Privacy Preserving Data Mining using Unrealized Data Sets – Scope Expansion and Data Compression

by

Pui Kuen Fong
BSc, University of Victoria, 2005
MSc, University of Victoria, 2008

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Supervisory Committee

Privacy Preserving Data Mining using Unrealized Data Sets – Scope Expansion and Data Compression

by

Pui Kuen Fong
BSc, University of Victoria, 2005
MSc, University of Victoria, 2008

Supervisory Committee

Dr. Jens H. Weber, Department of Computer Science
Co-Supervisor

Dr. Alex Thomo, Department of Computer Science
Co-Supervisor

Dr. Aaron Guillver, Department of Electrical & Computer Engineering
Outside Member

Abstract

Supervisory Committee
Dr. Jens H. Weber, Department of Computer Science
Co-Supervisor
Dr. Alex Thomo, Department of Computer Science
Co-Supervisor
Dr. Aaron Guillver, Department of Electrical & Computer Engineering
Outside Member

In previous research, the author developed a novel PPDM method – Data Unrealization – that preserves both data privacy and utility of discrete-value training samples. That method transforms original samples into unrealized ones and guarantees 100% accurate decision tree mining results. This dissertation extends their research scope and achieves the following accomplishments: (1) it expands the application of Data Unrealization on other data mining algorithms, (2) it introduces data compression methods that reduce storage requirements for unrealized training samples and increase data mining performance and (3) it adds a second-level privacy protection that works perfectly with Data Unrealization.

From an application perspective, this dissertation proves that statistical information (i. e. counts, probability and information entropy) can be retrieved precisely from unrealized training samples, so that Data Unrealization is applicable for all counting-based, probability-based and entropy-based data mining models with 100% accuracy.

For data compression, this dissertation introduces a new number sequence – J-Sequence – as a mean to compress training samples through the J-Sampling process. J-Sampling converts the samples into a list of numbers with many replications. Applying

run-length encoding on the resulting list can further compress the samples into a constant storage space regardless of the sample size. In this way, the storage requirement of the sample database becomes $O(1)$ and the time complexity of a statistical database query becomes $O(1)$.

J-Sampling is used as an encryption approach to the unrealized samples already protected by Data Unrealization; meanwhile, data mining can be performed on these samples without decryption. In order to retain privacy preservation and to handle data compression internally, a column-oriented database management system is recommended to store the encrypted samples.

Table of Contents

Supervisory Committee ii
Abstract iii
Table of Contents v
List of Tables vi
List of Figures vii
Acknowledgments viii
Dedication x
Chapter 1 1
INTRODUCTION 1
1.1 Research Scope, Facts and Assumptions 1
1.2 Previous Research and the Contribution of Current Work 2
1.3 Dissertation Organization 4
Chapter 2 6
FOUNDATION OF DATA UNREALIZATION APPROACH 6
2.1 Common PPDM Approaches 6
2.2 Data Unrealization Approach 18
2.3 Generating Decision Tree from Unrealized Training Set 32
2.4 Evaluation and Limitations 39
Chapter 3 41
DATA UNREALIZATION – SCOPE OF APPLICATION 41
3.1 Data Unrealization and Set Theory 41
3.2 Counting-based Data Mining Approaches 43
3.3 Probability-based Data Mining Approaches 52
3.4 Entropy-based Data Mining Approaches 56
Chapter 4 58
JOIN INFORMATION SEQUENCE 58
4.1 Introducing J-Sequence 58
4.2 Properties of J-Sequence and the J-Sequence Representation 59
4.3 Generating J-Sequence by a Brute Force Approach 62
4.4 Creating J-Sequence with Prime Numbers 71
4.5 J-Union, J-Intersection and J-Comparison 75
Chapter 5 80
COMPRESSING UNREALIZED TRAINING DATA SETS 80
5.1 Numerating Unrealized Training Samples 80
5.2 Data Compression, Storage Complexity and Query Time Complexity 87
5.3 Column-based Database Architecture 95
5.4 Second-level Protection and Protection-Key Updates 97
Chapter 6 101
CONCLUSION AND FUTURE WORKS 101
Bibliography 105

List of Tables

Table 2-1 Samples taken from a study case. 10
Table 2-2 Sanitized data table after 3 generalization steps. 11
Table 2-3 Sanitized data table after random substitution. 13
Table 2-4 Reconstructed data sets according to attribute *Outlook* 14
Table 2-5 6 samples with attributes [*Age*, *Salary*, *Risk*]. 17
Table 2-6 Transformed data sets of samples in Table 2-6. 17
Table 2-7 A universal set T^U of data table T 22
Table 2-8 Training data sets T' returned by the function call UNREALIZING TRAINING-SET(T_S, T^U, {}, {}). 23
Table 2-9 Perturbing data sets T^P returned by the function call UNREALIZING TRAINING-SET(T_S, T^U, {}, {}). 24
Table 2-10 A universal set T^U of data table T with dummy attribute values on *Outlook* and *Wind* 29
Table 2-11 Training data sets T' with dummy attribute values on *Outlook* and *Wind* 29
Table 2-12 Perturbing data sets T^P with dummy attribute value on *Outlook* and *Wind* 32
Table 3-1 The frequent itemset table T_O after the first iteration with support threshold = 30%. 50
Table 3-2 The frequent itemset table T_O after the second iteration with support threshold = 30%. 50
Table 3-3 The resulting frequent itemset table T_O after the third iteration with support threshold = 30%. 51
Table 3-4 The resulting frequent itemset table T_O by applying the APRIOR' function with the unrealized training sets. 52
Table 5-1 A universal set J_N in Table 2-11 after mapping to J_N = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39}..... 83
Table 5-2 Training data sets T' in Table 2-12 after mapping to J_{12} = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39}..... 84
Table 5-3 Perturbing data sets T^P in Table 2-13 after mapping to J_{12} = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39}..... 87
Table 5-4 Perturbing data sets T^P in Table 5-3 after sorting. 93
Table 5-5 A sample table in the form of $< value, count >$. 97
Table 5-6 The resulting frequent itemset table T_O by applying the APRIOR' function with encrypted samples. 99

