests, as long as we can negate their existing relationship. Then according to H_{50} and H_{80}, we modify H_{80} by Scheme 2 to get a new factor vector

$$\widehat{H}_{80} = [\ 2.6134 \quad 0.0496 \quad 0.8012\ ].$$

Running the $\mathcal{K}$-means clustering on $\widetilde{A}^* = \widehat{H}W^*$, $50R80$ is changed to $true$ as the membership of the 80^{th} subject is shifted from C_2 to C_3.

Test 2: Ψ : $(\text{IRIS}, (50, 30), \text{Scheme 2}) \rightarrow \widetilde{A}^*$. $50R30$ is $true$ in the clustering truth of IRIS. We need to find an $\widetilde{A}^*$ to change the relationship to $false$. The basis matrix in an optimal factorization is

$$W^* = \begin{bmatrix} 0 & 1.2589 & 0.9849 & 1.2493 \\ 0.5481 & 0 & 0.7449 & 0.2294 \\ 1.1574 & 0.8411 & 0.1990 & 0 \end{bmatrix}.$$

The corresponding factor vectors are

$$H_{50} = [\ 0 \quad 0.8505 \quad 3.9160\ ],$$

$$H_{30} = [\ 0.0650 \quad 1.0059 \quad 3.6315\ ].$$

We then modify H_{30} by Scheme 2 to get a new factor vector

$$\widehat{H}_{30} = [\ 3.6315 \quad 1.0059 \quad 0.0650\ ].$$

Running the $\mathcal{K}$-means clustering on $\widetilde{A}^* = \widehat{H}W^*$, $50R30$ is changed to $false$ as the membership of the 30^{th} subject is shifted from C_3 to C_2.

Test 3: Ψ : $(\text{IRIS}, (50, 30), (80, 130), \text{Scheme 2}) \rightarrow \widetilde{A}^*$. In this experiment, two confidential relationships are specified as $50R30$ and $80R130$. $50R30$ is $true$ and $80R130$ is $false$ in the truth clustering of IRIS. An $\widetilde{A}^*$ is required to negate these two relationships.

Compared to the previous two experiments, the number of iterations increases. After 16 iterations, an optimal factorization is found as

$$W^* = \begin{bmatrix} 0.2297 & 1.1217 & 1.7552 & 1.5272 \\ 1.3011 & 0.9201 & 0.2528 & 0 \\ 2.5082 & 0.7222 & 2.0846 & 0.5569 \end{bmatrix}.$$

H_{30} and H_{130} are modified based on Scheme 2. The modified factor vectors are

$$\widehat{H}_{30} = [\ 3.0946 \quad 0.0978 \quad 0.2770\ ],$$

$$\widehat{H}_{130} = [\ 0.2837 \quad 2.3770 \quad 0.9609\ ].$$

Then the $\mathcal{K}$-means clustering is run on $\widetilde{A}^* = \widehat{H}W^*$, $50R30$ is changed to $false$ as the membership of the 30^{th} subject is shifted from C_3 to C_2. $80R130$ is changed to $true$ as the membership of the 130^{th} subject is shifted from C_1 to C_2. The solution is not unique, however, the following solution is generated after 77 iterations:

$$W^* = \begin{bmatrix} 1.2481 & 1.5489 & 1.6029 & 1.1703 \\ 2.1535 & 0.2067 & 2.1337 & 0.5170 \\ 1.6640 & 1.1971 & 0.3128 & 0 \end{bmatrix}.$$

The above three experiments indicate the viability of Scheme 2 in changing subject relationships. Similar to the membership hiding, in our experiments, an optimal solution has always been obtainable with zero side effects on memberships.

5.4.4 Relationship Change Using Scheme 3

In this section, the experiments are to examine the effectiveness of Scheme 3 on solving the problem defined as $\Psi : (\text{IRIS}, (x, y), \text{Scheme 3}) \rightarrow \widetilde{A}^*$, where $\widetilde{A}^*$ is an optimal solution without any side effect. In order to make a comparison with Scheme 2, the three experiments are executed under the same conditions as in the previous section.

Test 1: $\Psi : (\text{IRIS}, (50, 80), \text{Scheme 3}) \rightarrow \widetilde{A}^*$. Scheme 3 is carried out to distort the factor vector of H_{80}. An optimal solution is generated after 6 iterations, where the 80^{th} subject is moved from C_2 to C_3 and $50R80$ becomes $true$ in the clustering result on the distorted dataset.

$$W^* = \begin{bmatrix} 2.9125 & 2.3836 & 0.3245 & 0 \\ 0.9380 & 0.1511 & 0.7462 & 0.1220 \\ 0 & 3.5909 & 1.5772 & 2.7915 \end{bmatrix}.$$

The two corresponding factor vectors are

$$H_{50} = [\ 1.2916 \quad 1.3134 \quad 0.0083\ ],$$

$$H_{80} = [\ 0.6076 \quad 4.1582 \quad 0.1534\ ].$$

The distorted H_{80} by Scheme 3 is

$$\widehat{H}_{80} = [\ 0.6076 \quad 0.0083 \quad 1.3134\ ].$$

Test 2: $\Psi : (\text{IRIS}, (50, 30), \text{Scheme } 3) \rightarrow \widetilde{A}^*$. One $\widetilde{A}^*$ is found. By running the $\mathcal{K}$-means clustering on $\widetilde{A}^* = \widehat{H}W^*$, the membership of the 30^{th} subject is shifted from C_3 to C_1. $50R30$ is changed to $false$. The basis matrix in the solution is

$$W^* = \begin{bmatrix} 1.3317 & 0.6553 & 0.3877 & 0 \\ 0.5512 & 1.4534 & 0 & 0.2278 \\ 1.0108 & 0.0979 & 1.9947 & 0.8511 \end{bmatrix}.$$

The two factor vectors are

$$H_{50} = [\ 3.4230 \quad 0.7278 \quad 0.0373\ ],$$

$$H_{30} = [\ 3.1115 \quad 0.7687 \quad 0.1648\ ].$$

We distort H_{30} by Scheme 3 as

$$\widehat{H}_{30} = [\ 0.0373 \quad 0.7687 \quad 3.4230\ ].$$

Test 3: $\Psi : (\text{IRIS}, (50, 30), (80, 130), \text{Scheme } 3) \rightarrow \widetilde{A}^*$. After just 2 iterations, an optimal factorization is produced as

$$W^* = \begin{bmatrix} 1.0557 & 0 & 2.0637 & 0.8439 \\ 0.0048 & 2.5042 & 0 & 0.7697 \\ 1.5353 & 0.8594 & 0.4032 & 0 \end{bmatrix}.$$

The related factor vectors are

$$H_{30} = [\ 0.1599 \quad 0.2412 \quad 2.9743\ ],$$

$$H_{50} = [\ 0.0480 \quad 0.2108 \quad 3.2206\ ],$$

$$H_{80} = [\ 1.1291 \quad 0.0318 \quad 2.9315\ ],$$

$$H_{130} = [\ 2.1085 \quad 0.0393 \quad 3.2880\ ].$$

H_{30} and H_{130} are modified based on Scheme 3. The modified factor vectors are

$$\widehat{H}_{30} = [\ 3.2206 \quad 0.2412 \quad 0.0480\ ],$$

$$\widehat{H}_{130} = [\ 2.1085 \quad 2.9315 \quad 0.0318\ ].$$

The $\mathcal{K}$-means clustering is run on $\widetilde{A}^* = \widehat{H}W^*$, $50R30$ is changed to $false$ as the membership of the 30^{th} subject is shifted from C_3 to C_2. $80R130$ is changed to $true$ as the membership of the 130^{th} subject is shifted from C_1 to C_2.

Through these three experiments, we show that Scheme 3 can change specified relationships as Scheme 2 does. By setting a stopping condition with which the side effects are zero, an optimal solution can be computed and it is not unique. We note that multiple relationship hiding does not necessarily take more time than the single relationship hiding.

5.5 Conclusion

Inspired by the equivalence between NMF and $\mathcal{K}$-means clustering, we present a novel technique to achieve simultaneous realization of data value hiding and pattern hiding. One scheme is proposed to achieve basic data distortion by way of NMF. Three schemes are designed to slightly modify the related factors based on a modified data set generated from NMF. Only through a single sequence of modifications on the original data set can these two contradictory goals be achieved simultaneously.

The attractive advantage of the proposed technique is that a single modified version satisfies both of the two contradictory goals. On one hand, matrix factorization provides a good approximation of the original data sets. That supports our technique for distortion on the data values and achieving comparable mining accuracy. On the other hand, taking advantage of an underlying relationship of the factor vectors with cluster properties in $\mathcal{K}$-means clustering, our technique is capable of hiding confidential patterns while keeping intact nonconfidential patterns. Practically, the merit of our technique is derived from the fact that one released data version can provide dual protection on general data and specified patterns. The strength and efficiency of privacy protection are enhanced. Empirical evaluation on the IRIS data set indicates that our technique is an attractive solution to a combined hiding of data values and data patterns. In particular, an optimal solution without any undesirable side effects on memberships can be easily computed as long as some particular constraints are imposed on the NMF iterations. Our preliminary results show the promising significance of NMF on privacy preserving data mining. More experiments are needed to test the robustness and scalability of this technique on other data sets of larger sizes. In addition, extension of pattern hiding concept from the data mining outcomes to more underlying mechanism is worth more study.

Table 5.2: Mutual information vector M, entropy distortion ED and $\mathcal{K}$-means accuracy.

Method No.	M: I (A^j; $\widetilde{A}^j$) Attr.1	Attr.2	Attr.3	Attr.4	ED	Accuracy (%)
NMF	1.0152	0.1194	1.14176	0.1467	0.2332	77.3
Arp	0	0	0	0	1	58.0
$Arpo$	0.9475	0.3554	0.05613	1.2699	0.5353	12.0
rpA	0	0	0	0	1	35.3
$rpAo$	0.4131	0.1597	0.4227	0.1376	1.1118	34.7
UD	0.4874	0.1797	1.17839	0.4318	0.3593	63.6
ND	0.3386	0.0887	0.78578	0.1452	0.7619	19.3
IRIS	1.8352	0.9933	2.4904	1.6686	0	88.7

Chapter 6

An Improvement on Real-time Performance of SVD-based Model

Besides effectiveness, a good PPDM model should be computationally economical and practically robust for constant and dynamical data sources. First, it should be scalable and computationally applicable to high-dimensional data. Secondly, it should be adaptive to external perturbations, including the addition of new data, the removal of old data and so on. Considering that data streaming is becoming more and more popular in online environments, it is desirable that a good PPDM model make a quick response to external perturbations and produce a new solution in real time.

The structural partition schemes in §3.5 can be used to speed up the SVD-based model. By using the idea of divide-and-conquer, an original data set is partitioned into several parts, then the distortion by the SVD is conducted on each part, a final result is generated by combining all the distorted parts. In this chapter, we will discuss an improved incremental SVD updating algorithm in the context of frequent data updates.

Before discussing the solutions for these two problems, it is helpful to have a look at Table 6.1, a simple comparison on the computation times of four data hiding methods on a 3000×3000 matrix: thin SVD-based, NMF-based, Arp, $Arpo$. The experiments were conducted in MATLAB 7.1. The absolute time do not have much meaning (it is machine-dependent), however, the relative differences in running time would imply an ordering of the speeds of these four methods.

6.1 Performance Improvement Analysis on thin SVD-based Model

Basically, a reduction of computation time for the model provides an increase in speed on the model response time with data updates. Before we discuss possible solutions, it is helpful to look at Table 6.1: a simple comparison of the computation times of four data hiding methods on a 3000×3000 matrix: thin SVD-based, NMF-based, Arp, $Arpo$. The experiment was conducted in MATLAB 7.1. The absolute time does not have much meaning (as it is machine-dependent), however, the relative differences on the running time would imply an ordering of the speeds of these four methods.

In Table 6.1, it is observable that the NMF-based model is significantly faster than the other three methods, with a running time of only about 7 seconds. Arp places second. However, $Arpo$ is very expensive, computationally, due to its orthogonalization operation, while the thin SVD-based model runs much faster than $Arpo$ partly because partial submatrices were used instead of the complete submatrices as in the complete SVD. By using the thin SVD instead of the complete SVD, the running time is significantly decreased. Refer to Table 6.1 to see that the CPU time for the complete SVD is 553.3256 seconds, while the thin SVD only takes 124.3388 seconds. However, compared to the NMF-based model and the Arp model, some improvement is still required for the thin SVD-based method.

Table 6.1: A comparison of computation times.

Methods	the source matrix: 3000×3000				
	NMF-based	thinSVD-based	Arp	$Arpo$	complete SVD
CPU time (s)	7.0501	124.3388	33.2897	325.9687	553.3256
Parameter	$K = 100$	$K = 100$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$K = 3000$
Computation Cost					
ALS NMF	iter$\times O(nmK)$+subIter$\times O(tmK^2 + tnK^2)$				
complete SVD	$O(n^2m + nm^2 + m^3)$				
thin SVD	$O(n^2K + nK^2 + K^3)$				
Arp	$O(nm^2)$				
$Arpo$	$O(nm^2 + n^3)$				

If the data set A is subjected to frequent element additions, and at each time, a new distorted data set $\widetilde{A}$ must be computed, then the thin SVD-based method and $Arpo$ are not scalable. In this chapter, we attempt to speed up the thin SVD-based model since the thin SVD-based method experimentally demonstrates a competitive data mining accuracy compared to the random projection model as shown in Table 6.1, which compares the two models by conducting $\mathcal{K}$-means clustering and classification by SVMlight [40] on the Wisconsin Diagnostic Breast Cancer Database (WDBC) [2]. WDBC contains 569 subjects and 30 real attributes. 357 subjects are in the group of benign, and 212 are in the malignant group. Its best known classification accuracy is 97.5% using 10-fold cross validation [2]. RE is the relative error between A and $\widetilde{A}$

$$\mathsf{RE} = \frac{||A - \widetilde{A}||_F}{||A||_F}. \tag{6.1}$$

Table 6.2: Accuracy comparison of five methods on WDBC.

Methods	RE	Parameter	$\mathcal{K}$-means %	SVMlight %
thinSVD	0.0054	K=4	91.7399	96.1300
Arp	0.9721	σ_r=0.1109	85.2373	95.0791
$Arpo$	1.0727	σ_r=5.8627	84.3585	93.6731
rpA	1.0255	σ_r=0.0100	50.9666	51.1424
$rpoA$	1.3417	σ_r=1.4227	52.5483	53.9543

The thin SVD-based model consists of matrix decompositions primarily from the SVD computation. Even though the algorithm is extremely stable, computing a full SVD is a problem of the order of $\mathcal{O}(nm^2 + n^2m + m^3)$ for a matrix of size n by m. All the data must be processed at one time, and the computation time increases quadratically or cubicly with the addition of new subjects into the database.

The intuitive choice is to only modify the old SVD model to reflect the addition of the new data records, not to re-compute the SVD of the new full data matrix. In the next section, we will introduce an improved incremental SVD updating algorithm to enhance the performance of the thin SVD-based data hiding method.

6.2 Improved Incremental SVD Updating Algorithm

The improved incremental SVD algorithm is based on the updating methods introduced in [87, 73]. This method requires one QR decomposition and one SVD per update. However, these potentially expensive computations are performed on small intermediate matrices, where the computational complexity depends on the size of the update and/or the reduced dimension K, but not on the size of the original data matrix. Depending on subject/attribute addition, there are two updating algorithms: subject-updating and attribute-updating. Essentially, our improved incremental SVD algorithm is based on the algorithms in [73].

6.2.1 Updating Subjects

Let $A \in \mathbb{R}^{n\times m}$ be the data matrix, and $A = [A0; T]^T$, where $A0 \in \mathbb{R}^{t\times m}$ and $T \in \mathbb{R}^{q\times m}$ with q the number of new subjects to be appended, and $n = t + q$.

$$\begin{bmatrix} A0 \\ T \end{bmatrix} \longrightarrow A. \tag{6.2}$$

Assuming the rank-K SVD of $A0$ is known in advance,

$$A0^{(K)} = U_{.(1:K)}\Sigma_K V^T_{.(1:K)}.$$

For simplicity, we use U_K for $U_{.(1:K)}$, and V_K for $V_{.(1:K)}$ in the following. The purpose of the algorithm is to modify the SVD of $A0$ based on the new data, T.

Let $\hat{T} \in \mathbb{R}^{m\times q}$ and

$$\hat{T} = (I_m - V_K V_K^T)T^T. \tag{6.3}$$

Perform the QR decomposition of $\hat{T}$, $Q_T R_T = \hat{T}$, where $Q_T \in \mathbb{R}^{m\times q}$ is orthonormal, and $R_T \in \mathbb{R}^{q\times q}$ is upper triangular. Then

$$\begin{aligned} A &= \begin{bmatrix} A0 \\ T \end{bmatrix} \approx \begin{bmatrix} A0^{(K)} \\ T \end{bmatrix} \\ &= \begin{bmatrix} U_K & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix} \begin{bmatrix} \Sigma_K & \mathbf{0} \\ TV_K & R_T^T \end{bmatrix} \begin{bmatrix} V_K & Q_T \end{bmatrix}^T. \end{aligned} \tag{6.4}$$

Now let $\hat{A} \in \mathbb{R}^{(K+q)\times(K+q)}$ be the matrix defined by

$$\hat{A} = \begin{bmatrix} \Sigma_K & \mathbf{0} \\ TV_K & R_T^T \end{bmatrix}. \tag{6.5}$$

In [73], a complete SVD of $\hat{A}$ is computed. Here, a small improvement is made and a rank-K approximation of $\hat{A}$ is computed instead.

$$\hat{A} \approx \hat{U}_K \hat{\Sigma}_K \hat{V}_K^T \tag{6.6}$$

where $\hat{U}_K \in \mathbb{R}^{(K+q)\times K}$, $\hat{V}_K \in \mathbb{R}^{(K+q)\times K}$ and $\hat{\Sigma}_K \in \mathbb{R}^{K\times K}$. Then the thin SVD of A in K dimensions is

$$A^{(K)} = \left(\begin{bmatrix} U_K & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix} \hat{U}_K \right) \hat{\Sigma}_K \left(\begin{bmatrix} V_K & Q_T \end{bmatrix} \hat{V}_K \right)^T. \tag{6.7}$$

This procedure has a computational complexity of $\mathcal{O}(K^3 + (m+t)K^2 + (m+t)Kq + q^3)$ [73].

6.2.2 Updating Attributes

Let $A \in \mathbb{R}^{n\times m}$ be the data matrix, and $A = [A0, F]$, where $A0 \in \mathbb{R}^{n\times t}$ and $F \in \mathbb{R}^{n\times p}$ with p the number of new attributes to be appended, and $m = t + p$.

$$\begin{bmatrix} A0 & F \end{bmatrix} \longrightarrow A. \tag{6.8}$$

Let $\hat{F} \in \mathbb{R}^{n\times p}$ and

$$\hat{F} = (I_n - U_K U_K^T)F. \tag{6.9}$$

Perform the QR decomposition of $\hat{F}$, $Q_F R_F = \hat{F}$, where $Q_F \in \mathbb{R}^{n\times p}$ is orthonormal, and $R_F \in \mathbb{R}^{p\times p}$ is upper triangular. Then

$$\begin{aligned} A &= \begin{bmatrix} A0 & F \end{bmatrix} \\ &\approx \begin{bmatrix} A0^{(K)} & F \end{bmatrix} \\ &= \begin{bmatrix} U_K & Q_F \end{bmatrix} \begin{bmatrix} \Sigma_K & U_K^T F \\ \mathbf{0} & R_F \end{bmatrix} \begin{bmatrix} V_K^T & \mathbf{0} \\ \mathbf{0} & I_p \end{bmatrix}. \end{aligned} \tag{6.10}$$

Now let $\hat{A} \in \mathbb{R}^{(K+p)\times(K+p)}$ be the matrix defined by

$$\hat{A} = \begin{bmatrix} \Sigma_K & U_K^T F \\ \mathbf{0} & R_F \end{bmatrix}, \tag{6.11}$$

Table 6.3: Run time and RE of two SVD algorithms.

Rows	Incremental thin SVD		Lanczos thin SVD	
	Run time(s)	RE	Run time(s)	RE
3000	218.7799	0.2729	242.9899	0.2720
4000	233.3299	0.2747	321.7100	0.2732
5000	228.0000	0.2758	396.6999	0.2740
6000	231.5399	0.2762	475.7899	0.2742
7000	242.0900	0.2764	568.7299	0.2743
8000	245.0100	0.2767	735.2900	0.2745
9000	244.5699	0.2772	736.9499	0.2749
10000	257.4699	0.2772	825.7900	0.2748

we do the same improvement as updating subjects in (6.6),

$$\hat{A} \approx \bar{U}_K \bar{\Sigma}_K \bar{V}_K^T \tag{6.12}$$

where $\bar{U}_K \in \mathbb{R}^{(K+p)\times K}$, $\bar{V}_K \in \mathbb{R}^{(K+p)\times K}$ and $\bar{\Sigma}_K \in \mathbb{R}^{K\times K}$. Then the thin SVD of A in K dimensions is

$$A^{(K)} = \left([U_K \quad Q_F]\, \bar{U}_K\right)^T \bar{\Sigma}_K \left(\begin{bmatrix} V_K & \mathbf{0} \\ \mathbf{0} & I_p \end{bmatrix} \bar{V}_K\right) \tag{6.13}$$

This procedure has a computational complexity of $\mathcal{O}(K^3 + (m+t)K^2 + (m+t)Kp + p^3)$[73].

6.3 Experiments and Results

Several experiments were conducted in MATLAB 7.1 on synthetic data sets and real data sets to compare the run time, relative error and data mining accuracy between Lanczos SVD and the improved incremental thin SVD.

6.3.1 Subject/Row Updating by Incremental Thin SVD

In this experiment, the incremental thin SVD is examined by adding new subjects. The data set is a synthetic real-value matrix of size of 10000×1000 with the rank of 100. The rank of approximation in the thin SVD is set up to 60. The starting matrix consists of the first 2000 subjects. The rest of the 8000 subjects are repetitively added to the starting matrix for 8

times. At each step, 1000 new subjects are added and a new rank-60 thin SVD is computed by two algorithms: Lanczos SVD and incremental SVD. The experimental results are listed in Table 6.3 and are plotted in Figure 6.1.

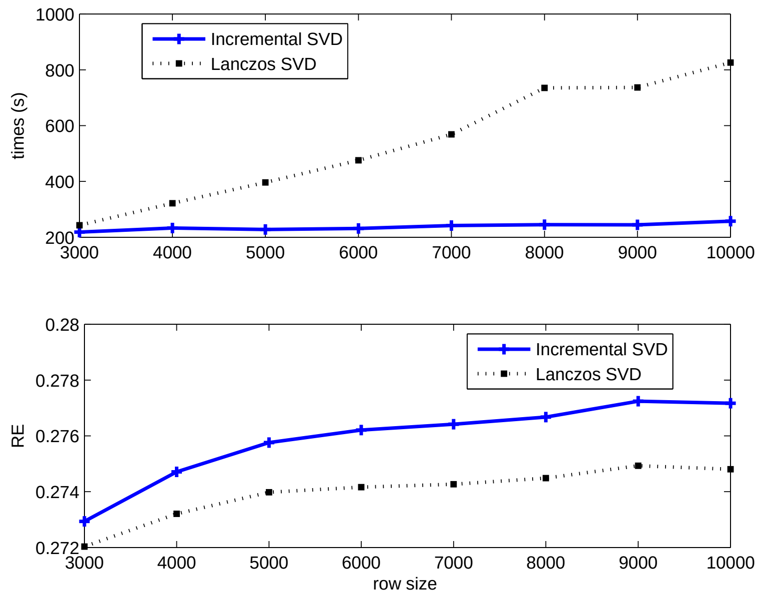


Figure 6.1: Run time and RE of incremental SVD updating (solid line) versus Lanczos SVD (dashed line), as a function of a repetitive addition of 1000 rows for 8 times, on a 10000×1000 random matrix and its rank is 100. The upper figure shows the run time of each addition. The lower figure shows RE.

The relative/approximation error here is defined in (6.1) as RE. If at each step, the augmentation size is 1000, then the run time of the incremental SVD based on the old SVD approximation is much less than that of the Lanczos SVD. At the same time, there is not much effect on the approximation error. For example, if calculating the full matrix by the Lanczos SVD, it takes 825.79 seconds; if updating the SVD from the size of 9000×1000, the run time is 257.47 seconds and is only 31.18% of the run time for the Lanczos thin SVD. Meanwhile, the relative error is 0.2772, which is very similar to 0.2748 by the Lanczos thin SVD.

6.3.2 Attribute/Column Updating by Incremental Thin SVD

A synthetic matrix of the size of 3000×3000 with the rank of 100 is randomly generated in order to examine the performance of attribute updating. The addition of the attributes/columns to the starting matrix of size 3000×80 is repeated 100 times with 22 columns each time. The rank of approximation is set to 80. The comparison is shown in Figure 6.2. For this data set, the advantages of incremental SVD are attractive, considering the cumulative CPU time for these 100 additions is only 29.1719 seconds, and at the same time, the Lanczos SVD requires 104.7306 seconds. Moreover, the approximation errors are very close for the two methods.

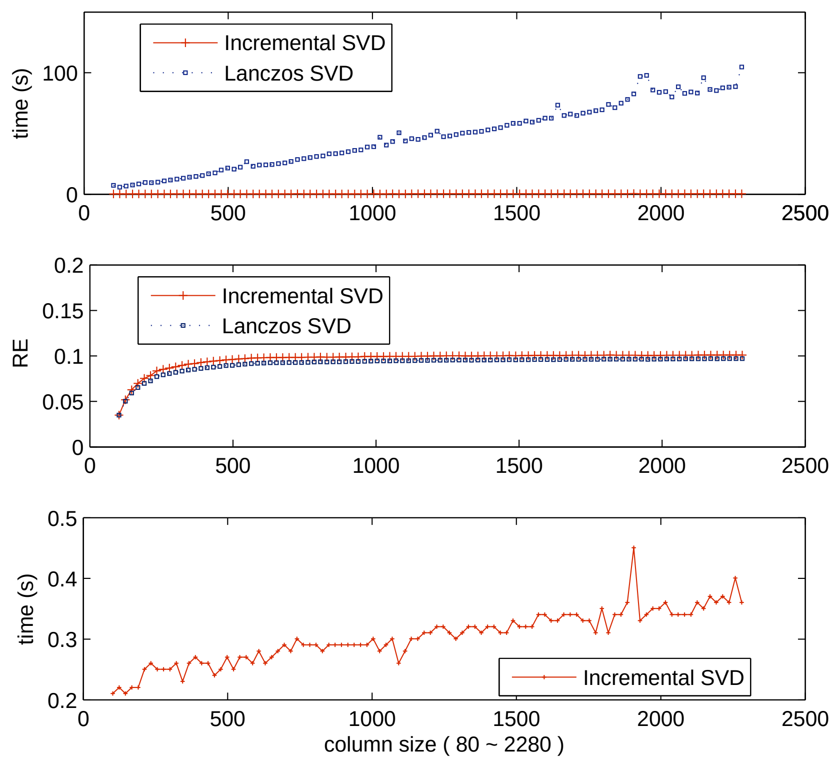


Figure 6.2: Run time and RE of incremental SVD updating (solid line) versus Lanczos SVD (dashed line), as a function of a repetitive addition of 22 columns for 100 times, on a 3000×3000 random matrix of rank 100. The top figure shows the run time of each addition. The middle figure shows RE. The bottom figure is the amplified plot of the run time of the incremental SVD.

6.3.3 Performance Evaluation of the Incremental Thin SVD on WBC

In this experiment, the data mining accuracies are considered in the comparison and the real WBC [2] database is used. WBC consists of 699 subjects and 10 integer-valued attributes. The experiment is designed as follows: the starting matrix is set up to the first 199 subjects/rows and the approximation rank in SVD is 7, then the rest of the 500 subjects are appended repeatedly by 50 rows each time for 10 times. At each time, for both methods, a new rank-7 approximation is computed and its data mining accuracies are evaluated both on SVMlight classification and $\mathcal{K}$-means clustering. Figure 6.3 shows the comparison of run times and approximation errors of the two methods. The time of each step in the incremental SVD is less than the time of the Lanczos SVD on the full data matrix. The difference between the two approximation errors is on the order of 0.001.

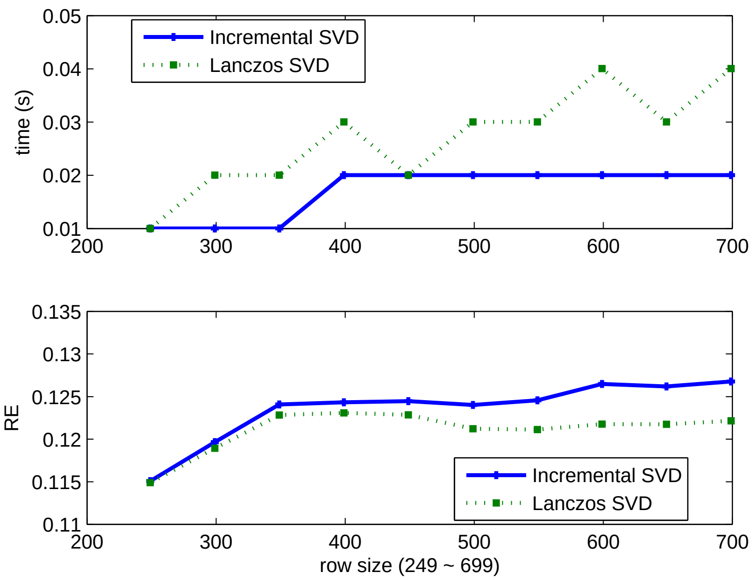


Figure 6.3: Run time and RE of incremental SVD updating (solid line) versus Lanczos SVD (dashed line), as a function of a repetitive addition of 50 rows for 10 times, on WBC. The upper figure shows the run time of each addition. The lower figure shows the RE.

Secondly, by the two methods, the twenty data matrices with row numbers of 249 to 699 are tested in SVMlight classification. Figure 6.4 shows that the accuracies of this data are the same as their counterparts by other methods. The test results imply that incremental SVD does not introduce any observable effect on the classification accuracy.

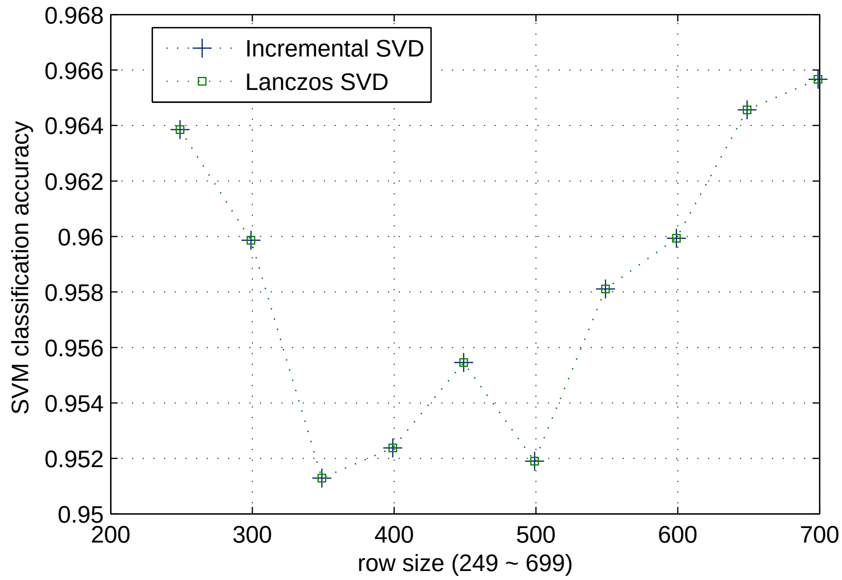


Figure 6.4: SVM classification accuracy of two rank-7 approximations as a function of a repetitive addition of 50 rows. Two methods: incremental SVD updating (solid line) versus Lanczos SVD (dashed line).

Thirdly, $\mathcal{K}$-means clustering is executed on the two rank-7 approximations. One is the rank-7 approximation by the Lanczos SVD of the original WBC and another is the rank-7 approximation by the incremental SVD, which is updated from 199 rows to 699 rows. We examine whether the incremental SVD will affect the clustering quality. Figure 6.5 shows the cluster distributions and Silhouette values for the two approximations of WBC.

In MATLAB 7.1, the Silhouette value, $s(i)$, is used as a measure of how similar the ith subject is to subjects in its own cluster compared to subjects in other clusters. It ranges from -1 to $+1$. It is defined in MATLAB 7.1 code as

$$s(i) = (min(b(i,:),2) - a(i))./max(a(i), min(b(i,:),2))$$

where $a(i)$ is the average distance from the ith point to the other points in its cluster, and $b(i,k)$ is the average distance from the ith point to the points in another cluster k. $./$ is an

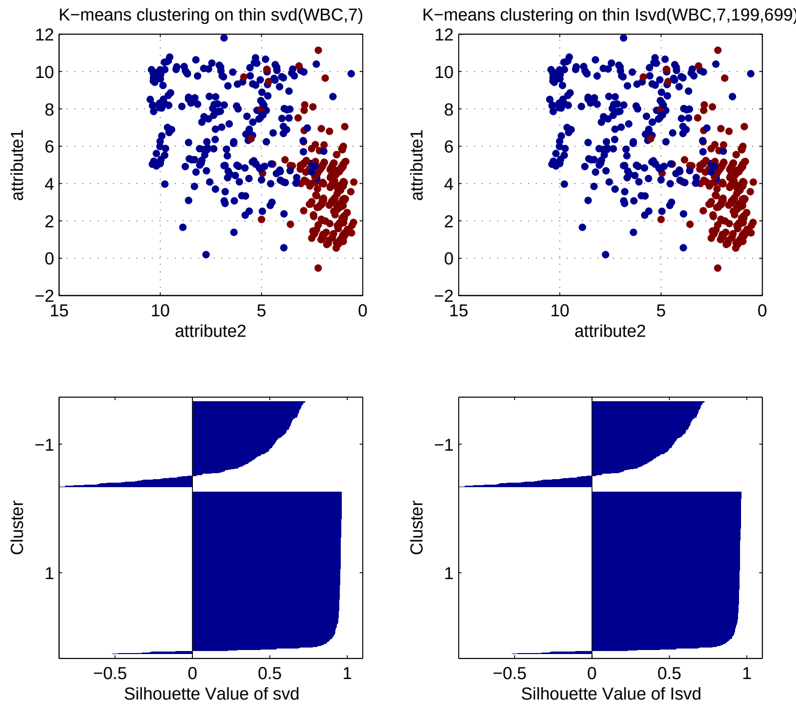


Figure 6.5: Cluster distribution and Silhouette Value of $\mathcal{K}$-means clustering on a rank-7 approximation of WBC, by Lanczos SVD and Incremental SVD, respectively. The two figures on the left are Cluster distribution and Silhouette Value using thin Lanczos SVD. The two figures on the right are cluster distribution and Silhouette Value using thin Incremental SVD, updated from 199 rows to 699 rows, and at each step increased by 50 rows.

element-wise division. In this experiment, the row updating in calculating the thin SVD does not negatively affect clustering.

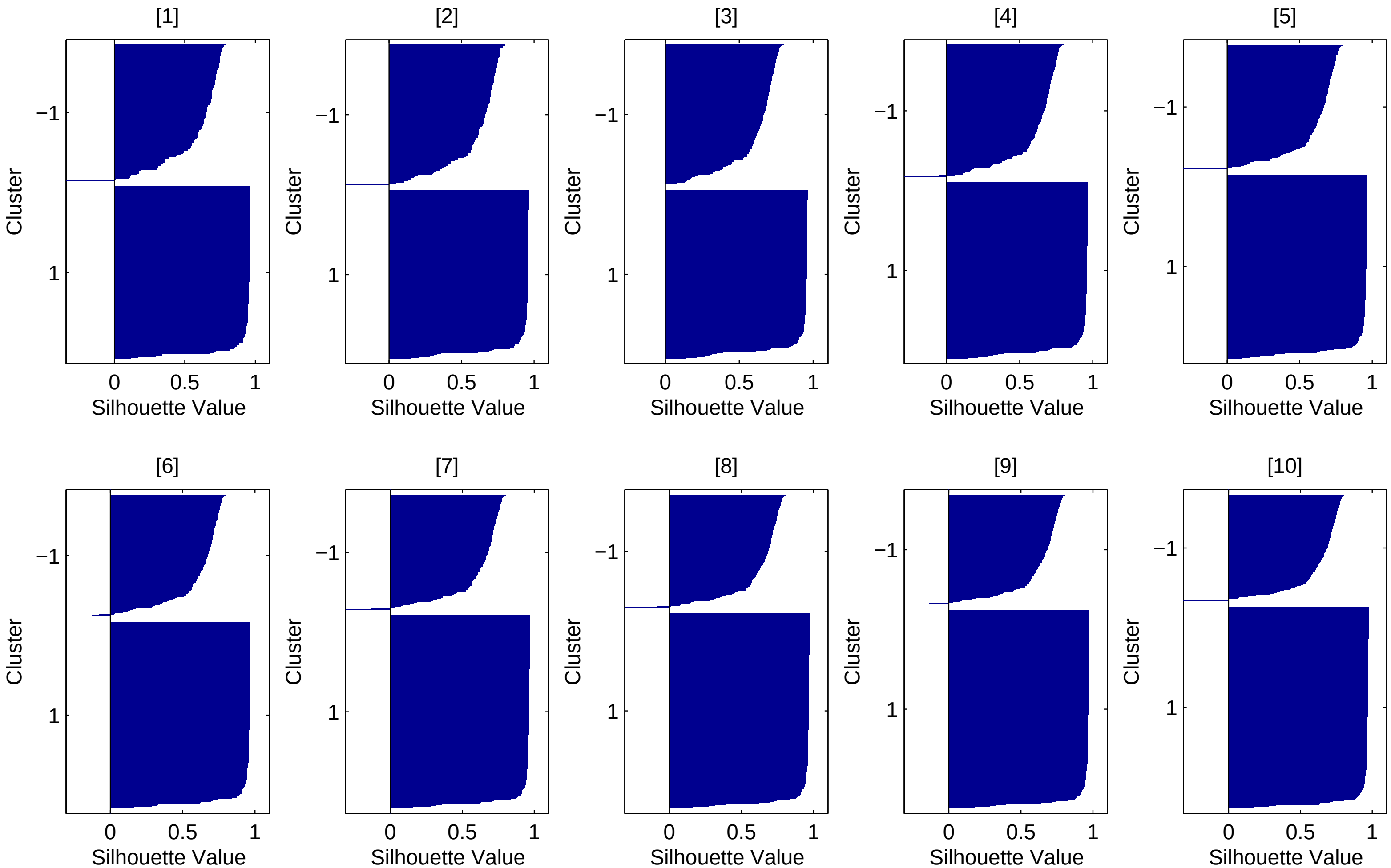


Figure 6.6: Silhouette Values of 10 rank-7 approximations of WBC by the Incremental thin SVD and $\mathcal{K}$-means clustering. The row size is increased from 199 to 699 by adding 50 rows at each step.

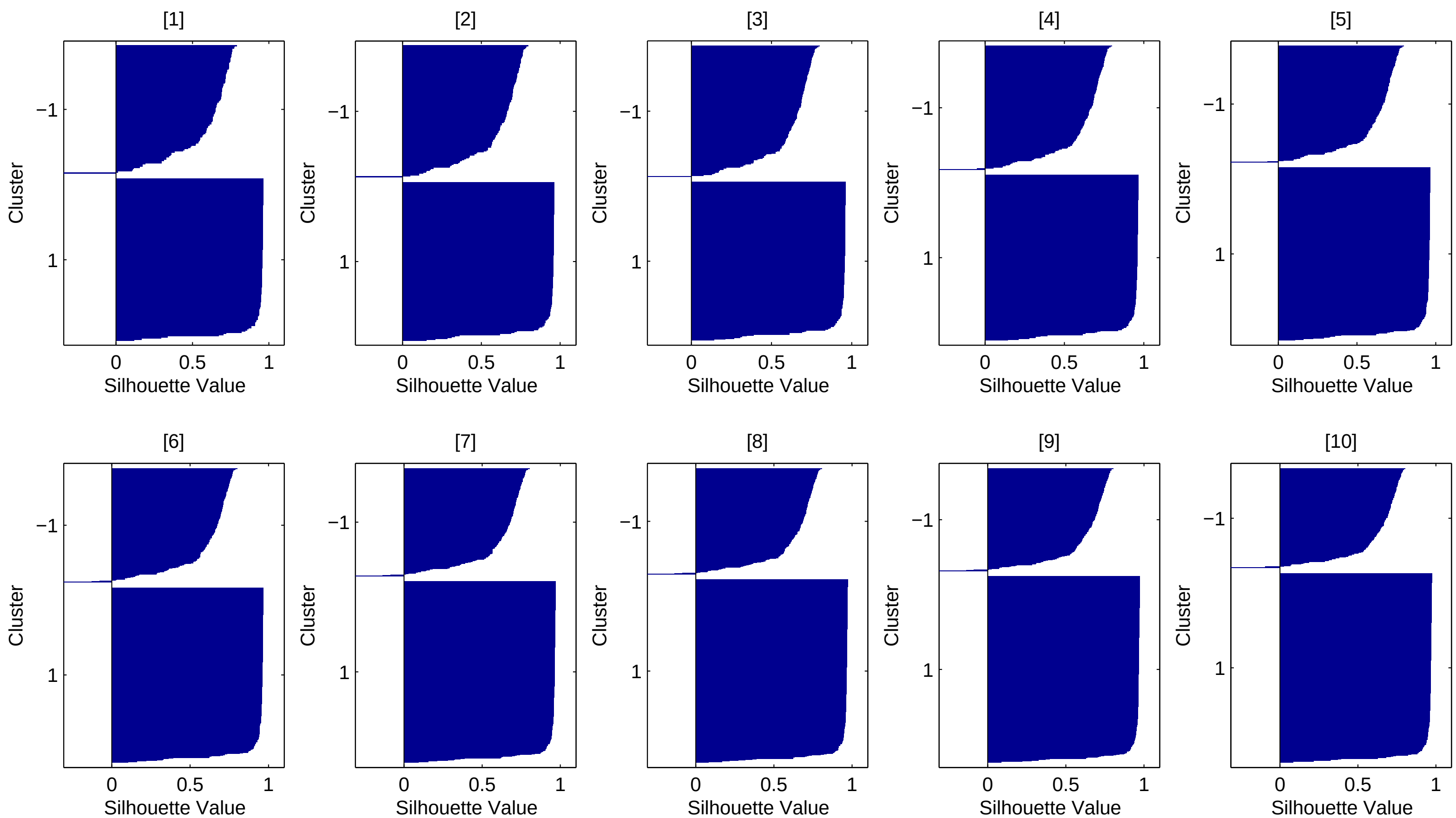


Figure 6.7: Silhouette Values of 10 rank-7 approximations of WBC by thin Lanczos SVD and $\mathcal{K}$-means clustering. The row size is increased from 199 to 699 by adding 50 rows at each step.

6.4 Summary

This chapter has presented an improved SVD-based data value hiding method. The decomposition is derived from updating the previous decomposition solution in an incremental way, instead of starting a new decomposition on the full data matrix. In our experiments, the increase in speed associated with this improved method is encouraging. More importantly, no real differences compared to the traditional SVD-based method are found in the data mining results. This will allow us to address the real-time performance concern with the SVD-based method when a quick response is required for updates of large size. In the meantime, this approach also provides possible support for the application of SVD in On-Line Analytical Processing, which is essential in business data analysis featuring large amounts and frequent growth of data.

Chapter 7

Future Works

Let A be an input data matrix, we can compute two low-rank matrices B and C so that the distance or distortion function between A and BC is minimized, *i.e.*,

$$\min \mathcal{J} = \triangle(A, BC)$$

Many matrix decompositions and fundamental tasks in data mining can be represented by this formulation. This generalization provides greater insight into the data patterns and affords an opportunity to develop new algorithms to discover inherent data patterns if we can impose suitable constraints on A, B and C, or select different distance functions.

Defining $\triangle$ as the Frobenius norm of $(A - BC)$ in the matrix decomposition problem, if B and C are unconstrained, the solution is a rank-k Singular Value Decomposition (SVD). If A, B and C are nonnegative, the decomposition can be formulated as a Nonnegative Matrix Factorization (NMF) problem.

In this dissertation, we have shown that matrix decomposition techniques can be very useful in data hiding or data disclosure control, in the application of privacy preserving data mining. The flexibility of NMF allows us to tailor the factorization process to serve our specific purposes in perturbing datasets.

I plan to continue my efforts on data mining related data processing and knowledge discovery and extend my interests to other new application areas.

Currently, I am continuing my work on **simultaneous data pattern and data value**

hiding. In particular, I will attempt to address the instability of NMF-based methods and improve their scalability by formulating the data pattern hiding requirements as penalty terms embedded into the objective function of NMF. Other problems that I am going to work on include the initialization of NMF, minimization of side effects, and generalization of our methods. Extension of the concepts of dual privacy protection to classification or association rule mining would be another great challenge in my future research agenda. In the meantime, I am interested in developing an inclusive evaluation of our proposed methods. My idea is to use spectral filtering techniques to analyze the reconstruction of the original data from the distorted data from the viewpoint of an attacker. This analysis will provide an important reference on selecting the final data version, considering the non-uniqueness of the solution of NMF. Furthermore, investigation on how to utilize our methods on privacy protection of distributed datasets is also a very interesting topic in my research plan.

In the meantime, I will conduct further study on a multi-basis wavelet-based data hiding strategy which has been proposed for fast data value protection in [52].

Another work in my plan is to study the situation of collaborative analysis, when the data components are from different partners, and different partners have used different data distortion methods to preprocess their datasets for privacy-preserving purposes. It is not clear if a data mining algorithm can be run efficiently on a dataset that has been processed using several different data distortion techniques. This study will be done by analyzing several popular data hiding techniques, to understand their properties, and to see if they have some properties that would make the collaborative analysis difficult. This is actually a very realistic situation, as one cannot in general ask the data owners to prepare the data according to specific requirements. The best way for a data owner to protect the data privacy is probably for the data owner not even to disclose the methods used to distort the datasets, if satisfactory data mining results can be achieved without that information.

In the long term, I will explore **new applications of data matrix decomposition tech-**

niques in the area of management science and economics, and the **privacy and trust issues from electronic collaboration and information sharing**. Powerful matrix computation techniques can be used to process the data and provide a feasible solution only if the collected data can be represented by a matrix. Collaborative prediction can be formalized as a learning problem where the training set is a matrix whose nonzero elements represent known preferences of one user on one item. By adding a low-norm penalty to the distance function $\triangle$ in the matrix decomposition problem, a solution of the matrix decomposition problem can be used to predict user preferences on unobserved items.

The third direction is **the application of higher-order matrix decomposition techniques to multidimensional data**. Images, video and medical data such as CT and MRI are multidimensional data. Information loss is inherent in traditional methods since they reduce multidimensional data to 2-dimensional data in order to apply the classical vector processing methods. Tensor decomposition can be used for medical image analysis by treating the training images as a 3-dimensional cube. I am interested in studying the extension of 2-dimensional SVD and NMF to a higher order and their application to multidimensional data analysis.

Appendices

Appendix A: the Thin SVD-based data modification on WDBC (569×30).

ThinSVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
rank	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
1	0.0872	187.4091	0.0116	0.6000	0.7000	0.0324	0.0978	0.0066	17.4713	85.0615	91.0400
2	0.0341	181.2036	0.0374	0.2667	0.8667	0.0051	0.5204	0.0009	23.2184	83.8313	93.1500
3	0.0188	177.9243	0.0504	0.0000	1.0000	0.0022	1.1386	0.0003	31.9540	86.8190	94.3800
4	0.0054	171.5687	0.0800	0.0000	1.0000	0.0007	12.8134	0.0000	53.3333	91.7399	96.1300
5	0.0022	167.4872	0.1005	0.0000	1.0000	0.0001	34.5962	0.0000	55.1724	90.6854	96.8400
6	0.0012	163.1298	0.1299	0.0000	1.0000	0.0000	53.2860	0.0000	59.3103	91.5641	95.6100
7	0.0006	155.0328	0.1721	0.0000	1.0000	0.0000	75.9419	0.0000	59.7701	91.7399	95.9600
8	0.0004	151.3756	0.1882	0.0000	1.0000	0.0000	85.1809	0.0000	62.7586	91.0369	94.7300
9	0.0003	149.2294	0.2028	0.0000	1.0000	0.0000	91.0004	0.0000	71.2644	89.2794	94.9000
10	0.0002	143.5117	0.2343	0.0000	1.0000	0.0000	96.5080	0.0000	73.5632	89.4552	94.5500
11	0.0001	136.0350	0.2827	0.0000	1.0000	0.0000	98.7277	0.0000	83.2184	91.0369	94.5500
12	0.0001	127.5364	0.3125	0.0000	1.0000	0.0000	99.3960	0.0000	85.7471	92.0914	94.5500
13	0.0001	127.1222	0.3190	0.0000	1.0000	0.0000	99.5916	0.0000	85.5172	91.9156	94.5500
14	0.0000	125.4364	0.3264	0.0000	1.0000	0.0000	99.7661	0.0000	88.7356	92.2671	95.7800
15	0.0000	121.2958	0.3454	0.0000	1.0000	0.0000	99.8564	0.0000	89.1954	91.7399	95.7800
16	0.0000	120.8856	0.3524	0.0000	1.0000	0.0000	99.8997	0.0000	91.0345	91.2127	96.6600
17	0.0000	120.4683	0.3560	0.0000	1.0000	0.0000	99.9196	0.0000	93.7931	91.3884	96.6600
18	0.0000	116.4028	0.3729	0.0000	1.0000	0.0000	99.9641	0.0000	97.0115	92.4429	96.4900
19	0.0000	116.1466	0.3856	0.0000	1.0000	0.0000	99.9814	0.0000	97.7011	92.9701	96.4900
20	0.0000	113.2773	0.4057	0.0000	1.0000	0.0000	99.9839	0.0000	99.0805	93.1459	96.8300
21	0.0000	106.1074	0.4314	0.0000	1.0000	0.0000	99.9889	0.0000	98.6207	93.1459	96.4900
22	0.0000	103.2286	0.4482	0.0000	1.0000	0.0000	99.9913	0.0000	99.0805	93.3216	96.3100
23	0.0000	100.6765	0.4623	0.0000	1.0000	0.0000	99.9913	0.0000	99.5402	92.7944	96.6600
24	0.0000	96.4647	0.4804	0.0000	1.0000	0.0000	99.9963	0.0000	100.0000	92.7944	96.8300
25	0.0000	88.4306	0.5241	0.0000	1.0000	0.0000	99.9975	0.0000	100.0000	92.7944	96.4900
26	0.0000	76.4833	0.5901	0.0000	1.0000	0.0000	99.9988	0.0000	100.0000	92.7944	96.4900
27	0.0000	65.0266	0.6511	0.0000	1.0000	0.0000	100.0000	0.0000	100.0000	92.7944	96.1300
28	0.0000	58.1180	0.6918	0.0000	1.0000	0.0000	100.0000	0.0000	100.0000	92.7944	95.7800
29	0.0000	36.5220	0.8096	0.0000	1.0000	0.0000	100.0000	0.0000	100.0000	92.7944	95.9600
30	0.0000	16.5925	0.9078	0.0000	1.0000	0.0000	100.0000	0.0000	100.0000	92.7944	96.4900

Appendix B: the Uniformly-Noise-Additive data modification on WDBC (569×30).

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise UpperLimit	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.5000	0.0012	166.2086	0.1150	0.3333	0.8667	0.0002	4.1461	0.0013	2.9885	89.9824	0.0000
0.6970	0.0017	170.2118	0.1011	1.0000	0.6667	0.0003	3.2451	0.0019	1.8391	89.8067	0.0000
0.8939	0.0022	171.7759	0.0943	1.9333	0.6667	0.0004	2.4190	0.0024	2.7586	89.4552	94.2900
1.0909	0.0027	170.6878	0.0853	1.8667	0.6667	0.0005	2.1294	0.0029	2.5287	90.5097	0.0000
1.2879	0.0032	171.8076	0.0822	3.8667	0.5333	0.0006	1.8218	0.0035	1.3793	89.8067	0.0000
1.4848	0.0036	174.3714	0.0752	3.6000	0.6333	0.0006	1.6566	0.0040	1.8391	88.5764	94.1300
1.6818	0.0041	174.5074	0.0705	2.8000	0.5667	0.0007	1.4468	0.0046	1.1494	88.5764	0.0000
1.8788	0.0046	175.1059	0.0690	4.1333	0.5667	0.0008	1.3565	0.0050	1.3793	89.6309	0.0000
2.0758	0.0051	177.6034	0.0661	4.0000	0.5333	0.0009	1.1987	0.0056	0.6897	88.0492	93.1200
2.2727	0.0056	176.4928	0.0651	2.7333	0.5667	0.0010	1.1362	0.0062	0.6897	89.1037	0.0000
2.4697	0.0060	177.9808	0.0606	4.2000	0.4667	0.0011	1.0025	0.0066	2.0690	87.8735	0.0000
2.6667	0.0065	178.2206	0.0610	2.2000	0.6333	0.0012	0.9344	0.0072	0.9195	89.2794	91.1100
2.8636	0.0070	178.0285	0.0582	2.4667	0.6000	0.0012	0.9357	0.0078	0.9195	88.0492	0.0000
3.0606	0.0075	178.1506	0.0551	3.8000	0.5000	0.0013	0.8472	0.0083	2.7586	89.1037	0.0000
3.2576	0.0079	178.0599	0.0545	3.8667	0.5667	0.0014	0.8459	0.0088	1.8391	89.8067	91.9500
3.4545	0.0084	178.8164	0.0505	2.8667	0.5333	0.0016	0.7655	0.0093	1.1494	89.2794	0.0000
3.6515	0.0089	178.0778	0.0534	4.6667	0.4667	0.0016	0.7890	0.0098	0.6897	88.5764	0.0000
3.8485	0.0094	178.7267	0.0506	4.4000	0.4667	0.0017	0.6770	0.0104	0.4598	88.2250	91.7800
4.0455	0.0099	178.8651	0.0474	5.1333	0.5000	0.0018	0.6374	0.0109	0.9195	88.2250	0.0000
4.2424	0.0103	180.2698	0.0452	5.0000	0.4333	0.0019	0.6126	0.0114	1.1494	50.0879	0.0000
4.4394	0.0108	180.1731	0.0441	3.1333	0.4667	0.0019	0.6281	0.0120	1.1494	88.9279	92.6200
4.6364	0.0113	180.8956	0.0444	2.9333	0.5667	0.0020	0.6015	0.0125	0.2299	88.0492	0.0000
4.8333	0.0118	180.3772	0.0454	4.2667	0.5000	0.0021	0.5477	0.0131	2.2989	87.1705	0.0000
5.0303	0.0123	181.7323	0.0404	3.8000	0.5000	0.0022	0.5675	0.0137	2.5287	88.0492	92.7900
5.2273	0.0127	181.8731	0.0407	5.4667	0.4000	0.0023	0.5353	0.0140	1.3793	89.4552	0.0000
5.4242	0.0133	179.6799	0.0428	2.4667	0.5000	0.0023	0.5347	0.0148	0.2299	88.4007	0.0000
5.6212	0.0138	181.9506	0.0374	5.4000	0.4333	0.0025	0.5508	0.0153	0.6897	88.0492	93.6200
5.8182	0.0142	181.2608	0.0400	4.8667	0.5000	0.0026	0.5081	0.0157	0.9195	87.6977	0.0000
6.0152	0.0146	181.8029	0.0386	4.0000	0.5000	0.0027	0.4852	0.0162	1.1494	88.5764	0.0000
6.2121	0.0152	183.4636	0.0377	4.2000	0.4667	0.0029	0.4437	0.0167	0.6897	88.5764	90.6000

Continued on next page

Appendix B **– continued from previous page**

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise UpperLimit	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	$\mathcal{K}$**-means**	**SVMlight**
6.4091	0.0155	182.9377	0.0378	4.4667	0.4000	0.0029	0.4635	0.0172	0.2299	86.6432	0.0000
6.6061	0.0160	183.0972	0.0352	5.5333	0.4667	0.0030	0.4208	0.0177	0.4598	88.9279	0.0000
6.8030	0.0167	182.6395	0.0366	5.2667	0.4333	0.0030	0.4128	0.0186	0.6897	86.6432	91.4400
7.0000	0.0170	182.9768	0.0366	2.3333	0.5333	0.0032	0.4084	0.0189	0.4598	87.6977	0.0000
7.1970	0.0174	182.7706	0.0325	4.1333	0.4667	0.0033	0.3960	0.0193	0.9195	87.5220	0.0000
7.3939	0.0181	181.9934	0.0332	2.6667	0.5333	0.0034	0.3942	0.0201	1.3793	87.6977	91.1000
7.5909	0.0185	183.3904	0.0323	4.5333	0.4333	0.0035	0.3583	0.0206	1.6092	88.0492	0.0000
7.7879	0.0190	182.9133	0.0307	6.8000	0.3667	0.0037	0.3676	0.0210	0.9195	88.4007	0.0000
7.9848	0.0195	184.5680	0.0322	4.0000	0.4333	0.0037	0.3447	0.0216	0.2299	88.9279	91.4400
8.1818	0.0199	183.3023	0.0306	5.2000	0.4333	0.0038	0.3923	0.0221	1.3793	88.2250	0.0000
8.3788	0.0205	183.1279	0.0293	5.6667	0.4667	0.0038	0.3546	0.0227	0.9195	86.8190	0.0000
8.5758	0.0208	183.0692	0.0299	5.3333	0.4333	0.0039	0.3404	0.0233	1.1494	86.9947	90.1000
8.7727	0.0214	183.1933	0.0298	6.5333	0.4000	0.0039	0.3515	0.0238	1.3793	88.4007	0.0000
8.9697	0.0218	182.7477	0.0272	5.8667	0.5000	0.0041	0.3441	0.0242	0.2299	89.8067	0.0000
9.1667	0.0223	184.0634	0.0293	5.3333	0.5000	0.0041	0.3366	0.0249	0.6897	86.6432	91.2700
9.3636	0.0228	185.6992	0.0279	4.8667	0.4667	0.0043	0.3434	0.0254	1.1494	86.6432	0.0000
9.5606	0.0232	183.4057	0.0287	4.6667	0.5000	0.0044	0.3218	0.0256	1.1494	87.3462	0.0000
9.7576	0.0238	183.9930	0.0280	4.1333	0.4333	0.0045	0.3181	0.0265	0.4598	87.3462	88.7500
9.9545	0.0244	183.9216	0.0274	4.2667	0.4667	0.0047	0.3366	0.0271	0.4598	87.5220	0.0000
10.1515	0.0249	184.4128	0.0261	5.2667	0.5333	0.0049	0.2803	0.0276	0.4598	88.0492	0.0000
10.3485	0.0251	183.8514	0.0281	5.5333	0.3667	0.0049	0.2618	0.0280	1.3793	88.2250	90.1000
10.5455	0.0259	181.5979	0.0281	6.8000	0.4000	0.0051	0.2673	0.0290	0.9195	86.4675	0.0000
10.7424	0.0262	184.8122	0.0262	7.0000	0.4667	0.0050	0.2871	0.0292	1.6092	87.5220	0.0000
10.9394	0.0267	183.5497	0.0279	6.6000	0.4000	0.0052	0.2618	0.0298	0.4598	87.1705	90.4400
11.1364	0.0273	183.4668	0.0258	6.1333	0.4667	0.0054	0.2581	0.0304	1.8391	86.6432	0.0000
11.3333	0.0276	183.3606	0.0241	3.8000	0.5000	0.0055	0.2649	0.0305	0.9195	88.9279	0.0000
11.5303	0.0282	184.0281	0.0273	4.4667	0.4333	0.0055	0.2655	0.0313	1.1494	88.4007	89.6000
11.7273	0.0285	184.8355	0.0247	4.0000	0.4667	0.0057	0.2420	0.0318	0.4598	33.7434	0.0000
11.9242	0.0291	185.2414	0.0230	5.6000	0.4000	0.0060	0.2302	0.0324	0.4598	87.1705	0.0000

Continued on next page

Appendix B **– continued from previous page**

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise UpperLimit	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
12.1212	0.0296	183.2047	0.0239	6.4667	0.4000	0.0059	0.2562	0.0331	0.9195	88.5764	90.6000
12.3182	0.0302	183.4226	0.0255	5.6667	0.4000	0.0061	0.2011	0.0337	1.6092	72.5835	0.0000
12.5152	0.0305	184.2375	0.0242	5.9333	0.4000	0.0062	0.2302	0.0340	1.3793	54.3058	0.0000
12.7121	0.0309	183.8168	0.0253	7.2000	0.3667	0.0063	0.2475	0.0344	0.6897	86.9947	90.4400
12.9091	0.0315	185.4111	0.0215	6.4000	0.4333	0.0064	0.2246	0.0352	0.4598	89.6309	0.0000
13.1061	0.0318	185.0987	0.0226	7.0667	0.4333	0.0066	0.2370	0.0354	1.8391	89.1037	0.0000
13.3030	0.0327	187.1018	0.0228	6.3333	0.4333	0.0064	0.2358	0.0365	0.0000	86.2917	92.9500
13.5000	0.0331	184.4030	0.0229	3.9333	0.4667	0.0065	0.2296	0.0371	1.3793	85.9402	0.0000
13.6970	0.0332	184.7971	0.0227	5.4667	0.4667	0.0069	0.2147	0.0372	0.9195	85.5888	0.0000
13.8939	0.0339	184.5234	0.0219	5.6667	0.4000	0.0069	0.2222	0.0377	0.4598	87.6977	91.7800
14.0909	0.0343	184.4776	0.0209	6.2000	0.3667	0.0072	0.2005	0.0381	0.2299	87.1705	0.0000
14.2879	0.0347	185.6205	0.0205	6.6000	0.4000	0.0071	0.2147	0.0387	0.6897	87.5220	0.0000
14.4848	0.0354	185.1784	0.0203	4.5333	0.4333	0.0072	0.2129	0.0395	1.3793	52.5483	90.9400
14.6818	0.0357	186.5571	0.0203	4.1333	0.5333	0.0073	0.2110	0.0400	0.6897	87.1705	0.0000
14.8788	0.0362	185.4887	0.0217	5.2000	0.4000	0.0075	0.2017	0.0402	0.4598	87.8735	0.0000
15.0758	0.0366	184.4733	0.0206	5.2000	0.4333	0.0076	0.2129	0.0408	0.9195	86.9947	91.9500
15.2727	0.0374	185.3475	0.0196	6.4000	0.4333	0.0077	0.2172	0.0419	0.4598	87.3462	0.0000
15.4697	0.0377	185.2288	0.0220	5.7333	0.5000	0.0077	0.2030	0.0418	0.6897	88.5764	0.0000
15.6667	0.0382	186.5018	0.0218	5.7333	0.3333	0.0080	0.2085	0.0426	0.9195	86.9947	89.4300
15.8636	0.0387	186.0360	0.0192	4.3333	0.4667	0.0083	0.1677	0.0433	0.4598	86.1160	0.0000
16.0606	0.0391	186.2168	0.0213	5.5333	0.4667	0.0081	0.1993	0.0435	0.4598	71.3533	0.0000
16.2576	0.0399	185.6729	0.0195	5.8667	0.4000	0.0085	0.1968	0.0446	0.9195	88.2250	90.1000
16.4545	0.0403	185.7472	0.0195	5.2000	0.4333	0.0083	0.1739	0.0451	1.3793	87.1705	0.0000
16.6515	0.0406	185.9144	0.0186	5.8667	0.3667	0.0088	0.1813	0.0453	0.2299	86.6432	0.0000
16.8485	0.0410	184.7092	0.0219	4.8667	0.4000	0.0087	0.1955	0.0458	0.6897	85.4130	90.9300
17.0455	0.0415	185.9636	0.0184	4.8667	0.5000	0.0088	0.2160	0.0463	0.4598	86.4675	0.0000
17.2424	0.0422	184.6250	0.0204	5.4667	0.4333	0.0091	0.1621	0.0471	1.1494	84.3585	0.0000
17.4394	0.0425	185.0998	0.0194	4.9333	0.4667	0.0090	0.1671	0.0475	0.4598	86.9947	89.9200
17.6364	0.0430	185.9390	0.0190	5.5333	0.4333	0.0093	0.1782	0.0480	1.3793	85.7645	0.0000

Continued on next page

Appendix B – continued from previous page

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise UpperLimit	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
17.8333	0.0436	185.7247	0.0187	5.0000	0.4333	0.0093	0.1559	0.0489	1.8391	88.2250	0.0000
18.0303	0.0441	185.4866	0.0182	4.3333	0.4000	0.0099	0.1429	0.0494	0.9195	86.2917	89.0900
18.2273	0.0448	184.9370	0.0185	5.8667	0.4333	0.0098	0.1906	0.0505	1.1494	86.8190	0.0000
18.4242	0.0449	185.7288	0.0185	5.0000	0.4000	0.0099	0.1671	0.0500	0.0000	49.3849	0.0000
18.6212	0.0456	187.6492	0.0159	6.5333	0.4333	0.0100	0.1696	0.0513	0.4598	85.9402	0.0000
18.8182	0.0460	187.2018	0.0180	4.2667	0.4667	0.0100	0.1516	0.0514	0.4598	40.9490	0.0000
19.0152	0.0464	185.3885	0.0184	6.9333	0.3667	0.0101	0.1485	0.0520	0.6897	88.9279	0.0000
19.2121	0.0467	186.4606	0.0170	5.8667	0.4333	0.0102	0.1708	0.0521	1.3793	85.9402	0.0000
19.4091	0.0475	186.7459	0.0170	6.0000	0.4000	0.0106	0.1603	0.0531	1.1494	52.3726	0.0000
19.6061	0.0478	184.1549	0.0180	5.8000	0.3667	0.0107	0.1733	0.0536	1.1494	50.4394	0.0000
19.8030	0.0483	185.1120	0.0167	4.6000	0.4333	0.0109	0.1628	0.0543	0.9195	85.7645	0.0000
20.0000	0.0492	185.9548	0.0175	6.0667	0.3667	0.0110	0.1355	0.0552	0.9195	88.5764	0.0000

Appendix C: the Normal-Noise-Additive data modification on WDBC (569×30).

Normal	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise σ	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	$\mathcal{K}$**-means**	**SVMlight**
0.2000	0.0008	168.1572	0.1062	0.0667	0.9333	0.0003	3.3850	0.0000	13.5632	89.9824	0.0000
0.3495	0.0015	171.4744	0.0859	2.1333	0.6000	0.0005	2.0211	0.0001	10.8046	87.8735	0.0000
0.4990	0.0021	173.9250	0.0756	3.5333	0.6000	0.0008	1.4481	0.0001	10.3448	88.0492	92.1100
0.6485	0.0027	176.5659	0.0667	4.5333	0.5000	0.0010	1.1040	0.0002	6.4368	88.9279	0.0000
0.7980	0.0034	177.3175	0.0613	2.6667	0.5667	0.0012	0.9542	0.0002	7.3563	86.9947	0.0000
0.9475	0.0040	178.6974	0.0556	1.2667	0.5333	0.0014	0.7822	0.0003	6.2069	87.1705	92.9500
1.0970	0.0047	176.8637	0.0524	2.6667	0.5667	0.0016	0.8026	0.0003	4.3678	88.5764	0.0000
1.2465	0.0053	180.3332	0.0485	3.4000	0.6000	0.0019	0.6597	0.0004	4.5977	87.6977	0.0000
1.3960	0.0059	180.1728	0.0450	6.2000	0.4667	0.0022	0.5854	0.0004	6.4368	86.6432	90.1000
1.5455	0.0066	180.9166	0.0430	6.1333	0.4333	0.0024	0.5192	0.0003	3.9080	87.6977	0.0000
1.6949	0.0072	181.4540	0.0426	4.2000	0.4333	0.0026	0.5013	0.0005	4.5977	87.5220	0.0000
1.8444	0.0078	182.2663	0.0401	3.5333	0.5667	0.0028	0.4982	0.0004	5.2874	88.2250	91.1000
1.9939	0.0084	181.3025	0.0384	6.2000	0.4333	0.0031	0.4121	0.0005	4.5977	86.6432	0.0000
2.1434	0.0090	183.9817	0.0383	4.4667	0.5000	0.0034	0.4041	0.0005	6.4368	88.5764	0.0000
2.2929	0.0096	181.5830	0.0357	4.4000	0.5000	0.0035	0.4035	0.0005	4.1379	88.0492	92.9500
2.4424	0.0103	183.3939	0.0320	4.4667	0.4333	0.0039	0.3571	0.0006	3.4483	86.6432	0.0000
2.5919	0.0109	182.2834	0.0327	3.5333	0.4667	0.0042	0.3162	0.0006	6.2069	85.5888	0.0000
2.7414	0.0117	183.5636	0.0303	6.7333	0.4333	0.0046	0.2816	0.0005	2.5287	87.1705	90.6000
2.8909	0.0122	183.7992	0.0309	5.5333	0.5000	0.0048	0.2927	0.0008	3.6782	86.6432	0.0000
3.0404	0.0127	183.4800	0.0315	6.2667	0.4333	0.0048	0.2983	0.0007	2.7586	88.0492	0.0000
3.1899	0.0135	183.6313	0.0298	5.1333	0.4333	0.0053	0.3088	0.0008	3.4483	87.6977	89.4200
3.3394	0.0142	184.9861	0.0266	5.0000	0.5000	0.0059	0.2797	0.0007	2.9885	87.8735	0.0000
3.4889	0.0146	184.8262	0.0264	4.5333	0.4667	0.0059	0.2754	0.0008	3.2184	87.1705	0.0000
3.6384	0.0154	184.1595	0.0263	8.2667	0.3667	0.0062	0.2599	0.0009	3.6782	86.1160	89.5900
3.7879	0.0160	182.9250	0.0248	6.6667	0.4333	0.0065	0.2469	0.0010	3.2184	88.2250	0.0000
3.9374	0.0165	185.0284	0.0258	4.9333	0.5000	0.0067	0.2389	0.0011	2.5287	87.5220	0.0000
4.0869	0.0173	186.5916	0.0241	5.4000	0.4000	0.0071	0.2246	0.0011	1.3793	86.6432	92.4400
4.2364	0.0177	184.8716	0.0237	4.3333	0.4667	0.0072	0.2234	0.0011	0.9195	87.6977	0.0000
4.3859	0.0184	184.5377	0.0232	5.8667	0.4333	0.0078	0.2123	0.0010	3.9080	86.9947	0.0000
4.5354	0.0192	184.3152	0.0224	5.8667	0.4000	0.0080	0.1894	0.0010	1.6092	87.6977	88.5900

Continued on next page

Appendix C – continued from previous page

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise σ	RE	RP	RK	CP	CK	DistVal	Dist Maintain	CorrVal	Corr Maintain	$\mathcal{K}$-means	SVMlight
4.6848	0.0198	186.4554	0.0223	5.2000	0.4333	0.0088	0.1714	0.0011	2.2989	85.9402	0.0000
4.8343	0.0206	185.5057	0.0219	5.1333	0.4333	0.0085	0.2048	0.0016	1.8391	87.5220	0.0000
4.9838	0.0209	184.9203	0.0213	6.1333	0.4667	0.0093	0.2055	0.0012	2.2989	87.3462	90.6000
5.1333	0.0218	185.1511	0.0200	6.8000	0.3667	0.0096	0.1714	0.0017	2.0690	89.2794	0.0000
5.2828	0.0224	185.7903	0.0182	5.8000	0.4000	0.0101	0.1529	0.0016	0.9195	85.5888	0.0000
5.4323	0.0230	184.4771	0.0193	7.0000	0.3667	0.0099	0.1739	0.0012	1.8391	86.4675	90.7700
5.5818	0.0236	184.5754	0.0188	6.9333	0.4000	0.0105	0.1566	0.0011	1.6092	86.8190	0.0000
5.7313	0.0243	186.0750	0.0189	3.4667	0.4667	0.0109	0.1646	0.0015	2.2989	87.5220	0.0000
5.8808	0.0249	186.3475	0.0203	5.9333	0.4333	0.0112	0.1504	0.0015	1.3793	87.3462	90.2700
6.0303	0.0256	184.8309	0.0190	6.2667	0.4000	0.0117	0.1708	0.0016	1.8391	86.9947	0.0000
6.1798	0.0258	187.0081	0.0183	4.9333	0.4333	0.0118	0.1590	0.0016	1.1494	88.0492	0.0000
6.3293	0.0270	184.8475	0.0181	5.9333	0.4000	0.0119	0.1671	0.0019	1.1494	86.9947	89.9300
6.4788	0.0274	187.1366	0.0157	4.4000	0.4000	0.0129	0.1473	0.0021	0.9195	86.8190	0.0000
6.6283	0.0281	187.6190	0.0162	5.4667	0.4000	0.0132	0.1287	0.0016	1.6092	87.6977	0.0000
6.7778	0.0288	185.4647	0.0170	4.6667	0.4333	0.0136	0.1225	0.0014	1.3793	87.6977	88.5900
6.9273	0.0292	186.1434	0.0163	5.2000	0.4333	0.0139	0.1102	0.0019	2.0690	86.9947	0.0000
7.0768	0.0301	186.2676	0.0178	6.6000	0.4000	0.0147	0.1250	0.0019	0.9195	86.8190	0.0000
7.2263	0.0301	185.3254	0.0170	6.0667	0.4000	0.0146	0.1275	0.0018	1.6092	85.7645	90.4300
7.3758	0.0311	185.6057	0.0159	5.0000	0.4000	0.0149	0.1337	0.0019	1.1494	86.4675	0.0000
7.5253	0.0315	186.8350	0.0152	5.6667	0.4000	0.0153	0.1163	0.0022	2.2989	87.3462	0.0000
7.6747	0.0323	185.7571	0.0160	6.2000	0.4000	0.0158	0.1225	0.0018	1.3793	86.1160	87.0800
7.8242	0.0328	187.0280	0.0136	6.8000	0.3667	0.0167	0.1021	0.0020	0.9195	88.5764	0.0000
7.9737	0.0339	188.4867	0.0131	4.6000	0.5000	0.0164	0.1250	0.0020	2.0690	88.5764	0.0000
8.1232	0.0342	185.7402	0.0145	6.6667	0.4000	0.0168	0.0972	0.0018	1.6092	84.7100	89.2600
8.2727	0.0349	186.9442	0.0141	4.2000	0.4333	0.0175	0.1170	0.0019	0.6897	87.3462	0.0000
8.4222	0.0354	186.0205	0.0142	6.4667	0.3333	0.0176	0.1083	0.0018	1.8391	86.1160	0.0000
8.5717	0.0364	186.4393	0.0136	6.3333	0.4000	0.0184	0.0990	0.0025	2.9885	87.3462	89.0900
8.7212	0.0366	186.4989	0.0129	7.5333	0.4333	0.0188	0.0941	0.0026	0.4598	87.6977	0.0000
8.8707	0.0374	186.0353	0.0134	5.9333	0.4667	0.0194	0.1015	0.0019	1.6092	87.3462	0.0000

Continued on next page

Appendix C – continued from previous page

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise σ	RE	RP	RK	CP	CK	DistVal	Dist Maintain	CorrVal	Corr Maintain	$\mathcal{K}$-means	SVMlight
9.0202	0.0384	186.3361	0.0120	6.4667	0.4667	0.0195	0.0953	0.0025	1.3793	88.0492	88.4200
9.1697	0.0389	186.2984	0.0141	4.7333	0.4000	0.0204	0.0978	0.0020	1.3793	85.5888	0.0000
9.3192	0.0393	186.6776	0.0145	4.3333	0.4333	0.0204	0.0978	0.0024	0.9195	87.3462	0.0000
9.4687	0.0400	185.8134	0.0143	6.3333	0.3667	0.0205	0.0885	0.0029	1.6092	88.0492	89.7600
9.6182	0.0407	186.6748	0.0123	3.5333	0.4333	0.0213	0.0959	0.0024	0.2299	86.6432	0.0000
9.7677	0.0418	187.0428	0.0127	5.4000	0.3667	0.0223	0.0990	0.0028	1.3793	86.8190	0.0000
9.9172	0.0422	187.7291	0.0127	6.3333	0.4333	0.0225	0.0866	0.0029	1.1494	86.8190	88.0900
10.0667	0.0428	187.5377	0.0126	6.6000	0.3667	0.0230	0.1052	0.0022	1.3793	85.5888	0.0000
10.2162	0.0433	188.5325	0.0129	8.4000	0.3667	0.0227	0.0972	0.0024	1.3793	87.8735	0.0000
10.3657	0.0433	187.1402	0.0128	7.4667	0.3333	0.0236	0.0885	0.0029	0.6897	86.2917	88.2600
10.5152	0.0447	187.9971	0.0135	5.1333	0.4000	0.0242	0.0934	0.0027	1.1494	86.9947	0.0000
10.6646	0.0451	187.9530	0.0125	5.6000	0.4000	0.0241	0.0829	0.0028	1.6092	86.9947	0.0000
10.8141	0.0459	187.5963	0.0123	6.4667	0.3667	0.0248	0.0681	0.0028	1.6092	86.2917	87.5800
10.9636	0.0463	185.3135	0.0115	6.2667	0.3667	0.0256	0.0823	0.0026	1.1494	85.5888	0.0000
11.1131	0.0469	187.5773	0.0128	8.2667	0.4000	0.0253	0.0650	0.0032	0.9195	88.5764	0.0000
11.2626	0.0477	188.2094	0.0114	5.6000	0.4667	0.0260	0.0842	0.0029	0.9195	86.4675	89.0900
11.4121	0.0483	187.9866	0.0109	5.6000	0.3333	0.0269	0.0743	0.0033	0.9195	87.6977	0.0000
11.5616	0.0490	187.2088	0.0112	4.0667	0.4333	0.0276	0.0792	0.0027	0.4598	87.5220	0.0000
11.7111	0.0495	188.3050	0.0101	5.7333	0.3667	0.0272	0.0668	0.0035	0.4598	85.7645	87.0800
11.8606	0.0501	186.1424	0.0112	5.0667	0.4000	0.0282	0.0835	0.0026	0.6897	86.1160	0.0000
12.0101	0.0507	185.4858	0.0115	7.1333	0.3667	0.0284	0.0860	0.0034	0.6897	86.8190	0.0000
12.1596	0.0516	186.1804	0.0104	6.6667	0.3667	0.0303	0.0804	0.0034	0.2299	86.9947	87.0800
12.3091	0.0518	187.6045	0.0105	6.2667	0.3667	0.0299	0.0829	0.0036	0.6897	87.3462	0.0000
12.4586	0.0527	188.5442	0.0100	5.2000	0.3667	0.0305	0.0873	0.0028	1.3793	85.7645	0.0000
12.6081	0.0531	186.8039	0.0112	4.5333	0.4667	0.0312	0.0699	0.0022	0.4598	86.4675	88.9200
12.7576	0.0538	186.6969	0.0113	6.6667	0.3667	0.0306	0.0774	0.0028	1.6092	87.6977	0.0000
12.9071	0.0544	187.1977	0.0096	5.3333	0.4333	0.0315	0.0767	0.0029	1.3793	86.4675	0.0000
13.0566	0.0550	189.4117	0.0100	6.4667	0.3333	0.0321	0.0786	0.0032	0.6897	86.4675	86.7500
13.2061	0.0555	188.1351	0.0112	5.2000	0.3333	0.0318	0.0774	0.0041	0.4598	86.4675	0.0000

Continued on next page

Appendix C – continued from previous page

Uniformly	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
Noise σ	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
13.3556	0.0560	187.3316	0.0110	6.0667	0.3667	0.0336	0.0650	0.0031	0.2299	86.8190	0.0000
13.5051	0.0567	187.8482	0.0107	6.4000	0.3667	0.0336	0.0761	0.0036	0.9195	85.0615	87.9200
13.6545	0.0582	188.4478	0.0101	6.7333	0.4000	0.0352	0.0619	0.0038	0.0000	86.6432	0.0000
13.8040	0.0588	188.1045	0.0107	7.0000	0.4000	0.0352	0.0699	0.0037	0.4598	86.6432	0.0000
13.9535	0.0588	188.6048	0.0089	5.9333	0.4667	0.0355	0.0576	0.0028	0.9195	86.1160	0.0000
14.1030	0.0599	188.7670	0.0098	6.7333	0.3667	0.0362	0.0699	0.0036	0.9195	87.5220	0.0000
14.2525	0.0601	189.5707	0.0103	4.8667	0.3667	0.0364	0.0644	0.0043	1.8391	85.9402	0.0000
14.4020	0.0610	188.0274	0.0098	5.1333	0.4333	0.0373	0.0699	0.0032	1.1494	86.8190	0.0000
14.5515	0.0620	188.3113	0.0095	5.3333	0.3667	0.0382	0.0606	0.0034	1.1494	86.1160	0.0000
14.7010	0.0619	187.1814	0.0097	6.7333	0.4000	0.0380	0.0545	0.0039	0.6897	85.7645	0.0000
14.8505	0.0620	187.8274	0.0102	4.4667	0.4333	0.0373	0.0644	0.0041	2.2989	85.5888	0.0000
15.0000	0.0635	186.8685	0.0098	5.7333	0.3667	0.0390	0.0606	0.0045	0.6897	86.6432	0.0000

Appendix D: the Random Projection data modification: Arp on WDBC (569×30).

Arp	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0100	0.9963	187.6294	0.0036	8.8667	0.0000	0.9458	0.1176	1.0000	0.2299	85.4130	94.2003
0.1109	0.9721	187.8826	0.0076	10.2000	0.0000	0.4036	0.1714	1.0186	0.0000	85.2373	95.0791
0.2118	1.5624	189.5604	0.0023	10.7333	0.0333	0.0796	0.0910	1.5318	0.2299	85.0615	0.0000
0.3127	2.0098	189.7584	0.0039	9.8667	0.0000	0.6252	0.1225	2.7850	0.2299	85.4130	0.0000
0.4136	2.9599	190.6371	0.0032	10.6667	0.0000	1.5249	0.0606	6.7040	0.2299	84.3585	94.2003
0.5145	3.4469	190.5158	0.0031	9.8000	0.0000	2.0339	0.0798	9.2151	0.2299	85.4130	0.0000
0.6155	4.1797	190.2887	0.0029	9.6000	0.0333	2.9394	0.0681	15.1901	0.4598	85.0615	0.0000
0.7164	4.4662	188.4155	0.0050	11.2000	0.0000	3.3307	0.1473	19.3068	0.4598	84.5343	0.0000
0.8173	4.7312	187.8888	0.0062	9.0000	0.0333	3.8021	0.0495	21.8154	0.4598	84.3585	0.0000
0.9182	4.4479	188.9959	0.0029	10.4667	0.0667	3.5071	0.0396	18.5340	0.4598	85.2373	94.0246
1.0191	4.4349	191.2043	0.0032	9.8000	0.0000	3.2958	0.0464	17.4507	0.0000	85.2373	0.0000
1.1200	5.6153	188.4621	0.0067	10.1333	0.0333	4.6682	0.0662	30.9928	0.2299	84.0070	0.0000
1.2209	6.3646	188.4514	0.0050	11.8667	0.0000	4.9376	0.1510	36.7378	0.2299	84.3585	0.0000
1.3218	7.7595	189.3078	0.0028	9.2000	0.0667	6.6529	0.1838	57.7161	0.0000	85.0615	0.0000
1.4227	5.5762	189.5121	0.0036	9.8000	0.0000	4.5186	0.0489	30.7221	0.2299	83.8313	93.6731
1.5236	9.1943	189.0623	0.0053	9.5333	0.0667	7.8590	0.1628	80.2890	0.2299	85.0615	0.0000
1.6245	10.2188	190.3129	0.0029	11.4667	0.0000	8.9104	0.2444	99.5053	0.0000	85.2373	0.0000
1.7255	7.3354	188.3338	0.0043	9.1333	0.0333	6.2098	0.0446	50.1463	0.9195	85.4130	0.0000
1.8264	8.5112	190.4949	0.0073	10.8667	0.0667	7.7929	0.1033	74.1914	0.0000	84.7100	0.0000
1.9273	12.5101	189.6566	0.0056	10.6000	0.0333	11.2453	0.1157	153.1007	0.0000	85.4130	94.0246
2.0282	10.3391	190.3475	0.0050	10.3333	0.0000	8.7943	0.0891	100.5912	0.4598	84.5343	0.0000
2.1291	11.6758	190.5493	0.0030	9.4000	0.0000	10.7592	0.2438	136.1119	0.0000	85.4130	0.0000
2.2300	12.3841	189.4865	0.0055	7.4000	0.0000	11.1830	0.0811	152.1637	0.2299	85.2373	0.0000
2.3309	13.0726	189.1013	0.0038	11.4667	0.0333	12.0904	0.1040	171.7942	0.0000	85.2373	0.0000
2.4318	11.2142	189.2438	0.0044	8.2667	0.0667	10.6504	0.0557	128.7451	0.2299	84.3585	94.0246
2.5327	12.4314	190.3950	0.0032	10.5333	0.0000	11.2975	0.1126	150.4965	0.2299	85.4130	0.0000
2.6336	13.3946	188.1910	0.0043	9.4667	0.0000	12.6341	0.2389	181.2714	0.0000	85.4130	0.0000
2.7345	15.7944	188.7656	0.0056	9.8667	0.0667	14.6447	0.2382	245.7511	0.6897	84.3585	0.0000
2.8355	14.7120	188.4964	0.0073	7.4000	0.1333	13.6134	0.1541	213.4117	0.0000	85.2373	0.0000
2.9364	10.7754	189.4892	0.0026	9.6667	0.0000	9.5638	0.0402	107.6143	0.0000	85.0615	94.5518

Continued on next page

Appendix D – continued from previous page

Arp	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
3.0373	21.2164	189.7763	0.0032	9.9333	0.0333	19.5816	0.1015	434.7173	0.2299	85.0615	0.0000
3.1382	18.4825	190.8956	0.0043	9.0000	0.0667	17.3734	0.2426	333.8700	0.6897	85.2373	0.0000
3.2391	21.6700	189.7332	0.0049	8.9333	0.0000	20.0263	0.0675	468.0279	0.0000	85.2373	0.0000
3.3400	19.2596	188.0771	0.0076	9.8667	0.0667	18.5881	0.0613	378.2357	0.2299	84.3585	0.0000
3.4409	17.4231	187.8976	0.0040	11.5333	0.0000	16.2602	0.0947	301.5109	0.6897	84.8858	94.0246
3.5418	15.0411	188.7477	0.0094	11.4000	0.0000	14.1689	0.1262	234.8956	0.0000	84.3585	0.0000
3.6427	23.7100	189.3564	0.0045	9.2667	0.0000	21.9010	0.0576	555.6873	0.4598	83.8313	0.0000
3.7436	19.0109	190.1172	0.0046	9.8000	0.0000	17.3713	0.0767	364.3127	0.4598	83.8313	0.0000
3.8445	21.4570	188.1557	0.0037	12.1333	0.0000	20.2747	0.1330	455.5118	0.2299	85.2373	0.0000
3.9455	20.7650	190.4068	0.0063	10.2667	0.0667	19.4324	0.0681	429.0740	0.0000	85.5888	94.3761
4.0464	19.9458	188.7645	0.0037	10.6000	0.0000	18.6196	0.0501	401.8321	0.2299	85.2373	0.0000
4.1473	20.3621	189.5183	0.0046	8.6667	0.0000	19.2450	0.0823	417.2109	0.6897	85.4130	0.0000
4.2482	20.6949	189.7076	0.0035	9.4000	0.1000	20.0106	0.1009	420.8518	0.0000	85.2373	0.0000
4.3491	25.8885	188.0791	0.0066	10.9333	0.0333	25.2236	0.0984	678.0326	0.0000	85.2373	0.0000
4.4500	21.9717	187.8601	0.0089	9.4667	0.0333	21.2482	0.1423	486.3347	0.0000	85.2373	95.0791
4.5509	28.0028	188.1049	0.0057	8.6667	0.1000	27.3464	0.2797	784.4789	0.2299	84.5343	0.0000
4.6518	26.8259	189.0767	0.0019	10.1333	0.0333	25.5281	0.1869	712.3529	0.0000	85.0615	0.0000
4.7527	24.5680	190.2910	0.0032	10.7333	0.0333	23.1878	0.0483	600.7528	0.4598	85.2373	0.0000
4.8536	30.7305	187.1511	0.0070	8.2000	0.0667	30.0304	0.0879	943.0497	0.0000	85.4130	0.0000
4.9545	29.4734	188.5535	0.0067	10.6667	0.0333	28.6310	0.2438	872.5026	0.0000	85.2373	93.4974
5.0555	29.4711	188.1716	0.0070	10.5333	0.0667	28.1340	0.1139	866.0572	0.0000	85.2373	94.2003
5.1564	30.4940	188.6204	0.0089	9.2667	0.0333	28.7153	0.0668	930.7394	0.0000	84.3585	95.0791
5.2573	26.0683	188.8682	0.0044	8.6667	0.0333	25.1123	0.2184	684.5486	0.9195	85.2373	0.0000
5.3582	20.4977	189.0175	0.0021	9.8667	0.0333	19.7759	0.1139	402.7490	0.0000	85.4130	0.0000
5.4591	32.1581	189.0941	0.0030	9.5333	0.1000	31.3871	0.3886	1031.5563	0.4598	84.8858	94.2003
5.5600	39.4263	188.7360	0.0046	9.2667	0.0667	38.1024	0.2710	1546.6324	0.4598	85.0615	0.0000
5.6609	33.4269	187.4998	0.0050	8.4667	0.0333	33.5779	0.0842	1119.9676	0.0000	85.4130	0.0000
5.7618	27.2938	188.7818	0.0023	10.6667	0.0333	26.1584	0.3744	730.3795	0.2299	85.4130	0.0000
5.8627	30.9421	188.5900	0.0036	10.0000	0.0333	28.9990	0.0767	942.3693	0.0000	84.8858	0.0000

Continued on next page

Appendix D – continued from previous page

Arp	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
5.9636	32.4484	190.2076	0.0035	11.2000	0.0000	31.7051	0.2556	1043.5807	0.0000	85.2373	94.0246
6.0645	26.2262	188.5666	0.0056	8.4000	0.1667	24.5119	0.0563	692.5932	0.0000	85.0615	0.0000
6.1655	26.3337	189.2738	0.0040	9.6000	0.0667	25.3399	0.0563	669.0307	0.0000	85.0615	0.0000
6.2664	32.8161	189.7609	0.0049	8.2000	0.1000	32.0890	0.1250	1074.5619	0.2299	85.0615	0.0000
6.3673	30.1044	190.6450	0.0021	8.7333	0.1000	27.6784	0.0377	899.0772	0.0000	85.2373	0.0000
6.4682	28.7324	189.7671	0.0034	8.4667	0.0000	27.3559	0.1194	818.6893	0.2299	84.0070	93.6731
6.5691	37.2357	188.0473	0.0041	11.0000	0.1000	35.7933	0.4437	1373.4500	0.0000	85.4130	0.0000
6.6700	41.1491	186.8243	0.0093	10.4000	0.0000	39.9816	0.1479	1708.0442	0.2299	85.2373	0.0000
6.7709	45.5862	189.7545	0.0056	10.8667	0.0667	45.1261	0.0897	2050.4625	0.4598	84.1828	0.0000
6.8718	29.5778	188.3932	0.0067	10.2667	0.0333	28.8099	0.0749	875.6351	1.1494	83.8313	0.0000
6.9727	39.0711	190.7528	0.0077	11.4667	0.0000	39.0443	0.1108	1544.1632	0.0000	85.4130	94.0246
7.0736	35.3240	187.9869	0.0041	11.4667	0.0000	35.1073	0.0576	1261.5395	0.0000	85.4130	0.0000
7.1745	40.1428	187.6029	0.0059	11.0667	0.0333	38.4880	0.0798	1618.9498	0.0000	85.2373	0.0000
7.2755	34.7268	188.6095	0.0060	10.6667	0.0000	33.9700	0.1262	1219.0095	0.2299	83.6555	0.0000
7.3764	47.7348	187.9282	0.0095	9.2667	0.0000	46.3591	0.0792	2291.3266	0.2299	85.4130	0.0000
7.4773	36.9803	187.6903	0.0042	8.7333	0.0000	36.1928	0.2358	1352.7409	0.2299	85.2373	94.0246
7.5782	42.9095	188.8626	0.0057	9.8667	0.0000	43.9501	0.0501	1842.3831	0.4598	85.4130	0.0000
7.6791	39.1150	186.0756	0.0046	11.1333	0.0667	36.2830	0.0520	1511.4261	0.2299	84.1828	0.0000
7.7800	44.1079	186.1893	0.0142	9.8000	0.0333	42.9924	0.1102	1968.5909	0.2299	85.2373	0.0000
7.8809	49.4144	191.1466	0.0037	7.0000	0.0667	48.1778	0.0804	2423.8709	0.4598	83.8313	0.0000
7.9818	47.6441	188.7147	0.0045	11.1333	0.0000	47.1122	0.0384	2273.1778	0.2299	85.2373	94.5518
8.0827	51.4000	187.1501	0.0086	9.6000	0.0333	50.3521	0.1238	2656.1747	0.2299	85.0615	0.0000
8.1836	43.5667	190.4537	0.0028	7.8000	0.0667	42.5235	0.1015	1888.8443	0.4598	85.0615	0.0000
8.2845	40.6959	188.9680	0.0091	11.2000	0.0667	40.8026	0.0761	1688.2990	0.0000	85.2373	0.0000
8.3855	49.1742	188.1029	0.0051	9.4000	0.0333	47.9858	0.0730	2404.7391	0.2299	85.4130	0.0000
8.4864	41.3593	188.8484	0.0028	8.8000	0.0667	39.6494	0.1572	1679.5623	0.0000	85.4130	94.0246
8.5873	45.8874	188.9856	0.0067	10.2667	0.0333	44.5039	0.1479	2145.7804	0.0000	84.3585	0.0000
8.6882	47.1059	191.1631	0.0017	9.7333	0.0000	45.9880	0.1553	2202.0194	0.0000	84.5343	0.0000
8.7891	34.9771	192.0012	0.0024	10.4667	0.0000	35.1872	0.0316	1202.2527	0.0000	85.2373	0.0000

Continued on next page

Appendix D – continued from previous page

Arp	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
8.8900	48.8253	190.5514	0.0045	10.7333	0.0667	46.8669	0.0947	2361.2249	0.2299	84.0070	0.0000
8.9909	57.2145	188.0180	0.0040	11.8667	0.0000	55.8795	0.1003	3286.7968	0.2299	85.2373	94.3761
9.0918	54.4009	188.3023	0.0045	9.0000	0.0000	52.5650	0.2036	2954.3199	0.0000	85.2373	0.0000
9.1927	57.1497	189.7723	0.0040	9.8667	0.0000	54.3799	0.0458	3259.7256	0.0000	85.2373	0.0000
9.2936	67.9944	189.3924	0.0049	8.6000	0.0000	67.6101	0.1355	4622.5949	0.2299	85.0615	0.0000
9.3945	51.0415	189.4368	0.0077	8.7333	0.0333	49.1700	0.0724	2643.1388	0.4598	84.7100	0.0000
9.4955	36.5536	188.4826	0.0054	8.2000	0.1000	34.8937	0.0675	1331.9581	0.0000	85.4130	95.0791
9.5964	46.3489	190.4333	0.0046	10.4667	0.0000	45.3427	0.1566	2152.0290	0.2299	85.2373	0.0000
9.6973	50.9302	188.8786	0.0036	10.9333	0.0000	50.0183	0.1559	2607.2154	0.4598	85.2373	0.0000
9.7982	73.6637	190.0790	0.0067	10.0667	0.0333	72.5247	0.1374	5432.0859	0.0000	85.0615	0.0000
9.8991	56.9086	189.9880	0.0067	9.9333	0.0000	54.9873	0.0619	3228.4743	0.2299	84.8858	0.0000
10.0000	53.5993	188.5561	0.0037	10.7333	0.0000	51.8061	0.1776	2853.9325	0.6897	84.8858	93.4974

Appendix E: the Random Projection data modification *Arpo* on WDBC (569×30).

Arpo	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0100	1.1818	189.2079	0.0042	9.3333	0.0000	0.0000	100.0000	1.3460	0.4598	85.2373	93.8489
0.1109	1.5003	188.3823	0.0040	10.0667	0.0333	0.0000	100.0000	1.4024	0.0000	85.0615	94.2003
0.2118	1.3894	190.5769	0.0022	11.0667	0.0000	0.0000	100.0000	1.4133	0.4598	85.0615	0.0000
0.3127	1.5206	189.7984	0.0060	11.1333	0.0000	0.0000	100.0000	1.3965	0.0000	84.0070	0.0000
0.4136	1.3868	189.6685	0.0051	9.5333	0.0000	0.0000	100.0000	1.4129	0.0000	85.4130	94.0246
0.5145	1.5006	188.3087	0.0030	9.2000	0.1333	0.0000	100.0000	1.4027	0.2299	85.2373	0.0000
0.6155	1.4242	188.1309	0.0074	8.2000	0.0667	0.0000	100.0000	1.4130	0.2299	85.2373	0.0000
0.7164	1.5909	188.2052	0.0043	8.9333	0.0333	0.0000	100.0000	1.3624	0.4598	85.4130	0.0000
0.8173	1.4457	187.7379	0.0066	10.0667	0.0333	0.0000	100.0000	1.4123	0.0000	85.4130	0.0000
0.9182	1.4447	189.3024	0.0057	10.1333	0.0000	0.0000	100.0000	1.4126	0.2299	85.0615	93.4974
1.0191	1.3318	188.1203	0.0046	9.0000	0.0000	0.0000	100.0000	1.4049	0.4598	85.2373	0.0000
1.1200	1.3964	189.0394	0.0058	10.2667	0.0000	0.0000	100.0000	1.4131	0.0000	84.3585	0.0000
1.2209	1.5695	189.4259	0.0026	9.3333	0.0333	0.0000	100.0000	1.3751	0.0000	84.8858	0.0000
1.3218	1.5041	190.0422	0.0031	10.0000	0.0000	0.0000	100.0000	1.4012	0.0000	85.2373	0.0000
1.4227	1.5703	189.9870	0.0028	12.2000	0.0000	0.0000	100.0000	1.3742	0.0000	85.0615	94.2003
1.5236	1.2625	185.6964	0.0076	8.2000	0.0333	0.0000	100.0000	1.3835	0.0000	84.3585	0.0000
1.6245	1.5030	188.6909	0.0057	8.6000	0.0000	0.0000	100.0000	1.4020	0.2299	85.2373	0.0000
1.7255	1.4413	188.3242	0.0062	9.4667	0.0000	0.0000	100.0000	1.4127	0.4598	85.4130	0.0000
1.8264	1.3490	187.6520	0.0073	9.2667	0.0000	0.0000	100.0000	1.4083	0.0000	85.4130	0.0000
1.9273	1.2824	189.5782	0.0040	11.0000	0.0000	0.0000	100.0000	1.3912	0.0000	85.4130	94.0246
2.0282	1.4145	189.6340	0.0053	8.4000	0.0333	0.0000	100.0000	1.4141	0.0000	85.2373	0.0000
2.1291	1.6278	189.8409	0.0035	9.3333	0.0333	0.0000	100.0000	1.3363	0.2299	85.4130	0.0000
2.2300	1.4818	189.1290	0.0045	11.4000	0.0000	0.0000	100.0000	1.4059	0.4598	85.2373	0.0000
2.3309	1.4348	190.2799	0.0030	9.0000	0.0333	0.0000	100.0000	1.4127	0.0000	85.0615	0.0000
2.4318	1.5279	189.7745	0.0032	9.8667	0.0667	0.0000	100.0000	1.3937	0.0000	85.2373	94.3761
2.5327	1.2710	188.4975	0.0060	9.8000	0.0333	0.0000	100.0000	1.3864	0.2299	85.2373	0.0000
2.6336	1.1509	187.8751	0.0080	8.0667	0.0667	0.0000	100.0000	1.3290	0.0000	85.0615	0.0000
2.7345	1.4121	188.3470	0.0040	8.6667	0.1333	0.0000	100.0000	1.4139	0.2299	84.8858	0.0000
2.8355	1.1991	188.1834	0.0096	10.6667	0.1000	0.0000	100.0000	1.3558	0.6897	84.5343	0.0000
2.9364	1.4802	188.1802	0.0028	8.7333	0.0000	0.0000	100.0000	1.4065	0.2299	85.4130	94.9033

Continued on next page

$Arpo$	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0,\sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
3.0373	1.2976	189.5162	0.0029	10.1333	0.0000	0.0000	100.0000	1.3956	0.0000	83.8313	0.0000
3.1382	1.3235	189.6490	0.0056	11.5333	0.0667	0.0000	100.0000	1.4019	0.2299	85.4130	0.0000
3.2391	1.2555	189.7250	0.0093	11.8000	0.0333	0.0000	100.0000	1.3815	0.0000	83.8313	0.0000
3.3400	1.4331	188.5671	0.0071	10.3333	0.0000	0.0000	100.0000	1.4129	0.2299	85.2373	0.0000
3.4409	1.5858	188.5052	0.0056	8.9333	0.0000	0.0000	100.0000	1.3660	0.2299	85.4130	94.7276
3.5418	1.3883	188.2498	0.0054	10.0000	0.0667	0.0000	100.0000	1.4132	0.4598	85.0615	0.0000
3.6427	1.3707	190.5195	0.0033	11.0000	0.0667	0.0000	100.0000	1.4113	0.4598	84.5343	0.0000
3.7436	1.3457	190.4528	0.0029	9.6667	0.0667	0.0000	100.0000	1.4071	0.2299	85.4130	0.0000
3.8445	1.6994	189.9640	0.0052	8.7333	0.0000	0.0000	100.0000	1.2640	0.2299	84.3585	0.0000
3.9455	1.4658	190.0320	0.0029	9.0000	0.0333	0.0000	100.0000	1.4097	0.2299	85.2373	93.6731
4.0464	1.3482	187.9381	0.0049	10.7333	0.0000	0.0000	100.0000	1.4084	0.2299	84.5343	0.0000
4.1473	1.3803	189.5066	0.0050	9.8000	0.0333	0.0000	100.0000	1.4117	0.2299	84.3585	0.0000
4.2482	1.3109	188.0351	0.0063	9.3333	0.0667	0.0000	100.0000	1.4002	0.2299	85.2373	0.0000
4.3491	1.2437	191.4764	0.0057	10.1333	0.0000	0.0000	100.0000	1.3765	0.0000	85.4130	0.0000
4.4500	1.4280	189.3373	0.0050	11.6000	0.0000	0.0000	100.0000	1.4138	0.0000	84.0070	93.8489
4.5509	1.3423	187.7980	0.0062	10.2000	0.0333	0.0000	100.0000	1.4071	0.0000	85.4130	0.0000
4.6518	1.3265	189.5126	0.0057	10.8667	0.0333	0.0000	100.0000	1.4032	0.4598	85.2373	0.0000
4.7527	1.6202	189.3827	0.0029	10.6000	0.1000	0.0000	100.0000	1.3410	0.4598	85.2373	0.0000
4.8536	1.0845	187.6170	0.0063	10.9333	0.0000	0.0000	100.0000	1.2863	0.2299	85.2373	0.0000
4.9545	1.3953	189.6287	0.0033	9.4000	0.0667	0.0000	100.0000	1.4134	0.9195	85.0615	94.7276
5.0555	1.5863	188.1352	0.0035	10.2667	0.0333	0.0000	100.0000	1.3647	0.2299	85.4130	0.0000
5.1564	1.6047	190.7523	0.0025	8.6667	0.0667	0.0000	100.0000	1.3546	0.2299	84.3585	0.0000
5.2573	1.3585	188.1660	0.0069	8.3333	0.0333	0.0000	100.0000	1.4099	0.0000	85.4130	0.0000
5.3582	1.4425	190.1701	0.0040	8.4000	0.0000	0.0000	100.0000	1.4125	0.2299	83.8313	0.0000
5.4591	1.4764	188.8550	0.0042	9.2667	0.1000	0.0000	100.0000	1.4083	0.0000	85.2373	0.0000
5.5600	1.1957	188.0411	0.0057	9.1333	0.0667	0.0000	100.0000	1.3542	0.0000	85.2373	0.0000
5.6609	1.4697	189.5347	0.0044	11.0000	0.0333	0.0000	100.0000	1.4092	0.4598	84.5343	0.0000
5.7618	1.2695	187.5794	0.0070	8.6667	0.0667	0.0000	100.0000	1.3863	0.2299	84.3585	0.0000
5.8627	1.0727	188.3002	0.0060	9.6667	0.0667	0.0000	100.0000	1.2769	0.2299	84.3585	0.0000

Continued on next page

Appendix E – continued from previous page

Arpo	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0,\sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
5.9636	1.4734	189.4912	0.0036	10.1333	0.0333	0.0000	100.0000	1.4086	0.0000	84.7100	0.0000
6.0645	1.3161	188.6011	0.0079	7.9333	0.0000	0.0000	100.0000	1.4009	0.0000	85.2373	0.0000
6.1655	1.6031	189.1077	0.0042	9.4000	0.1000	0.0000	100.0000	1.3550	0.2299	85.4130	0.0000
6.2664	1.6149	190.2901	0.0040	10.5333	0.0333	0.0000	100.0000	1.3458	0.2299	85.2373	0.0000
6.3673	1.4018	186.8280	0.0076	7.6000	0.1333	0.0000	100.0000	1.4138	0.2299	85.4130	0.0000
6.4682	1.4019	189.0532	0.0037	10.4667	0.0333	0.0000	100.0000	1.4138	0.0000	85.2373	0.0000
6.5691	1.4501	186.8395	0.0066	11.1333	0.0000	0.0000	100.0000	1.4119	0.0000	85.0615	0.0000
6.6700	1.6494	190.5707	0.0046	9.0667	0.0333	0.0000	100.0000	1.3164	0.2299	85.4130	0.0000
6.7709	1.4227	189.7311	0.0024	10.0667	0.0333	0.0000	100.0000	1.4134	0.0000	85.4130	0.0000
6.8718	1.5332	189.6586	0.0033	12.3333	0.0333	0.0000	100.0000	1.3916	0.2299	85.4130	0.0000
6.9727	1.1127	189.4738	0.0080	8.8000	0.1000	0.0000	100.0000	1.3045	0.0000	85.2373	0.0000
7.0736	1.3027	187.9888	0.0096	11.5333	0.0333	0.0000	100.0000	1.3979	0.2299	85.2373	0.0000
7.1745	1.5690	189.0069	0.0030	9.2667	0.0333	0.0000	100.0000	1.3753	0.2299	85.4130	0.0000
7.2755	1.2007	188.4531	0.0043	7.6667	0.0667	0.0000	100.0000	1.3568	0.0000	85.0615	0.0000
7.3764	1.2603	189.3822	0.0063	10.5333	0.0333	0.0000	100.0000	1.3833	0.4598	83.8313	0.0000
7.4773	1.2069	188.3995	0.0067	8.5333	0.0333	0.0000	100.0000	1.3589	0.0000	85.2373	0.0000
7.5782	1.3821	188.8814	0.0055	8.5333	0.0333	0.0000	100.0000	1.4122	0.4598	85.4130	0.0000
7.6791	1.3838	189.7021	0.0038	9.3333	0.0667	0.0000	100.0000	1.4128	0.4598	85.2373	0.0000
7.7800	1.3955	187.9931	0.0080	11.4000	0.0333	0.0000	100.0000	1.4134	0.4598	85.4130	0.0000
7.8809	1.2299	188.5427	0.0069	12.6667	0.0000	0.0000	100.0000	1.3708	0.2299	85.2373	0.0000
7.9818	1.0878	189.2455	0.0062	8.8667	0.0333	0.0000	100.0000	1.2872	0.2299	85.4130	0.0000
8.0827	1.4321	187.4443	0.0065	10.4000	0.0333	0.0000	100.0000	1.4131	0.2299	85.2373	0.0000
8.1836	1.4061	190.1072	0.0049	9.7333	0.0667	0.0000	100.0000	1.4140	0.0000	85.2373	0.0000
8.2845	1.2145	189.0075	0.0046	9.2667	0.0000	0.0000	100.0000	1.3629	0.0000	85.2373	0.0000
8.3855	1.4467	190.1575	0.0032	10.7333	0.0000	0.0000	100.0000	1.4121	0.9195	85.2373	0.0000
8.4864	1.2286	189.6404	0.0061	10.0000	0.0333	0.0000	100.0000	1.3700	0.2299	85.4130	0.0000
8.5873	1.3415	189.2221	0.0034	9.8667	0.0000	0.0000	100.0000	1.4069	0.0000	85.4130	0.0000
8.6882	1.4510	188.1392	0.0064	9.8000	0.0333	0.0000	100.0000	1.4117	0.0000	85.0615	0.0000
8.7891	1.2051	188.2831	0.0060	10.4667	0.0667	0.0000	100.0000	1.3592	0.4598	83.8313	0.0000

Continued on next page

Appendix E **– continued from previous page**

Arpo	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
8.8900	1.5234	189.6075	0.0040	9.8000	0.0333	0.0000	100.0000	1.3956	0.0000	85.2373	0.0000
8.9909	1.4847	188.8963	0.0048	10.0667	0.1000	0.0000	100.0000	1.4059	0.0000	85.2373	0.0000
9.0918	1.2900	190.1018	0.0036	11.4667	0.0000	0.0000	100.0000	1.3935	0.2299	85.2373	0.0000
9.1927	1.1593	188.6469	0.0086	10.3333	0.0333	0.0000	100.0000	1.3347	0.0000	85.2373	0.0000
9.2936	1.6666	190.0232	0.0056	8.6667	0.0000	0.0000	100.0000	1.3006	0.0000	85.2373	0.0000
9.3945	1.2674	189.4292	0.0070	9.8667	0.0333	0.0000	100.0000	1.3864	0.0000	85.0615	0.0000
9.4955	1.4381	187.9024	0.0042	11.5333	0.0000	0.0000	100.0000	1.4128	0.0000	85.4130	0.0000
9.5964	1.3661	188.6621	0.0026	9.6667	0.0333	0.0000	100.0000	1.4108	0.2299	85.2373	0.0000
9.6973	1.3007	186.9893	0.0088	10.9333	0.0333	0.0000	100.0000	1.3970	0.0000	84.8858	0.0000
9.7982	1.3293	190.1673	0.0040	10.0667	0.0000	0.0000	100.0000	1.4038	0.4598	85.0615	0.0000
9.8991	1.3774	189.7847	0.0032	10.8667	0.0000	0.0000	100.0000	1.4124	0.2299	85.2373	0.0000
10.0000	1.3125	188.7087	0.0059	8.6000	0.0000	0.0000	100.0000	1.4003	0.0000	85.2373	0.0000

Appendix F: the Random Projection data modification: rpA on WDBC (569×30).

rpA	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0,\sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0100	1.0255	188.9100	0.0019	10.0000	0.0000	0.8226	0.0012	0.9442	67.3563	50.9666	51.1424
0.1109	2.9476	190.1832	0.0018	2.0000	0.6333	4.8264	0.0000	6.7497	42.5287	52.8998	56.0633
0.2118	5.0594	190.8349	0.0018	7.6667	0.0000	8.8939	0.0006	23.0469	53.1034	49.3849	0.0000
0.3127	7.3963	190.4995	0.0017	10.1333	0.0000	13.5138	0.0006	52.1890	67.5862	50.4394	0.0000
0.4136	10.0277	189.5033	0.0016	6.9333	0.4667	18.6969	0.0000	99.0988	70.3448	51.4938	51.4938
0.5145	12.8864	188.6976	0.0019	9.1333	0.1667	24.3457	0.0000	166.0336	58.6207	51.6696	0.0000
0.6155	14.7647	189.9690	0.0021	8.6000	0.1667	27.8577	0.0000	216.5006	51.7241	50.6151	0.0000
0.7164	17.7747	190.6508	0.0019	9.6000	0.1000	33.6931	0.0006	314.6888	45.0575	46.7487	0.0000
0.8173	19.9307	190.3659	0.0021	7.6667	0.3333	37.8446	0.0006	395.6322	72.6437	50.4394	0.0000
0.9182	21.0201	189.5238	0.0011	6.4000	0.2333	39.9285	0.0006	439.9028	62.2989	49.5606	53.7786
1.0191	25.1639	191.2163	0.0012	9.8667	0.0000	47.8453	0.0000	629.7400	49.1954	51.6696	0.0000
1.1200	27.4374	190.1958	0.0022	9.2667	0.0667	52.3975	0.0000	753.4051	54.9425	46.5729	0.0000
1.2209	29.6143	190.7394	0.0014	10.5333	0.0333	56.4877	0.0019	875.3902	50.5747	51.1424	0.0000
1.3218	31.9740	188.9631	0.0011	9.8667	0.0333	61.1519	0.0006	1024.4111	57.2414	50.0879	0.0000
1.4227	34.3562	189.0739	0.0015	9.4667	0.0333	65.5806	0.0006	1176.0270	66.8966	49.5606	0.0000
1.5236	34.6933	190.5951	0.0018	9.0000	0.0000	66.3695	0.0012	1202.8601	59.3103	50.7909	0.0000
1.6245	37.4507	188.6876	0.0014	8.2667	0.1000	71.6809	0.0012	1403.1668	31.0345	49.3849	0.0000
1.7255	43.1507	190.6088	0.0020	3.6667	0.4000	82.5415	0.0012	1861.1719	62.0690	50.0879	0.0000
1.8264	41.8727	190.9632	0.0016	1.7333	0.7000	79.9037	0.0006	1757.7448	52.4138	49.0334	0.0000
1.9273	46.9986	190.1433	0.0015	11.4000	0.0000	89.7725	0.0000	2202.0068	71.7241	49.0334	52.3726
2.0282	47.7109	189.3139	0.0023	8.4000	0.0667	91.4657	0.0006	2275.7783	52.4138	52.5483	0.0000
2.1291	52.8871	189.5644	0.0021	1.6667	0.7000	100.7123	0.0000	2807.9961	45.7471	52.5483	0.0000
2.2300	54.1867	189.8517	0.0026	10.2667	0.0000	103.8695	0.0000	2935.0204	55.8621	46.7487	0.0000
2.3309	53.0153	190.5104	0.0021	10.2000	0.0333	101.6023	0.0006	2800.5273	53.1034	47.6274	0.0000
2.4318	57.3772	188.9261	0.0019	8.2000	0.0667	110.1020	0.0000	3289.6072	48.9655	42.7065	50.7909
2.5327	59.4014	189.7954	0.0015	10.4667	0.0000	113.3297	0.0006	3516.5401	60.4598	48.6819	0.0000
2.6336	62.0352	190.2550	0.0022	9.2667	0.1333	119.0595	0.0012	3848.4286	67.1264	47.8032	0.0000
2.7345	64.4819	187.8635	0.0018	4.4000	0.6333	123.7581	0.0012	4155.7857	57.0115	49.9121	0.0000
2.8355	67.5673	191.5809	0.0018	10.8000	0.0333	129.6485	0.0019	4554.8165	74.2529	47.8032	0.0000
2.9364	68.6021	189.9011	0.0023	4.6667	0.5333	131.7064	0.0006	4706.6288	40.0000	53.0756	51.3181

Continued on next page

Appendix F – continued from previous page

rpA	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ $\sigma - r$	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
3.0373	70.3643	188.6351	0.0019	9.7333	0.0000	135.2067	0.0012	4952.8565	48.9655	50.7909	0.0000
3.1382	75.0239	189.8200	0.0016	5.0667	0.6000	144.0096	0.0012	5631.7270	49.6552	52.3726	0.0000
3.2391	75.7172	188.1319	0.0020	6.3333	0.2333	145.3184	0.0006	5736.2570	47.8161	50.2636	0.0000
3.3400	79.6232	190.8837	0.0013	10.0000	0.0000	152.9175	0.0012	6335.5615	48.9655	51.3181	0.0000
3.4409	88.0715	191.2837	0.0022	11.0667	0.0000	168.7328	0.0006	7752.9582	61.3793	47.4517	52.0211
3.5418	85.6351	188.9647	0.0015	8.6000	0.0667	164.4314	0.0000	7319.4022	62.9885	46.0457	0.0000
3.6427	88.9226	189.4787	0.0012	10.9333	0.0333	170.9348	0.0000	7907.4661	61.6092	52.7241	0.0000
3.7436	88.7213	188.0699	0.0013	11.0000	0.0333	170.4516	0.0006	7868.7021	53.7931	51.8453	0.0000
3.8445	91.8989	189.0302	0.0019	9.8667	0.0000	176.1837	0.0006	8426.7432	46.2069	50.2636	0.0000
3.9455	91.2852	188.0197	0.0020	10.6000	0.0333	175.3778	0.0006	8318.4743	47.3563	54.6573	51.6696
4.0464	93.0691	190.4704	0.0016	7.3333	0.0333	178.8824	0.0006	8645.8261	69.1954	52.0211	0.0000
4.1473	102.3242	188.7726	0.0016	9.4667	0.1000	196.8310	0.0012	10483.1366	58.8506	51.1424	0.0000
4.2482	93.5961	190.6418	0.0012	11.0667	0.0000	179.9015	0.0006	8742.9594	48.2759	52.1968	0.0000
4.3491	96.9375	190.5093	0.0018	4.4000	0.7333	186.1355	0.0000	9392.0031	47.8161	48.1547	0.0000
4.4500	102.9173	189.2175	0.0016	10.2667	0.0000	197.7163	0.0006	10577.5293	52.6437	47.4517	56.0633
4.5509	105.3957	189.7434	0.0018	10.1333	0.0333	202.6841	0.0006	11096.9765	56.5517	51.3181	0.0000
4.6518	113.6588	189.5269	0.0016	9.5333	0.0000	217.8457	0.0012	12902.1323	59.5402	49.3849	0.0000
4.7527	110.5891	188.1852	0.0018	5.4000	0.3667	212.6757	0.0000	12245.2930	52.6437	49.5606	0.0000
4.8536	112.8578	189.1156	0.0015	6.9333	0.3667	217.1102	0.0000	12739.9064	63.4483	47.9789	0.0000
4.9545	121.9331	190.4586	0.0012	4.7333	0.4667	234.6220	0.0000	14869.1685	54.7126	47.8032	52.1968
5.0555	117.6319	191.8598	0.0019	11.3333	0.0333	226.2366	0.0000	13823.9364	58.3908	49.9121	0.0000
5.1564	121.9034	189.9640	0.0015	8.7333	0.0000	233.5913	0.0019	14841.6768	69.4253	53.4271	0.0000
5.2573	131.9531	190.0191	0.0016	4.8667	0.4000	253.3441	0.0000	17432.3982	44.1379	51.8453	0.0000
5.3582	123.3097	189.6437	0.0021	9.5333	0.0000	237.0571	0.0006	15182.0862	64.3678	47.8032	0.0000
5.4591	134.4226	191.5946	0.0018	8.0667	0.0000	258.4806	0.0000	18071.5966	70.5747	47.1002	0.0000
5.5600	132.2477	190.1896	0.0014	8.5333	0.0000	254.0178	0.0012	17474.9184	70.3448	51.3181	0.0000
5.6609	139.1211	190.6916	0.0012	8.8000	0.0000	266.5252	0.0000	19340.4033	62.5287	50.4394	0.0000
5.7618	139.8503	192.2979	0.0012	9.2000	0.0000	268.5420	0.0000	19528.6404	43.9080	49.2091	0.0000
5.8627	145.7410	189.4123	0.0025	2.9333	0.7667	280.0223	0.0006	21274.1978	60.0000	44.4640	0.0000

Continued on next page

Appendix F – continued from previous page

rpA	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ $\sigma - r$	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
5.9636	149.0896	190.4005	0.0019	1.1333	0.6667	286.7357	0.0000	22255.4794	64.5977	50.6151	0.0000
6.0645	149.2367	190.1544	0.0013	9.6000	0.0000	286.9920	0.0000	22273.6541	47.3563	50.2636	0.0000
6.1655	148.6122	188.0326	0.0018	10.6000	0.0333	285.9571	0.0006	22066.1868	45.9770	52.8998	0.0000
6.2664	145.4679	187.8161	0.0016	11.6667	0.0000	279.8479	0.0006	21132.3027	29.8851	50.4394	0.0000
6.3673	149.5499	191.3322	0.0011	6.2000	0.4667	287.6098	0.0019	22365.8579	62.9885	49.0334	0.0000
6.4682	157.6656	189.9699	0.0015	3.5333	0.6000	303.3885	0.0000	24885.4048	60.9195	48.5062	0.0000
6.5691	158.8172	190.9383	0.0018	10.9333	0.1000	305.7483	0.0000	25238.2724	57.0115	50.6151	0.0000
6.6700	157.8186	190.3276	0.0016	3.4667	0.4667	303.3422	0.0000	24928.7716	49.6552	46.3972	0.0000
6.7709	160.6925	188.6915	0.0015	10.0000	0.0333	309.4056	0.0012	25824.5130	62.0690	52.0211	0.0000
6.8718	155.8792	190.4274	0.0017	4.9333	0.4667	300.1033	0.0000	24262.0413	47.8161	45.5185	0.0000
6.9727	174.8567	188.0346	0.0021	9.1333	0.0000	336.2836	0.0000	30586.0076	36.5517	52.3726	0.0000
7.0736	164.1636	190.4277	0.0013	5.8667	0.2333	315.8853	0.0006	26944.9059	51.9540	53.6028	0.0000
7.1745	174.1024	189.5293	0.0015	7.7333	0.0000	335.0017	0.0006	30304.1957	56.7816	49.0334	0.0000
7.2755	176.1722	190.6369	0.0018	6.1333	0.5667	339.0822	0.0012	31050.3987	59.7701	51.1424	0.0000
7.3764	174.7405	190.3087	0.0019	4.6000	0.3000	336.4436	0.0025	30537.8857	60.4598	51.1424	0.0000
7.4773	181.1392	188.5117	0.0019	5.4667	0.2667	348.6711	0.0012	32823.7395	46.6667	50.4394	0.0000
7.5782	184.2834	188.8057	0.0023	6.4667	0.1667	354.9192	0.0000	33983.8962	51.9540	53.6028	0.0000
7.6791	183.2310	187.9768	0.0017	4.8667	0.4333	352.4129	0.0000	33583.4306	56.3218	49.2091	0.0000
7.7800	181.9794	189.7129	0.0025	11.4000	0.0667	350.4624	0.0000	33101.3545	48.0460	49.9121	0.0000
7.8809	191.5518	189.1237	0.0016	7.8667	0.0000	368.7233	0.0006	36678.7361	65.7471	51.6696	0.0000
7.9818	187.5067	189.0153	0.0016	4.4667	0.7000	359.9022	0.0000	35185.7400	57.4713	49.7364	0.0000
8.0827	190.4633	189.3725	0.0018	3.0000	0.7000	365.2285	0.0000	36317.1065	51.7241	49.7364	0.0000
8.1836	199.3905	189.6983	0.0017	2.0667	0.7333	382.8672	0.0000	39804.7219	61.3793	49.9121	0.0000
8.2845	194.7559	189.2206	0.0015	2.1333	0.6667	374.0243	0.0000	37948.6437	54.7126	53.2513	0.0000
8.3855	202.9098	188.8839	0.0013	10.3333	0.0000	390.6239	0.0000	41173.0841	54.9425	46.3972	0.0000
8.4864	202.8055	190.2224	0.0021	1.8667	0.7000	389.3564	0.0012	41171.2176	67.8161	49.0334	0.0000
8.5873	197.0407	188.6185	0.0018	11.0667	0.0333	379.5046	0.0000	38812.1101	53.3333	53.9543	0.0000
8.6882	212.6210	189.6026	0.0024	7.4000	0.2333	409.3531	0.0000	45225.4534	68.7356	52.1968	0.0000
8.7891	209.4096	192.7673	0.0015	9.5333	0.0000	403.1187	0.0006	43846.7225	70.5747	48.8576	0.0000

Continued on next page

Appendix F **– continued from previous page**

rpA	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ $\sigma - r$	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
8.8900	216.0106	190.7873	0.0020	9.6667	0.0000	414.9974	0.0006	46608.8865	41.8391	49.3849	0.0000
8.9909	217.4669	189.8458	0.0021	3.2000	0.6000	418.2498	0.0012	47304.1880	50.5747	51.1424	0.0000
9.0918	222.5247	189.3220	0.0022	9.8000	0.0000	427.3837	0.0006	49515.6117	65.7471	49.3849	0.0000
9.1927	213.9600	189.0841	0.0019	9.9333	0.0000	411.9759	0.0025	45771.5931	62.2989	49.7364	0.0000
9.2936	223.1847	187.1333	0.0016	8.8000	0.0000	429.5726	0.0006	49817.3844	50.1149	49.9121	0.0000
9.3945	228.9925	189.7481	0.0013	5.8667	0.5667	441.0160	0.0012	52445.5506	59.7701	49.0334	0.0000
9.4955	228.3972	188.8244	0.0023	9.1333	0.0000	439.4519	0.0012	52124.3442	60.9195	47.8032	0.0000
9.5964	224.3466	188.4394	0.0017	11.2667	0.0667	431.9813	0.0006	50361.3566	57.4713	49.0334	0.0000
9.6973	229.1669	189.8374	0.0016	9.0000	0.0000	440.6262	0.0000	52472.0316	56.7816	55.0088	0.0000
9.7982	224.1132	188.7336	0.0022	8.8000	0.0000	431.3731	0.0012	50168.5773	44.3678	53.7786	0.0000
9.8991	254.9169	190.3637	0.0022	6.9333	0.3333	490.9384	0.0012	65010.2893	54.4828	46.9244	0.0000
10.0000	248.3585	190.2059	0.0018	3.3333	0.7000	477.2748	0.0000	61737.2010	52.1839	47.4517	0.0000

Appendix G: the Random Projection data modification: $rpoA$ on WDBC (569×30).

$rpoA$	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0100	1.4342	190.0144	0.0015	9.0000	0.0000	1.5873	0.0012	0.0000	100.0000	49.7364	52.5483
0.1109	1.4273	190.8490	0.0015	8.6000	0.0667	1.5901	0.0000	0.0000	100.0000	52.5483	56.0633
0.2118	1.3887	190.3886	0.0022	8.1333	0.1333	1.5488	0.0000	0.0000	100.0000	49.0334	0.0000
0.3127	1.3772	189.2055	0.0013	1.5333	0.9333	1.5488	0.0012	0.0000	100.0000	50.4394	0.0000
0.4136	1.4586	189.8460	0.0015	8.6667	0.0000	1.5590	0.0000	0.0000	100.0000	49.0334	56.7663
0.5145	1.4249	190.2913	0.0021	11.1333	0.0000	1.5843	0.0006	0.0000	100.0000	54.1301	0.0000
0.6155	1.3968	188.1011	0.0022	4.0667	0.4667	1.5730	0.0012	0.0000	100.0000	49.5606	0.0000
0.7164	1.4410	189.8856	0.0017	12.1333	0.0667	1.5797	0.0012	0.0000	100.0000	49.0334	0.0000
0.8173	1.4307	188.6425	0.0025	8.2667	0.0333	1.5632	0.0006	0.0000	100.0000	51.3181	0.0000
0.9182	1.4654	187.7311	0.0016	9.6000	0.0000	1.5644	0.0006	0.0000	100.0000	46.5729	51.4938
1.0191	1.4088	191.6266	0.0013	4.7333	0.4333	1.5683	0.0000	0.0000	100.0000	49.0334	0.0000
1.1200	1.4128	190.3219	0.0016	9.0667	0.0667	1.5734	0.0000	0.0000	100.0000	48.8576	0.0000
1.2209	1.4484	189.1163	0.0015	8.7333	0.0000	1.5878	0.0019	0.0000	100.0000	50.6151	0.0000
1.3218	1.4441	189.3459	0.0020	11.1333	0.0000	1.5780	0.0019	0.0000	100.0000	50.2636	0.0000
1.4227	1.3417	189.3051	0.0015	3.0667	0.7000	1.5605	0.0012	0.0000	100.0000	52.5483	53.9543
1.5236	1.3828	190.1667	0.0022	7.6667	0.0667	1.5976	0.0012	0.0000	100.0000	49.5606	0.0000
1.6245	1.4123	191.0161	0.0015	9.4667	0.0333	1.5883	0.0000	0.0000	100.0000	52.1968	0.0000
1.7255	1.4168	189.6301	0.0016	10.3333	0.0000	1.5738	0.0000	0.0000	100.0000	47.6274	0.0000
1.8264	1.4283	190.6296	0.0013	10.4667	0.0000	1.5626	0.0006	0.0000	100.0000	48.6819	0.0000
1.9273	1.4527	188.8448	0.0015	9.7333	0.0000	1.5814	0.0000	0.0000	100.0000	52.0211	52.0211
2.0282	1.4375	190.2916	0.0017	8.7333	0.0000	1.5851	0.0012	0.0000	100.0000	47.8032	0.0000
2.1291	1.4413	190.1441	0.0023	9.2000	0.0000	1.5569	0.0019	0.0000	100.0000	50.4394	0.0000
2.2300	1.4364	191.8605	0.0015	7.6000	0.0000	1.5863	0.0000	0.0000	100.0000	50.2636	0.0000
2.3309	1.3763	190.8077	0.0018	2.9333	0.4333	1.5653	0.0006	0.0000	100.0000	55.7118	0.0000
2.4318	1.3820	189.5135	0.0016	8.7333	0.2667	1.5593	0.0006	0.0000	100.0000	46.9244	53.9543
2.5327	1.3924	188.7565	0.0012	3.1333	0.5667	1.5727	0.0000	0.0000	100.0000	52.0211	0.0000
2.6336	1.4524	190.5376	0.0015	9.0000	0.1000	1.5699	0.0000	0.0000	100.0000	47.8032	0.0000
2.7345	1.3893	188.8344	0.0022	8.2000	0.1000	1.5692	0.0006	0.0000	100.0000	47.8032	0.0000
2.8355	1.4057	191.1570	0.0018	11.6667	0.0333	1.5755	0.0000	0.0000	100.0000	49.9121	0.0000
2.9364	1.4226	189.6773	0.0016	9.6000	0.0000	1.5604	0.0000	0.0000	100.0000	49.0334	55.7118

Continued on next page

Appendix G – continued from previous page

$rpoA$	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0,\sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
3.0373	1.4220	189.8400	0.0015	7.8000	0.0333	1.5564	0.0012	0.0000	100.0000	51.3181	0.0000
3.1382	1.4133	189.4395	0.0017	11.4667	0.0000	1.5730	0.0000	0.0000	100.0000	50.4394	0.0000
3.2391	1.3969	191.5469	0.0013	11.4667	0.0000	1.5639	0.0012	0.0000	100.0000	52.1968	0.0000
3.3400	1.4543	188.4251	0.0023	8.6667	0.0000	1.5739	0.0012	0.0000	100.0000	47.6274	0.0000
3.4409	1.4020	189.0166	0.0019	0.4000	0.7667	1.5649	0.0000	0.0000	100.0000	48.1547	49.7364
3.5418	1.4729	189.9322	0.0021	9.2667	0.0000	1.5351	0.0012	0.0000	100.0000	50.7909	0.0000
3.6427	1.4392	189.6257	0.0016	8.4000	0.0667	1.5672	0.0006	0.0000	100.0000	48.8576	0.0000
3.7436	1.4116	189.9481	0.0014	6.5333	0.5000	1.5890	0.0012	0.0000	100.0000	44.6397	0.0000
3.8445	1.4286	189.2043	0.0019	7.4667	0.1333	1.5701	0.0000	0.0000	100.0000	51.3181	0.0000
3.9455	1.4249	188.2894	0.0017	9.2000	0.0000	1.5725	0.0000	0.0000	100.0000	50.0879	54.1301
4.0464	1.4026	189.0557	0.0019	3.3333	0.7667	1.5600	0.0006	0.0000	100.0000	47.9789	0.0000
4.1473	1.4336	188.4402	0.0019	7.8000	0.0000	1.5890	0.0012	0.0000	100.0000	49.0334	0.0000
4.2482	1.4285	188.8888	0.0013	12.4667	0.0333	1.5800	0.0000	0.0000	100.0000	51.4938	0.0000
4.3491	1.3926	189.1338	0.0019	9.3333	0.1333	1.5790	0.0000	0.0000	100.0000	49.5606	0.0000
4.4500	1.3841	190.6913	0.0014	4.2667	0.5667	1.5600	0.0019	0.0000	100.0000	48.6819	53.6028
4.5509	1.4732	188.6205	0.0015	10.8667	0.0000	1.5791	0.0006	0.0000	100.0000	50.7909	0.0000
4.6518	1.4534	190.6310	0.0016	9.5333	0.0000	1.5718	0.0000	0.0000	100.0000	46.5729	0.0000
4.7527	1.4357	189.9379	0.0018	11.8667	0.0000	1.5898	0.0019	0.0000	100.0000	48.1547	0.0000
4.8536	1.3856	188.5508	0.0019	9.9333	0.1000	1.5781	0.0000	0.0000	100.0000	50.7909	0.0000
4.9545	1.4608	191.0050	0.0012	9.6000	0.0000	1.5548	0.0000	0.0000	100.0000	49.7364	55.7118
5.0555	1.4132	189.4011	0.0012	10.4667	0.1000	1.5721	0.0000	0.0000	100.0000	47.6274	0.0000
5.1564	1.3892	189.0158	0.0022	10.0000	0.0667	1.5681	0.0000	0.0000	100.0000	50.6151	0.0000
5.2573	1.4477	191.6609	0.0018	9.9333	0.0000	1.5766	0.0006	0.0000	100.0000	51.1424	0.0000
5.3582	1.3883	190.4867	0.0018	13.0000	0.0333	1.5682	0.0006	0.0000	100.0000	52.0211	0.0000
5.4591	1.4397	189.0674	0.0022	10.1333	0.0000	1.5630	0.0006	0.0000	100.0000	49.0334	0.0000
5.5600	1.4401	189.9445	0.0020	9.7333	0.0333	1.5827	0.0000	0.0000	100.0000	48.1547	0.0000
5.6609	1.3869	189.6544	0.0018	10.9333	0.0000	1.5862	0.0012	0.0000	100.0000	54.4815	0.0000
5.7618	1.4057	189.6184	0.0025	6.4667	0.3667	1.5973	0.0000	0.0000	100.0000	48.6819	0.0000
5.8627	1.4060	189.0533	0.0016	10.0667	0.0667	1.5600	0.0000	0.0000	100.0000	52.3726	0.0000

Continued on next page

Appendix G – continued from previous page

$rpoA$	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
5.9636	1.3829	189.6882	0.0017	3.8000	0.3667	1.5697	0.0019	0.0000	100.0000	49.7364	0.0000
6.0645	1.4738	189.5910	0.0017	8.0000	0.0000	1.5828	0.0000	0.0000	100.0000	52.5483	0.0000
6.1655	1.3921	187.9263	0.0025	4.2667	0.5667	1.5342	0.0006	0.0000	100.0000	48.5062	0.0000
6.2664	1.3480	187.4630	0.0021	2.8667	0.7333	1.5744	0.0000	0.0000	100.0000	47.9789	0.0000
6.3673	1.3664	189.4232	0.0025	8.6000	0.2333	1.5533	0.0006	0.0000	100.0000	52.5483	0.0000
6.4682	1.4738	190.2511	0.0017	9.9333	0.0000	1.5741	0.0000	0.0000	100.0000	52.0211	0.0000
6.5691	1.4040	189.8649	0.0019	5.6000	0.5667	1.5799	0.0006	0.0000	100.0000	46.0457	0.0000
6.6700	1.4231	188.2178	0.0019	8.0000	0.1333	1.5857	0.0006	0.0000	100.0000	48.6819	0.0000
6.7709	1.4276	188.4535	0.0016	11.4667	0.0333	1.5725	0.0006	0.0000	100.0000	50.6151	0.0000
6.8718	1.4005	191.3417	0.0018	7.0000	0.1333	1.5813	0.0000	0.0000	100.0000	52.7241	0.0000
6.9727	1.3836	192.2837	0.0017	9.1333	0.1000	1.5691	0.0000	0.0000	100.0000	52.1968	0.0000
7.0736	1.4113	189.2682	0.0021	9.7333	0.0000	1.5598	0.0012	0.0000	100.0000	50.0879	0.0000
7.1745	1.3697	190.0231	0.0020	7.3333	0.1667	1.5591	0.0006	0.0000	100.0000	55.1845	0.0000
7.2755	1.4210	189.3227	0.0019	9.8000	0.0333	1.5704	0.0000	0.0000	100.0000	52.1968	0.0000
7.3764	1.4310	188.7584	0.0016	12.5333	0.0000	1.5534	0.0006	0.0000	100.0000	50.2636	0.0000
7.4773	1.3847	188.6186	0.0022	10.6000	0.1000	1.5831	0.0000	0.0000	100.0000	48.5062	0.0000
7.5782	1.3809	189.3773	0.0020	6.7333	0.4333	1.5874	0.0000	0.0000	100.0000	50.4394	0.0000
7.6791	1.4400	190.2364	0.0019	10.2667	0.0000	1.5625	0.0000	0.0000	100.0000	55.7118	0.0000
7.7800	1.3975	191.6344	0.0009	9.4000	0.1667	1.5600	0.0000	0.0000	100.0000	53.2513	0.0000
7.8809	1.3609	189.9432	0.0021	3.6667	0.7000	1.5728	0.0000	0.0000	100.0000	50.6151	0.0000
7.9818	1.4565	187.6137	0.0019	9.0000	0.0333	1.5548	0.0000	0.0000	100.0000	50.7909	0.0000
8.0827	1.4561	188.6336	0.0015	8.6000	0.0333	1.5439	0.0006	0.0000	100.0000	49.2091	0.0000
8.1836	1.4527	189.2145	0.0021	9.8000	0.0000	1.5746	0.0012	0.0000	100.0000	51.1424	0.0000
8.2845	1.4019	191.4660	0.0010	9.2000	0.1000	1.5722	0.0006	0.0000	100.0000	52.0211	0.0000
8.3855	1.4801	189.6978	0.0025	8.4000	0.0000	1.5595	0.0012	0.0000	100.0000	52.3726	0.0000
8.4864	1.3772	189.4616	0.0023	6.7333	0.1333	1.5587	0.0006	0.0000	100.0000	53.4271	0.0000
8.5873	1.4687	189.0088	0.0016	9.7333	0.0000	1.5944	0.0000	0.0000	100.0000	50.0879	0.0000
8.6882	1.3654	187.9579	0.0027	5.5333	0.4667	1.5806	0.0000	0.0000	100.0000	52.1968	0.0000
8.7891	1.4158	189.4528	0.0018	10.5333	0.0667	1.6029	0.0006	0.0000	100.0000	52.1968	0.0000

Continued on next page

Appendix G **– continued from previous page**

$rpoA$	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
$\mathcal{N}(0, \sigma_r^2)$ σ_r	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
8.8900	1.4162	189.1413	0.0020	8.8000	0.0000	1.5844	0.0006	0.0000	100.0000	55.1845	0.0000
8.9909	1.4080	189.2320	0.0015	10.8667	0.0000	1.5734	0.0000	0.0000	100.0000	48.5062	0.0000
9.0918	1.3981	189.7351	0.0018	4.8000	0.4667	1.5756	0.0012	0.0000	100.0000	54.6573	0.0000
9.1927	1.4054	191.0460	0.0015	9.0667	0.1333	1.5696	0.0006	0.0000	100.0000	46.3972	0.0000
9.2936	1.3945	189.3845	0.0018	2.3333	0.7333	1.5687	0.0006	0.0000	100.0000	48.5062	0.0000
9.3945	1.4527	189.7793	0.0021	8.8667	0.0000	1.5652	0.0000	0.0000	100.0000	48.6819	0.0000
9.4955	1.3971	188.4598	0.0015	9.2000	0.1667	1.5595	0.0019	0.0000	100.0000	50.0879	0.0000
9.5964	1.3783	190.9483	0.0022	1.6000	0.4667	1.5948	0.0006	0.0000	100.0000	50.9666	0.0000
9.6973	1.4114	190.6226	0.0020	11.3333	0.0000	1.5964	0.0012	0.0000	100.0000	48.6819	0.0000
9.7982	1.3990	189.3769	0.0019	5.4667	0.4667	1.5696	0.0019	0.0000	100.0000	51.8453	0.0000
9.8991	1.3907	189.5692	0.0015	3.4000	0.6000	1.5788	0.0012	0.0000	100.0000	49.0334	0.0000
10.0000	1.4139	190.2545	0.0016	7.4667	0.0667	1.5627	0.0019	0.0000	100.0000	52.7241	0.0000

Appendix H1: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 3, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1676	196.6213	0.0119	7.8000	0.2333	0.2847	0.0606	0.0497	0.2299	86.4675	
0.0220	0.2178	197.4333	0.0107	7.8000	0.2333	0.3559	0.0452	0.0632	0.2299	87.5220	
0.0240	0.2739	198.2269	0.0093	7.8000	0.2333	0.4223	0.0303	0.0862	0.2299	89.2794	
0.0260	0.3144	197.0480	0.0079	7.8000	0.2333	0.4613	0.0285	0.1081	0.0000	77.1529	
0.0280	0.3480	197.3468	0.0077	7.8000	0.2333	0.4881	0.0167	0.1289	0.4598	82.2496	
0.0300	0.3938	198.1094	0.0076	6.4667	0.3000	0.5183	0.0105	0.1616	0.4598	86.8190	
0.0320	0.4197	198.7196	0.0071	6.4667	0.3000	0.5317	0.0056	0.1821	0.4598	88.9279	
0.0340	0.4594	199.2827	0.0064	6.4667	0.3000	0.5487	0.0037	0.2162	0.4598	88.9279	
0.0360	0.4889	198.8714	0.0061	6.4667	0.3000	0.5585	0.0043	0.2436	0.4598	90.8612	91.3884
0.0380	0.5041	199.4095	0.0067	8.6000	0.1667	0.5636	0.0074	0.2583	0.4598	90.6854	91.7399
0.0400	0.5231	199.4928	0.0060	8.6000	0.1667	0.5705	0.0056	0.2775	0.4598	89.1037	
0.0420	0.5404	200.1094	0.0059	8.6000	0.1667	0.5769	0.0031	0.2956	0.4598	87.8735	
0.0440	0.5554	200.6228	0.0052	8.6000	0.1667	0.5832	0.0043	0.3118	0.4598	86.6432	
0.0460	0.5664	200.5406	0.0050	8.6000	0.1667	0.5892	0.0025	0.3239	0.4598	85.9402	
0.0480	0.5760	201.7858	0.0047	8.6000	0.1667	0.5949	0.0056	0.3348	0.4598	85.0615	
0.0500	0.5943	201.4410	0.0054	8.6000	0.1667	0.6068	0.0031	0.3560	0.4598	83.4798	
0.0520	0.6134	201.4789	0.0050	8.6000	0.1667	0.6177	0.0050	0.3788	0.4598	82.2496	
0.0540	0.6335	201.9500	0.0046	8.6000	0.1667	0.6294	0.0025	0.4039	0.4598	80.6678	
0.0560	0.6453	201.6102	0.0044	8.6000	0.1667	0.6375	0.0012	0.4188	0.4598	79.7891	
0.0580	0.6649	202.0446	0.0044	8.6000	0.1667	0.6489	0.0025	0.4444	0.4598	78.3831	
0.0600	0.6752	202.1166	0.0043	8.6000	0.1667	0.6562	0.0019	0.4583	0.4598	77.6801	

Original accuracies: $\mathcal{K}$-means= 92.79%, SVMlight = 96.49%.

Parameters in SVMlight: 10-fold crossvalidation, rbf kernel function, $\gamma = 1$.

Appendix H2: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 4, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	196.0153	0.0133	7.3333	0.3333	0.2849	0.0606	0.0500	0.2299	85.4130	
0.0220	0.2171	196.4950	0.0117	6.0000	0.4000	0.3560	0.0452	0.0635	0.2299	86.4675	
0.0240	0.2733	195.8716	0.0106	6.0000	0.4000	0.4223	0.0353	0.0865	0.0000	87.8735	
0.0260	0.3140	197.4021	0.0098	6.0000	0.4000	0.4612	0.0303	0.1083	0.9195	90.3339	
0.0280	0.3476	196.2722	0.0092	6.0000	0.4000	0.4879	0.0167	0.1291	0.9195	82.2496	
0.0300	0.3935	197.4905	0.0084	6.0000	0.4000	0.5179	0.0099	0.1618	0.9195	86.8190	
0.0320	0.4194	196.6559	0.0082	6.0000	0.4000	0.5312	0.0074	0.1822	0.9195	88.9279	
0.0340	0.4591	197.2141	0.0081	6.0000	0.4000	0.5481	0.0050	0.2163	0.9195	88.9279	
0.0360	0.4886	197.6087	0.0070	6.0000	0.4000	0.5576	0.0062	0.2437	0.9195	90.8612	91.2127
0.0380	0.5039	197.7193	0.0074	7.6667	0.2333	0.5625	0.0056	0.2584	0.9195	90.6854	90.8612
0.0400	0.5229	198.1725	0.0070	8.8000	0.1667	0.5692	0.0037	0.2776	0.9195	89.1037	
0.0420	0.5402	199.0519	0.0067	8.8000	0.1667	0.5757	0.0068	0.2956	0.2299	87.8735	
0.0440	0.5552	199.6280	0.0059	8.8000	0.1667	0.5820	0.0043	0.3119	0.2299	86.6432	
0.0460	0.5662	200.2668	0.0056	8.8000	0.1667	0.5879	0.0012	0.3239	0.0000	85.9402	
0.0480	0.5759	200.9586	0.0057	8.8000	0.1667	0.5934	0.0062	0.3348	0.0000	85.0615	
0.0500	0.5942	201.0861	0.0057	8.8000	0.1667	0.6054	0.0025	0.3560	0.0000	83.4798	
0.0520	0.6132	201.1830	0.0054	8.8000	0.1667	0.6163	0.0050	0.3788	0.0000	82.2496	
0.0540	0.6333	201.8586	0.0050	8.8000	0.1667	0.6281	0.0019	0.4039	0.0000	80.6678	
0.0560	0.6451	201.7866	0.0047	8.8000	0.1667	0.6362	0.0056	0.4188	0.0000	79.7891	
0.0580	0.6647	202.1588	0.0050	8.8000	0.1667	0.6474	0.0000	0.4444	0.0000	78.3831	
0.0600	0.6751	202.1618	0.0050	8.8000	0.1667	0.6549	0.0025	0.4583	0.0000	77.6801	

Appendix H3: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 7, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	194.9809	0.0145	6.6667	0.3667	0.2850	0.0606	0.0500	0.0000	86.6432	
0.0220	0.2171	195.6281	0.0122	6.6667	0.3667	0.3561	0.0421	0.0635	0.0000	87.1705	
0.0240	0.2733	195.5047	0.0105	5.3333	0.4333	0.4223	0.0303	0.0865	0.2299	87.3462	
0.0260	0.3139	195.9868	0.0097	5.3333	0.4333	0.4612	0.0248	0.1083	0.4598	77.1529	
0.0280	0.3476	195.7181	0.0092	5.6667	0.4000	0.4879	0.0149	0.1291	0.4598	82.2496	
0.0300	0.3934	195.5414	0.0084	5.6667	0.4000	0.5178	0.0099	0.1618	0.2299	86.8190	
0.0320	0.4194	196.2844	0.0077	5.6667	0.4000	0.5310	0.0087	0.1822	0.2299	88.9279	
0.0340	0.4591	196.2247	0.0080	5.6667	0.4000	0.5478	0.0068	0.2163	0.0000	88.9279	
0.0360	0.4886	196.0144	0.0071	5.6667	0.4000	0.5571	0.0043	0.2436	0.0000	90.8612	91.5641
0.0380	0.5038	196.7210	0.0080	7.0667	0.2333	0.5619	0.0050	0.2584	0.0000	90.6854	91.5641
0.0400	0.5229	196.7913	0.0076	8.3333	0.1667	0.5684	0.0056	0.2775	0.0000	89.1037	
0.0420	0.5402	197.0479	0.0068	8.5333	0.1667	0.5747	0.0019	0.2956	0.0000	87.8735	
0.0440	0.5552	196.8544	0.0060	8.2000	0.1667	0.5808	0.0062	0.3119	0.0000	86.6432	
0.0460	0.5662	197.5049	0.0062	8.2000	0.1667	0.5866	0.0037	0.3239	0.0000	85.9402	
0.0480	0.5758	197.4678	0.0060	8.2000	0.1667	0.5921	0.0050	0.3348	0.0000	85.0615	
0.0500	0.5941	198.3399	0.0062	8.2000	0.1667	0.6040	0.0031	0.3560	0.0000	83.4798	
0.0520	0.6132	198.8381	0.0054	8.5333	0.1667	0.6149	0.0012	0.3788	0.2299	82.2496	
0.0540	0.6333	198.9206	0.0053	8.2000	0.1667	0.6265	0.0031	0.4039	0.2299	80.6678	
0.0560	0.6451	199.8449	0.0047	8.2000	0.1667	0.6346	0.0043	0.4188	0.2299	79.7891	
0.0580	0.6647	200.2224	0.0050	8.2000	0.1667	0.6458	0.0050	0.4444	0.4598	78.3831	
0.0600	0.6751	200.2861	0.0052	8.2000	0.1667	0.6533	0.0012	0.4583	0.0000	77.6801	

Appendix H4: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 20, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	184.5486	0.0185	7.9333	0.3667	0.2850	0.0600	0.0500	0.0000	87.8735	
0.0220	0.2171	185.3241	0.0166	6.9333	0.4000	0.3561	0.0427	0.0636	0.4598	89.4552	
0.0240	0.2733	186.0995	0.0147	4.0000	0.5667	0.4223	0.0303	0.0865	0.2299	70.8260	
0.0260	0.3139	186.4344	0.0126	5.2000	0.4667	0.4612	0.0254	0.1083	0.0000	77.1529	
0.0280	0.3476	184.9814	0.0114	4.0000	0.4333	0.4879	0.0173	0.1291	0.0000	82.2496	
0.0300	0.3934	184.3631	0.0125	5.8000	0.4333	0.5178	0.0099	0.1618	0.0000	86.8190	
0.0320	0.4194	187.0567	0.0114	6.5333	0.4333	0.5310	0.0074	0.1822	0.0000	88.9279	
0.0340	0.4591	185.6988	0.0107	6.0667	0.4000	0.5478	0.0080	0.2163	0.0000	88.9279	
0.0360	0.4886	185.9220	0.0108	4.6000	0.4667	0.5571	0.0068	0.2436	0.0000	90.8612	92.4429
0.0380	0.5038	185.5426	0.0115	7.4667	0.3000	0.5619	0.0031	0.2584	0.2299	90.6854	91.9156
0.0400	0.5229	188.0767	0.0108	8.0667	0.1667	0.5684	0.0043	0.2775	0.2299	89.1037	
0.0420	0.5402	188.1541	0.0100	8.1333	0.2000	0.5747	0.0025	0.2956	0.0000	87.8735	
0.0440	0.5552	186.5085	0.0087	7.8000	0.2000	0.5808	0.0043	0.3119	0.0000	86.6432	
0.0460	0.5662	186.5142	0.0088	7.9333	0.1667	0.5866	0.0043	0.3239	0.0000	85.9402	
0.0480	0.5758	188.0746	0.0086	7.4000	0.1667	0.5920	0.0031	0.3348	0.2299	85.0615	
0.0500	0.5941	188.4041	0.0088	7.2667	0.1667	0.6040	0.0031	0.3560	0.2299	83.4798	
0.0520	0.6132	188.7114	0.0081	6.4000	0.2000	0.6148	0.0025	0.3788	0.0000	82.2496	
0.0540	0.6333	189.1973	0.0064	6.6000	0.2000	0.6265	0.0031	0.4039	0.2299	80.6678	
0.0560	0.6451	190.9414	0.0075	6.6667	0.1667	0.6345	0.0037	0.4188	0.0000	79.7891	
0.0580	0.6647	190.3899	0.0071	6.6667	0.1667	0.6457	0.0050	0.4444	0.2299	78.3831	
0.0600	0.6751	190.8435	0.0071	5.6667	0.2000	0.6532	0.0025	0.4583	0.2299	77.6801	

Appendix H5: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 22, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	183.8841	0.0187	8.1333	0.3333	0.2850	0.0600	0.0500	0.0000	87.8735	0.0000
0.0220	0.2171	185.5681	0.0158	6.8667	0.4000	0.3561	0.0427	0.0636	0.2299	89.4552	0.0000
0.0240	0.2733	184.5250	0.0144	3.9333	0.5667	0.4223	0.0303	0.0865	0.0000	70.8260	0.0000
0.0260	0.3139	185.5365	0.0130	5.2000	0.5000	0.4612	0.0254	0.1083	0.0000	77.1529	0.0000
0.0280	0.3476	185.9583	0.0126	3.7333	0.4333	0.4879	0.0173	0.1291	0.0000	82.2496	0.0000
0.0300	0.3934	183.7070	0.0132	5.8667	0.4333	0.5178	0.0099	0.1618	0.2299	86.8190	0.0000
0.0320	0.4194	187.6572	0.0109	5.2000	0.4333	0.5310	0.0074	0.1822	0.2299	88.9279	0.0000
0.0340	0.4591	184.9988	0.0108	4.6000	0.4333	0.5478	0.0080	0.2163	0.0000	88.9279	0.0000
0.0360	0.4886	185.4096	0.0110	4.6000	0.4667	0.5571	0.0043	0.2436	0.9195	90.8612	92.0914
0.0380	0.5038	186.0745	0.0115	7.4000	0.2667	0.5619	0.0031	0.2584	0.2299	90.6854	92.9701
0.0400	0.5229	187.9649	0.0100	6.9333	0.1667	0.5684	0.0043	0.2775	0.4598	89.1037	0.0000
0.0420	0.5402	186.4956	0.0096	8.2000	0.1667	0.5747	0.0019	0.2956	0.0000	87.8735	0.0000
0.0440	0.5552	187.1188	0.0088	6.8000	0.1667	0.5808	0.0062	0.3119	0.0000	86.6432	0.0000
0.0460	0.5662	186.0724	0.0084	7.9333	0.1667	0.5866	0.0031	0.3239	0.0000	85.9402	0.0000
0.0480	0.5758	187.5862	0.0080	7.8667	0.1667	0.5920	0.0037	0.3348	0.0000	85.0615	0.0000
0.0500	0.5941	188.1748	0.0086	7.4000	0.1667	0.6040	0.0031	0.3560	0.0000	83.4798	0.0000
0.0520	0.6132	187.2811	0.0078	7.8000	0.1667	0.6148	0.0037	0.3788	0.2299	82.2496	0.0000
0.0540	0.6333	187.3752	0.0071	6.8000	0.2000	0.6265	0.0025	0.4039	0.2299	80.6678	0.0000
0.0560	0.6451	189.9262	0.0074	6.6667	0.1667	0.6345	0.0025	0.4188	0.0000	79.7891	0.0000
0.0580	0.6647	189.9475	0.0077	6.6667	0.1667	0.6457	0.0056	0.4444	0.0000	78.3831	0.0000
0.0600	0.6751	189.7095	0.0074	5.6667	0.2000	0.6532	0.0012	0.4583	0.2299	77.6801	0.0000

Original accuracies: $\mathcal{K}$-means= 92.79%, SVMlight = 96.49%.

Parameters in SVMlight: 10-fold crossvalidation, rbf kernel function, $\gamma = 1$.

Appendix H6: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 23, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	184.8777	0.0188	7.9333	0.3333	0.2850	0.0600	0.0500	0.0000	87.8735	0.0000
0.0220	0.2171	185.7134	0.0165	6.2667	0.4000	0.3561	0.0427	0.0636	0.2299	89.4552	0.0000
0.0240	0.2733	185.2272	0.0141	3.9333	0.5333	0.4223	0.0303	0.0865	0.0000	70.8260	0.0000
0.0260	0.3139	185.0583	0.0128	5.6667	0.4333	0.4612	0.0254	0.1083	0.0000	77.1529	0.0000
0.0280	0.3476	186.1620	0.0118	3.7333	0.4333	0.4879	0.0173	0.1291	0.2299	82.2496	0.0000
0.0300	0.3934	184.8374	0.0131	6.4000	0.4333	0.5178	0.0099	0.1618	0.0000	86.8190	0.0000
0.0320	0.4194	186.6097	0.0114	5.1333	0.4667	0.5310	0.0074	0.1822	0.2299	88.9279	0.0000
0.0340	0.4591	185.7107	0.0112	5.0667	0.4333	0.5478	0.0080	0.2163	0.0000	88.9279	0.0000
0.0360	0.4886	186.7781	0.0100	4.6000	0.4333	0.5571	0.0043	0.2436	0.0000	90.8612	91.7399
0.0380	0.5038	186.8480	0.0113	5.8000	0.3000	0.5619	0.0031	0.2584	0.0000	90.6854	92.6186
0.0400	0.5229	187.9749	0.0104	6.9333	0.1667	0.5684	0.0037	0.2775	0.2299	89.1037	0.0000
0.0420	0.5402	186.6457	0.0097	7.1333	0.1667	0.5747	0.0019	0.2956	0.0000	87.8735	0.0000
0.0440	0.5552	187.4869	0.0091	6.8000	0.1667	0.5808	0.0062	0.3119	0.2299	86.6432	0.0000
0.0460	0.5662	186.3084	0.0088	7.9333	0.1667	0.5866	0.0031	0.3239	0.6897	85.9402	0.0000
0.0480	0.5758	186.8261	0.0086	7.9333	0.1667	0.5920	0.0031	0.3348	0.2299	85.0615	0.0000
0.0500	0.5941	188.1479	0.0093	7.2000	0.1667	0.6040	0.0031	0.3560	0.0000	83.4798	0.0000
0.0520	0.6132	187.6658	0.0085	7.6000	0.1667	0.6148	0.0037	0.3788	0.4598	82.2496	0.0000
0.0540	0.6333	187.0770	0.0073	6.6000	0.2000	0.6265	0.0025	0.4039	0.0000	80.6678	0.0000
0.0560	0.6451	189.1809	0.0079	5.6000	0.2000	0.6345	0.0037	0.4188	0.0000	79.7891	0.0000
0.0580	0.6647	189.5367	0.0079	6.6667	0.1667	0.6457	0.0037	0.4444	0.0000	78.3831	0.0000
0.0600	0.6751	189.7965	0.0077	6.4667	0.2000	0.6532	0.0031	0.4583	0.2299	77.6801	0.0000

Original accuracies: $\mathcal{K}$-means= 92.79%, SVMlight = 96.49%.

Parameters in SVMlight: 10-fold crossvalidation, rbf kernel function, $\gamma = 1$.

Appendix H7: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 25, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	183.6507	0.0186	8.0667	0.3333	0.2850	0.0600	0.0500	0.0000	87.8735	0.0000
0.0220	0.2171	185.7795	0.0165	6.8667	0.4000	0.3561	0.0427	0.0636	0.2299	89.4552	0.0000
0.0240	0.2733	184.5114	0.0141	3.8667	0.5333	0.4223	0.0303	0.0865	0.0000	70.8260	0.0000
0.0260	0.3139	184.9035	0.0129	5.6667	0.4333	0.4612	0.0254	0.1083	0.0000	77.1529	0.0000
0.0280	0.3476	185.5289	0.0120	4.2667	0.4000	0.4879	0.0173	0.1291	0.2299	82.2496	0.0000
0.0300	0.3934	184.5729	0.0131	5.8667	0.4333	0.5178	0.0099	0.1618	0.2299	86.8190	0.0000
0.0320	0.4194	186.6777	0.0107	5.0667	0.5000	0.5310	0.0074	0.1822	0.2299	88.9279	0.0000
0.0340	0.4591	186.1159	0.0114	4.5333	0.4667	0.5478	0.0080	0.2163	0.0000	88.9279	0.0000
0.0360	0.4886	187.1899	0.0103	4.5333	0.4667	0.5571	0.0043	0.2436	0.0000	90.8612	91.9156
0.0380	0.5038	186.3745	0.0105	5.8667	0.3333	0.5619	0.0043	0.2584	0.4598	90.6854	92.6168
0.0400	0.5229	187.9503	0.0108	6.8667	0.1667	0.5684	0.0043	0.2775	0.0000	89.1037	0.0000
0.0420	0.5402	186.3731	0.0100	7.0667	0.1667	0.5747	0.0012	0.2956	0.2299	87.8735	0.0000
0.0440	0.5552	185.6280	0.0092	6.8000	0.1667	0.5808	0.0087	0.3119	0.0000	86.6432	0.0000
0.0460	0.5662	187.3252	0.0088	7.9333	0.1667	0.5866	0.0043	0.3239	0.0000	85.9402	0.0000
0.0480	0.5758	187.3245	0.0089	7.8667	0.1667	0.5920	0.0031	0.3348	0.0000	85.0615	0.0000
0.0500	0.5941	187.5421	0.0089	7.4000	0.1667	0.6040	0.0019	0.3560	0.0000	83.4798	0.0000
0.0520	0.6132	186.4991	0.0088	7.8000	0.1667	0.6148	0.0037	0.3788	0.2299	82.2496	0.0000
0.0540	0.6333	187.2439	0.0076	6.6000	0.2000	0.6265	0.0050	0.4039	0.0000	80.6678	0.0000
0.0560	0.6451	189.7227	0.0078	5.6000	0.2000	0.6345	0.0043	0.4188	0.0000	79.7891	0.0000
0.0580	0.6647	189.4450	0.0076	5.6000	0.2000	0.6457	0.0043	0.4444	0.2299	78.3831	0.0000
0.0600	0.6751	189.8929	0.0079	6.4667	0.1667	0.6532	0.0012	0.4583	0.4598	77.6801	0.0000

Original accuracies: $\mathcal{K}$-means= 92.79%, SVMlight = 96.49%.

Parameters in SVMlight: 10-fold crossvalidation, rbf kernel function, $\gamma = 1$.

Appendix H8: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 27, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1667	184.4560	0.0186	8.0667	0.3333	0.2850	0.0600	0.0500	0.2299	87.8735	0.0000
0.0220	0.2171	185.1629	0.0165	6.8667	0.4000	0.3561	0.0427	0.0636	0.2299	89.9824	0.0000
0.0240	0.2733	186.5994	0.0152	3.7333	0.5333	0.4223	0.0303	0.0865	0.4598	70.8260	0.0000
0.0260	0.3139	186.0260	0.0136	5.6667	0.4333	0.4612	0.0254	0.1083	0.0000	77.1529	0.0000
0.0280	0.3476	185.3991	0.0125	4.2667	0.4000	0.4879	0.0173	0.1291	0.4598	82.2496	0.0000
0.0300	0.3934	185.4814	0.0128	6.4000	0.4333	0.5178	0.0099	0.1618	0.0000	86.8190	0.0000
0.0320	0.4194	187.5504	0.0112	5.0667	0.5000	0.5310	0.0074	0.1822	0.0000	88.9279	0.0000
0.0340	0.4591	185.6567	0.0111	4.6000	0.4333	0.5478	0.0080	0.2163	0.0000	88.9279	0.0000
0.0360	0.4886	186.0928	0.0108	4.6000	0.4333	0.5571	0.0043	0.2436	0.2299	90.8612	92.2671
0.0380	0.5038	186.0368	0.0116	5.8667	0.3333	0.5619	0.0043	0.2584	0.2299	90.6854	91.9156
0.0400	0.5229	187.0673	0.0113	6.8667	0.1667	0.5684	0.0043	0.2775	0.2299	89.1037	0.0000
0.0420	0.5402	185.9545	0.0106	7.0667	0.1667	0.5747	0.0012	0.2956	0.2299	87.8735	0.0000
0.0440	0.5552	186.2619	0.0091	6.8000	0.1667	0.5808	0.0080	0.3119	0.0000	86.6432	0.0000
0.0460	0.5662	185.9756	0.0091	7.9333	0.1667	0.5866	0.0043	0.3239	0.0000	85.9402	0.0000
0.0480	0.5758	186.6963	0.0094	7.8667	0.1667	0.5920	0.0050	0.3348	0.0000	85.0615	0.0000
0.0500	0.5941	187.8497	0.0095	7.4000	0.1667	0.6040	0.0025	0.3560	0.0000	83.4798	0.0000
0.0520	0.6132	187.5231	0.0095	7.6000	0.1667	0.6148	0.0043	0.3788	0.4598	82.2496	0.0000
0.0540	0.6333	187.9401	0.0076	6.6000	0.2000	0.6265	0.0050	0.4039	0.0000	80.6678	0.0000
0.0560	0.6451	190.2349	0.0084	5.6000	0.2000	0.6345	0.0056	0.4188	0.0000	79.7891	0.0000
0.0580	0.6647	189.0025	0.0081	5.6000	0.2000	0.6457	0.0019	0.4444	0.0000	78.3831	0.0000
0.0600	0.6751	189.5523	0.0082	6.4667	0.2000	0.6532	0.0043	0.4583	0.2299	77.6801	0.0000

Original accuracies: $\mathcal{K}$-means= 92.79%, SVMlight = 96.49%.

Parameters in SVMlight: 10-fold crossvalidation, rbf kernel function, $\gamma = 1$.

Appendix H9: the Sparsified SVD-based data modification: s-SVD on WDBC (569×30). $rank = 1, \epsilon_v = 0.02$

s-SVD	Data Value Distortion					Data Pattern Distortion (- % - %)				Mining Accuracy (%)	
ϵ_u	**RE**	**RP**	**RK**	**CP**	**CK**	**DistVal**	**Dist** Maintain	**CorrVal**	**Corr** Maintain	**$\mathcal{K}$-means**	**SVMlight**
0.0200	0.1872	199.7647	0.0049	9.1333	0.2000	0.2847	0.0340	0.0483	0.2299	86.8190	0.0000
0.0220	0.2329	199.6620	0.0043	9.1333	0.2000	0.3554	0.0278	0.0612	0.2299	86.6432	0.0000
0.0240	0.2858	199.5709	0.0049	9.1333	0.2000	0.4217	0.0198	0.0841	0.2299	88.9279	0.0000
0.0260	0.3246	199.6294	0.0046	9.1333	0.2000	0.4612	0.0167	0.1057	0.2299	90.8612	0.0000
0.0280	0.3570	199.5579	0.0041	9.1333	0.2000	0.4886	0.0118	0.1266	0.2299	82.2496	0.0000
0.0300	0.4014	199.9155	0.0040	9.1333	0.2000	0.5198	0.0093	0.1592	0.2299	86.8190	0.0000
0.0320	0.4265	199.7323	0.0042	9.1333	0.2000	0.5338	0.0037	0.1796	0.2299	88.9279	0.0000
0.0340	0.4654	200.0318	0.0037	9.1333	0.2000	0.5524	0.0068	0.2138	0.2299	88.9279	0.0000
0.0360	0.4943	199.8766	0.0037	9.1333	0.2000	0.5622	0.0025	0.2414	0.2299	90.8612	0.0000
0.0380	0.5092	200.0475	0.0037	9.1333	0.2000	0.5674	0.0074	0.2562	0.2299	90.6854	0.0000
0.0400	0.5279	200.2960	0.0033	9.1333	0.2000	0.5737	0.0043	0.2756	0.2299	89.1037	0.0000
0.0420	0.5449	200.5802	0.0034	9.1333	0.2000	0.5800	0.0050	0.2937	0.2299	87.8735	0.0000
0.0440	0.5597	200.8043	0.0033	9.1333	0.2000	0.5864	0.0043	0.3101	0.2299	86.6432	0.0000
0.0460	0.5704	200.9503	0.0033	9.1333	0.2000	0.5915	0.0037	0.3222	0.2299	85.9402	0.0000
0.0480	0.5800	200.9996	0.0032	9.1333	0.2000	0.5969	0.0037	0.3332	0.2299	85.0615	0.0000
0.0500	0.5980	201.2574	0.0032	9.1333	0.2000	0.6074	0.0037	0.3545	0.2299	83.4798	0.0000
0.0520	0.6169	201.6301	0.0032	9.1333	0.2000	0.6185	0.0025	0.3774	0.2299	82.2496	0.0000
0.0540	0.6369	201.8900	0.0029	9.1333	0.2000	0.6306	0.0074	0.4026	0.2299	80.6678	0.0000
0.0560	0.6485	202.0480	0.0029	9.1333	0.2000	0.6380	0.0019	0.4176	0.2299	79.7891	0.0000
0.0580	0.6680	202.0944	0.0028	9.1333	0.2000	0.6509	0.0050	0.4433	0.2299	78.3831	0.0000
0.0600	0.6783	202.1714	0.0029	9.1333	0.2000	0.6581	0.0025	0.4572	0.2299	77.6801	0.0000

Original accuracies: $\mathcal{K}$-means= 92.79%, SVMlight = 96.49%.

Parameters in SVMlight: 10-fold crossvalidation, rbf kernel function, $\gamma = 1$.

Bibliography

[1] UCI Machine Learning Repository, http://www.ics.uci.edu/ mlearn/mlrepository.html.

[2] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.

[3] A. L. Abul-Ela, B. G.Greenberg, and D. G.Horvitz. A multi-proportions randomized response model. *J.Am.Stat.Assoc*, 62(319):990–1008, 1967.

[4] Dimitris Achlioptas. Random matrices in data analysis. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 1–7, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[5] Nabil R. Adam and John C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, December 1989.

[6] A.F.Karr, A.P.Sanil, and D.L.Banks. Data quality: A statistical perspective., 2005.

[7] Charu C. Aggarwal and Philip S. Yu. A condensation approach to privacy preserving data mining. In *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004*, pages 183–199, 2004.

[8] Charu C. Aggarwal and Philip S. Yu, editors. *Privacy-preserving data mining models and algorithms*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2008.

[9] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, Texas, May 2000. ACM Press.

[10] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*, pages 45–52, 1999.

[11] X.Y. Wu B. Long, Z.F. Zhang and P.S. Yu. Relational clustering by symmetric convex coding. In Z. Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 569–576, Corvallis, OR, 2007. Omnipress.

[12] M. Berry, M. Browne, A. Langville, P. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 2006.

[13] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1995.

[14] E. Bertino, I. N. Fovino, and L. P. Provenza. A framework for evaluating privacy preserving data mining algorithms. *Data Mining and Knowledge Discovery*, 11(2):121–154, 2005.

[15] P. Raghavan C.D. Manning and H. Schiitze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[16] Keke Chen and Ling Liu. Privacy preserving data classification with rotation perturbation. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 589–592, Washington, DC, USA, 2005. IEEE Computer Society.

[17] Keke Chen, Gordon Sun, and Ling Liu. Towards attack-resilient geometric data perturbation. In *Proceedings of the 2007 SIAM International Conference on Data Mining.*, April 2007.

[18] A. Cichocki and R. Zdunek. *NMFLAB – MATLAB toolbox for non-negative matrix factorization*. The Laboratory for Advanced Brain Signal Processing (ABSP), Japan.

[19] Chris Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2):1–7, 2003.

[20] Chris Clifton and Don Marks. Security and privacy implication of data mining. In *Proceedings of the Workshop on Data Mining and Knowledge Discovery*, number 96-08, pages 15–19, Montreal, Canada, June 1996. University of British Columbia Department of Computer Science.

[21] R. Conway and D. Strip. Selective partial access to a database. In *ACM Annual Conference/Annual Meeting Proceedings of the annual conference*, pages 85–89. ACM, 1976.

[22] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Wiley, 1991.

[23] L. Cranor and editor. Special issue on internet privacy. *Comm. ACM*, 42(2), 1999.

[24] C.C D.Agrawal, Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGACT-SIGMODSIGART Symposium on Principles of Database Systems*, 2001.

[25] Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. In *Lecture Notes In Computer Science; Vol. 2137, Proceedings of the 4th International Workshop on Information Hiding*, pages 369–383. Springer-Verlag, April 2001.

[26] C. Ding, X. He, and H. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of SIAM Data Mining Conference*, 2005.

[27] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts. In *In Proc. NIPS*, 2003.

[28] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *2001 Workshop on New Security Paradigms*, Cloudcraft, NM., 2001.

[29] C. Eckart and G. Young. The approximation of one matrix by another of low rank. *Psychometrika*, 1(3):211–218, 1936.

[30] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.

[31] W. Frawley, G.Piatesky-Shapiro, and C.Matheus. Knowledge discovery in databases: an overview. *AI Magazine*, pages 213–228, 1992.

[32] J. Gao and J. Zhang. Clustered svd strategies in latent semantic indexing. *Information Processing and Management*, 41(5):1051–1063, 2005.

[33] G. Golub and C.Van. Loan. *Matrix Computation*. Johns Hopkins, Baltimore, 2nd edition, 1989.

[34] Devid Guillamet and Jordi Vitria. Determining a suitable metric when using non-negative matrix factorization. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, pages 20128–20131, aug 11-15 2002.

[35] D. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. MIT Press, Cambridge, MA, 2001.

[36] Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private information from randomized data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA,2005*, pages 37–48, June 14-16 2005.

[37] L. Hubert, J. Meulman, and W. Heiser. Two purposes for matrix factorization: a historical appraisal. *SIAM Review*, 42(4):68–82, 2000.

[38] L. Wo J. T.Giles and M. W. Berry. *GTP(general text parser) software for text mining*, pages 455–471. CRC Press, Boca Raton, 2003.

[39] J.M.Mateo-Sanz, A.M.Balleste, and J.D.Ferrer. Fast generation of accurate synthetic microdata. In *Lecture Notes in Computer Science*, volume 3050, chapter Privacy in statistical databases, pages 298–306. Berlin: Springer-Verlag, 2004.

[40] T. Joachims. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning.* MIT-Press, 1999.

[41] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[42] M. Juvela, K. Lehtinen, and P. Paatero. The use of positive matrix factorization in the analysis of molecular line spectra from the thumbprint nebula. In *Proceedings of the Fourth Haystack Conference on Clouds, Cores and Low Mass Stars*, volume 65, pages 176–180. Astronomical Society of the Pacific Conference Series, 1994.

[43] H. Kargupt K. Liu and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining . *IEEE Transactions on Knowledge and Data Engineering*, 18(1):92–106, 2006.

[44] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 99–106. IEEE Computer Society, 2003.

[45] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information System*, 7(4):387–414, 2005.

[46] Daniel Kifer and Johannes Gehrke. Injecting utility into anonymized datasets. In *SIGMOD Conference*, pages 217–228, 2006.

[47] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[48] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[49] D. D. Lee and H. S. Seung. Intelligence and security informatics for homeland security: Information, communication, and transportation. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):329–341, 2004.

[50] Chong K. Liew, Uinam J. Choi, and Chung J. Liew. A data distortion by probability distribution. *ACM Trans. Database Syst.*, 10(3):395–411, 1985.

[51] C.J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[52] Lian Liu, Jie Wang, and Jun Zhang. Wavelet-based data perturbation for simultaneous privacy-preserving and statistics-presering. In *8th IEEE International Conference on 8th IEEE International Conference on Data Mining (ICDM 08) - Workshops, icdmw. the 2nd Workshop on Reliability Issues in Knowledge Discovery.*, Pisa, Italy, December 2008.

[53] W. Liu and J. Yi. Existing and new algorithms for non-negative matrix factorization. Technical report, Computer Sciences Dept., UT Austin, 2003.

[54] M.L. Mahta. *Random matrices*. Academic, London, 2nd edition, 1991.

[55] B. Marlin and L. Sweeney. Determining the identifiability of dna database entries. *Journal of the American Medical Informatics Association.*, pages 537–541, November 2000.

[56] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford*, 11:50–59, 1960.

[57] K. Muralidhar and K. Batra et al. Accessibility, security and accuracy in statistical databases: the case for multiplicative fixed data perturbation approach. *Management Science*, 9(4):1549–1564, 1995.

[58] Krishnamurty Muralidhar and Rathindra Sarathy. Security of random data perturbation methods. *ACM Trans. Database Syst.*, 24(4):487–493, 1999.

[59] Juggapong Natwichai, Xue Li, and Maria E. Orlowska. Hiding classification rules for data sharing with privacy preservation. In *Proceedings of 7th International ConferenceData Warehousing and Knowledge Discovery (DaWak 2005)*, pages 468–477, August 2005.

[60] Juggapong Natwichai, Xue Li, and Maria E. Orlowska. A reconstruction-based algorithm for classification rules hiding. In *Database Technologies 2006, Proceedings of the 17th Australasian Database Conference*, pages 49–58, Hobart, Tasmania, Australia, January 2006.

[61] E. Newton, L. Sweeney, and B. Malin. Preserving privacy by de-identifying facial images. *IEEE Transaction on Knowledge and Data Engineering*, 17(2):232–243, Feburary 2005.

[62] D.E. O'Leary. Knowledge discovery as a threat to database security. In *Proceedings of the First International Conference on Knowledge Discovery and Databases*, pages 507–517, 1991.

[63] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[64] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J.Plemmons. Text mining using non-negative matrix factorizations. In *Proceedings of the 4th SIAM International Conference on Data Minin*, pages 452–456, 2004.

[65] P.O.Hoyer. Non-negative sparse coding. In *Proceedings of 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[66] Huseyin Polat and Wenliang Du. Svd-based collaborative filtering with privacy. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 791–795, New York, NY, USA, 2005. ACM Press.

[67] J. P. Reiss. Practical data swapping: the first steps. In *Proceeding of IEEE Symposium on Security and Privacy*, pages 36–44, 1980.

[68] J. P. Reiss. Practical data swapping: the first steps. *ACM Transaction on Database Systems*, 9(1):20–37, 1984.

[69] Arie Shoshani. Statistical databases: Characteristics, problems, and some solutions. In *Proceedings of Eighth International Conference on Very Large Data Bases*, pages 208–222, Mexico City, Mexico, September 1982. Morgan Kaufmann.

[70] G. W. Stewart. Perturbation theory of the singular value decomposition. Technical Report CS-TR-2539, University of Maryland, College park, MD., September 1990.

[71] L. Sweeney and R.Gross. Mining images in publicly-available cameras for homeland security. In *AAAI Spring Symposium AI Technologies for Homeland Security.*, 2005.

[72] Michael Totty. The dangers within. *The Wall Street Journal*, February 2006.

[73] Jane E. Tougas and Raymod J. Spiteri. Updating the partial singular value decomposition in latent semantic indexing. *Computational Statistics and Data Analysis*, 52:174–183, 2007.

[74] Joseph F. Traub, Yechiam Yemini, and Henryk Wozniakowski. The statistical security of a statistical database. *ACM Transactions on Database Systems*, 9(4):672–679, 1984.

[75] J. Vaidya and C. Clifton. Privacy-preserving data mining: why, how, and when. *IEEE Security and Privacy*, pages 19–27, November/December 2004.

[76] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[77] Vassilios S. Verykios, Elisa Bertino, I.N. Fovino, L.P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[78] Vassilios S. Verykios, Ahmed K Elmagarmid, Elisa Bertino, Yucel Saygin, and Elena Dasseni. Association rule hiding. *IEEE Transaction on Knowledge and Data Engineering*, 16(4):414–447, April 2004.

[79] Jie Wang, W.J.Zhong, J.Zhang, and S.T.Xu. Selective data distortion via structural partition and ssvd for privacy preservation. In *Proceedings of the 2006 International conference on Information Knowledge Engineering*, pages 114–120, Las Vegas, Nevada, USA, June 2006. CSREA Press.

[80] Jie Wang, Jun Zhang, Lian Liu, and Dianwei Han. Simultaneous data and pattern hiding in unsupervised learning. In *The 7th IEEE International Conference on Data Mining - Workshops(ICDMW'07)*, pages 729–734, Omaha, NE, USA, October 2007. IEEE Computer Society.

[81] Jie Wang, Jun Zhang, Shuting Xu, and Weijun Zhong. A novel data distortion approach via selective SSVD for privacy protection. *International Journal of Information and Computer Security*, 2007. to appear.

[82] Jie Wang, Weijun Zhong, and Jun Zhang. NNMF-based factorization techniques for high-accuracy privacy protection on non-negative-valued datasets. In *Proceedings of the 2006 IEEE Conference of Data Mining, International Workshop on Privacy Aspects of Data Mining*, pages 513–517. IEEE Computer Society, 2006.

[83] S. Wild, J. Curry, and A. Dougherty. Motivating non-negative matrix factorizations. In *Proceedings of the 8th SIAM Conference on Applied Linear Algebra*, Williamsburg,VA, July 15-17 2003.

[84] Yi Huang Wu, Chia Ming Chiang, and Arbee L.P. Chen. Hiding sensitive association rules with limited side effects. *IEEE Transaction on Knowledge and Data Engineering*, 19(1):29–42, 2007.

[85] W. Xu, X. Liu, and Y. Gong. Document-clustering based on non-negative matrix factorization. In *Proceedings of SIGIR'03*, pages 267–273, Toronto, CA, July 28-August 1 2003.

[86] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *In proceedings of the 5th SIAM International Conference on Data Mining*, 2005.

[87] H. Zha and H.D.Simon. On updating problems in latent semantic indexing. *SIAM J. Sci. Comput.*, 21(2):782–791, 1999.

[88] Jun Zhang, Jie Wang, and Shuting Xu. *the Encyclopedia of Data Warehousing and Mining (2nd Edition)*, chapter Matrix decomposition techniques for data privacy, pages 1183–1193. Information Science Reference, 2008.

[89] Nan Zhang, Shengquan Wang, and Wei Zhao. A new scheme on privacy-preserving data classification. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383, Chicago, IL, August 2005.

Vita

Personal Data:

Name: Jie Wang
Place of Birth: Nanjing, China
Date of Birth: October 25, 1971

Educational Background:

- Master of Engineering in Electrical Engineering, Beijing University of Chemical Technology, China, 1996.
- Bachelor of Engineering in Electrical Engineering, Nanjing University of Chemical Technology, China, 1993.

Research Interests:

- Matrix decomposition technique and its applications.
- Information security and privacy.
- Knowledge discovery and bioinformatics.

Awards and Grants:

⋆ Thaddeus B. Curtz Memorial Scholarship Award, 2008.

⋆ Kentucky Opportunity Fellowship, 2007 - 2008

⋆ University of Kentucky Commonwealth Research Award in the amount of $1000 to support the expenses to present research at the 2007 IEEE International Conference on Intelligence and Security Informatics in NewBrunswick, NJ. May 23 - 24, 2007.

⋆ Full Travel Support from Lawrence Berkeley National Laboratory for The Eight DOE Advanced CompuTational Software (ACTS) Collection Workshop: Gearing up Scientific Applications for the Petascale Computing Era. August 21 - 24, 2007. Berkeley, CA.

⋆ Student Travel Support from Graduate School Fellowship of University of Kentucky, 2007 - 2008.

⋆ Student Travel Support from Graduate School Fellowship of University of Kentucky, 2006 - 2007.

⋆ Student Travel Support from Graduate School Fellowship of University of Kentucky, 2005 - 2006

⋆ Principle investigator for Province Science Foundation Project under grant No. 02KJB630001: Modelling and Simulation of Process Mode of Collaborative E-business. Jiangsu, China. 2002 - 2004.

Research Experience:

- Research Assistant, 08/2004 - present. Laboratory for High Performance Scientific Computing & Computer Simulation, Department of Computer Science, University of Kentucky.

- Software Engineer, 09/1999 - 10/2000. Sino-American CS Software Technology Ltd., New Products Dept., Nanjing, China.

  Software testing and English technique document writing.

- Research Assistant, 10/1994 - 03/1996. System Simulation Centre, Beijing University of Chemical Technology, Beijing, China. .

  - A Training Simulator for Industrial Process, led by Prof. Chongguang Wu, 1995-1996.
  - Integrated Simulation Environment for Large-scale Power Plant, led by Prof. Chenglin Shen, 1994 -1996.

Teaching Experience:

- Guest Lecturer, 04/2005 - 05/2007. Department of Computer Science, University of Kentucky, USA.

  Course: CS689 - Computational Medical Imaging Processing.

- Assistant Professor, 09/1996 - 08/2004. Department of Automation, Nanjing University of Technology, China.

  Course:

  - Computer Network Technology.
  - Computer Architecture & Assembly Language
  - Computer Control Technology
  - Control Theory Lab
  - Specialized English for Automation

Publications:

- **Book**:

  - *Information Technology*, editor: Weijun Zhong, Ji Chi, Jie Wang and Shue Mei. Scientific and Technical Documents Publishing House. Beijing, China. 2005.

- **Book Chapter**:
  - *Matrix Decomposition-Based Data Distortion Techniques for Privacy Preservation in Data Mining*. Jun Zhang, Jie Wang and Shuting Xu, 2007.
- **Ph.D Dissertation Proposal**:
  - *Matrix Decomposition for Data Distortion and Data Mining Applications*, Jie Wang, 2007.
- **Master Thesis**:
  - *Simulation Software Development Environment for Large-Scale Chemical Process - Simulation of Flow and Pressure Network of Fluid (Chinese)*. 1996.
- **Undergraduate Thesis**:
  - *Intelligent PID Controller on Mixer Temperature (Chinese)*. 1993.
- **Refereed Journal Papers**
  - A Novel Data Distortion Approach via Selective SSVD for Privacy Protection. Jie Wang, Jun Zhang, Weijun Zhong and Shuting Xu. Special Issue of *International Journal of Information and Computer Security* on Security and Privacy Aspects of Data Mining, 2(1):48-70. 2008.
  - High Order Compact Computation and Nonuniform Grids for Streamfunction Vorticity Equations. Jie Wang, Weijun Zhong and Jun Zhang. *Journal of Applied Mathematics and Computation*, Vol.179, No.1, pp:108-120. 2006.
  - A General Meshsize Fourth-order Compact Difference Discretization Scheme for 3D Poisson Equation. Jie Wang, Weijun Zhong and Jun Zhang. *Applied Mathematics and Computation*, Vol.183, No.2, pp:804-812. 2006.
  - A Singular Value Decomposition Based Data Distortion Strategy for Privacy Protection. Shuting Xu, Jun Zhang, Dianwei Han and Jie Wang. *Knowledge and Information Systems*, Vo.10, No.3, pp:383-397. 2006.
  - Application of XML technology to Electronic Commerce Systems. Wang Jie and Lin Jinguo. *Journal of Nanjing University of Chemical Technology (Natural Science Edition) (Chinese)*, Vol.23, No.5, pp:36 - 40. 2001
  - Improved Network Simulation Algorithm and its Implementation. Wang Jie. *Journal of Nanjing University of Chemical Technology (Natural Science Edition) (Chinese)*, Vol.22, No.4, pp:39 - 42.
  - On Inter-Enterprise Electronic Procurement. Lu Qing, Wang Jie and Lin Jinguo. *Business Research (Chinese)*. October 2004. China.
  - Design and Implementation of E-Procurement System for Manufacturing Enterprises. Lu Qing, Wang Jie and Lin Jinguo. *Logistics Technology (Chinese)*, December 2003.
  - XML Technology and E-Sourcing. Hou Ping and Wang Jie. *Market Weekly (Chinese)*. August 2002.

- **Refereed Conference Papers**
  - Wavelet-based Data Perturbation for Simultaneous Privacy-preserving and Statistics-presering. Lian Liu, Jie Wang and Jun Zhang. accepted by *8th IEEE International Conference on Data Mining (ICDM 08) - Workshops, icdmw. the 2nd Workshop on Reliability Issues in Knowledge Discovery*. December 15, 2008, Pisa, Italy.
  - Towards Real-time Performance of Data Value Hiding for Frequent Data Updates by Incremental Matrix Decomposition. Jie Wang, Justin Zhan and Jun Zhang. *the 2008 IEEE International Conference on Granular Computing (GrC2008)*(ISBN: 978-1-4244-2512-9). pp:606 - 611, August 26-28, 2008. Hangzhou, China.
  - Simultaneous Data and Pattern Hiding in Unsupervised Learning. Jie Wang, Jun Zhang, Lian Liu and Dianwei Han. *7th IEEE International Conference on Data Mining - Workshops (ICDMW'07), icdmw*, pp. 729-734, Oct. 28, 2007. Omaha, NE.
  - Addressing Accuracy Issues in Privacy Preserving Data Mining through Matrix Factorization. Jie Wang and Jun Zhang. *Intelligence and Security Informatics, 2007 IEEE* pp:217-220. 23-24 May 2007. NewBrunswick, NJ.
  - NNMF-Based Factorization Techniques for High-Accuracy Privacy Protection on Non-negative-valued Datasets. Jie Wang, Weijun Zhong and Jun Zhang. *6th IEEE International Conference on Data Mining - Workshops (ICDMW'06), icdmw*, pp. 513-517, Dec.18, 2006. Hongkong, China.
  - Selective Data Distortion via Structural Partition and SSVD for Privacy Preservation. Jie Wang, Weijun Zhong, Jun Zhang and Shuting Xu. *Proceedings of the 2006 International Conference on Information & Knowledge Engineering*, pp: 114 - 120, June 26-29, 2006 CSREA Press, Las Vegas, Nevada, USA.
  - Data Distortion for Privacy Protection in a Terrorist Analysis System. Shuting Xu, Jun Zhang, Dianwei Han and Jie Wang. *Proceedings of IEEE International Conference on Intelligence and Security Informatics*, pp:459-464. ISI 2005, May 19-20, 2005, Atlanta, GA.
  - Support Vector Machine Approach for Aartner Selection of Virtual Enterprises. Jie Wang, Weijun Zhong and Jun Zhang. *Computational and Information Science 2004 Proceedings. LECTURE NOTES IN COMPUTER SCIENCE 3314*, pp:1247-1253, 2004.
  - Multiagent-Based Partner Selection of Dynamic Alliances in Inter-organizational Collaborative E-commerce. Jie Wang, Xingguo Shi and Weijun Zhong. *2004 International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES 2004) Proceedings*. pp:793, China, 2004.
  - The Application of XML Technology in E-procurement. Ping Hou, Jinguo Lin, and Jie Wang. *2002 International Symposium on Distributed Computing and Application to Business, Engineering and Science (DCABES 2002)*, China, Dec 16-20, 2002.

- **Presentations with Published Abstracts**

  - Iterative-based Matrix Factorization Techniques for High-Accuracy Privacy Protection on Non-negative-valued Datasets. Jie Wang and Jun Zhang. *Eighth IMACS International Symposium on Iterative Methods in Scientific Computation*, College Station, Texas. November 14-17, 2006.
  - SSVD for Privacy Protection - Poster Presentation, Jie Wang and Jun Zhang. *12th Annual kentucky Statewide EPSCoR Conference: Exploring shared interests with National Labs*. Louisville Marriott Downtwon Hotel, Kentucky. May 15, 2006.
  - Support Vector Machine Approach for Partner Selection of Virtual Enterprises. Jie Wang, Weijun Zhong and Jun Zhang. *The 19th Annual Symposium in Mathematical, Statistical and Computer Sciences*, Richmond, Kentucky. April 2005.

- **Technical Reports** (with Department of Computer Science, University of Kentucky, KY.)

  - Technical Report No. 487-07: Simultaneous Pattern and Data Hiding in Unsupervised Learning. Jie Wang, Lian Liu, Dianwei Han and Jun Zhang, 2007.
  - Technical Report No. 482-07: Wavelet-Based Data Distortion for Privacy-Preserving Collaborative Analysis. Lian Liu, Jie Wang, Zhenmin Lin and Jun Zhang, 2007.
  - Technical Report No. 477-07: Data Pattern Maintenance by Matrix Approximation: An Application to Information Security. Jie Wang and Jun Zhang, 2007.
  - Technical Report No. 472-07: Matrix Decomposition-Based Data Distortion Techniques for Privacy Preservation in Data Mining. Jun Zhang, Jie Wang and Shuting Xu, 2007.
  - Technical Report No. 464-06: NNMF-Based Factorization Techniques for High-Accuracy Privacy Protection on Non-negative-valued Datasets. Jie Wang and Jun Zhang, 2006.
  - Technical Report No. 460-06: Pipe-Entrance Flow Simulation Using Multigrid Finite Volume. Wensheng Shen, Jun Zhang, Jie Wang, and Fuqian Yang, 2006.
  - Technical Report No. 449-05: Selective Data Distortion via Structural Partition and SSVD for Privacy Protection. Jie Wang, Jun Zhang, Weijun Zhong and Shuting Xu, 2005.
  - Technical Report No. 432-05: Data Distortion for Privacy Protection in a Terrorist Analysis System. Shuting Xu, Jun Zhang, Dianwei Han and Jie Wang, 2005.
  - Project Report: Management Architecture of High Technology Industry(Chinese). Province Science Research Project (BR2001010), Jie Wang, Jan. 2003.