List of Figures

Figure 2-1 Domain generalization hierarchy of quasi-identifier $\{Outlook, Humidity, Wind, Play\}$ with generalization sequences $< Outlook_1, Humidity_1, Wind_1, Outlook_2, Play_1 >$. 9

Figure 2-2 Pseudocode of unrealizing training set algorithm. 21

Figure 2-3(a) Distributing data sets in qT^U by data set value. 25

Figure 2-3(b) The rectangles contain data sets of T_S. The rest are in $(T'+T^P)$. 25

Figure 2-4 Pseudocode of the decision tree learning algorithm. 35

Figure 2-5 The final decision tree built from the training set in Table 2-1. 36

Figure 2-6 Pseudocode of the modified decision tree learning algorithm using T' and T^P. 38

Figure 2-7 The final decision tree built from data sets in Table 2-12 and 2-13. 39

Figure 3-1(a) The illustration of space U and two compliment subsets in U : A and A'. 43

Figure 3-1(b) The illustration of space U as qT^U and two compliment subsets in the space: A as T_S and A' as $(T'+T^P)$. 43

Figure 3-2 Pseudocode of the classic Aprior algorithm. 47

Figure 3-3 Pseudocode of the modified Aprior algorithm applied for unrealized training samples. 48

Figure 3-4 Pseudocode of the modified Aprior algorithm applied for unrealized training samples with optimization. 49

Figure 4-1 Neighborhood Graph of four objects: object1, object2, object3 and object4 with (a) bitmap representation and (b) J-Sequence representation $J_4 = \{5, 6, 7, 8\}$, $N = 4$ and $\log(P_4) = 3.2253$. 62

Figure 4-2 Pseudocode of the algorithm that generates a sorted J_N. 68

Figure 4-3 Pseudocode of the GENERATE-NEXT-J-SEQUENCE algorithm. 69

Figure 4-4 Pseudocode of the GET-NEXT-NUMBER algorithm. 69

Figure 4-5 Pseudocode of the VERIFY-MODULUS-RULE algorithm. 70

Figure 4-6 Pseudocode of the VERIFY-PRODUCT-RULE algorithm. 70

Figure 4-7 Pseudocode of the VERIFY-SIZE-RULE algorithm. 71

Figure 4-8 Illustration of a province with ten cities and four selections of first server location. The server coverage scopes from spot A, B, C and D are shown as the circles centered by the spots. 77

Figure 5-1 Pseudocode of the PROTECTION-KEY-UPDATE algorithm. 100

Acknowledgments

I would like to extend thanks and appreciation to my supervisors, Dr. Jens H. Weber and Dr. Alex Thomo, who have provided the best support towards my study.

Dr. Weber has been my supervisor since my Master study and he gives me all of his trust. When I began my career outside of the city in 2006, he made arrangements to afford me with academic support remotely. Distance did not affect his trust, and he offered me a position as his Ph.D. student shortly after my Master defence. From all these years, he has always provided me with excellent guidance and knowledge, all with patience and respect.

I see Dr. Thomo as my mentor as he guides me in finding the fun of study. When I first took in his AI course during my Bachelor study, he opened my mind and I saw the beautiful part of those difficult subjects. Then I followed him to learn database system, data mining, and machine learning from other courses that he taught. After all these studies, I found my lifetime commitment and acquired the energy to further pursue my research.

Finally, I would like to share the honour of my Ph.D. title with my wife, my lover and my best friend, Jessica Zhao, for her selfless support and encouragement. She treats my dream more valuable than her happiness and contributes to every success that I had approached / will approach. I initially found Information Sequence (in short as I-Sequence) that works up to size 11 with some broken theories in 2003. As I have my Ms. J. standing by me with dedication, this sequence eventually becomes Join Information Sequence (in short as J-Sequence, as the research covered by Chapter 4 of this dissertation) with all the solid proofs of J-Product, J-Size, J-Comparison, J-Intersection and J-Union in 2012 and it further supports all my work of J-Samples and J-Sampling (covered by Chapter 5 of this dissertation). For someone who references any J-Concepts,

please put the following QR codes ( for Chapter 4 and  for Chapter 5) in his / her paper.

Dedication

This dissertation is dedicated to my grandmother, Kwai Lan Choi[1] (1910-2007), who brought me up, loved me, and inspired me positively all the time. She saw my value as the extension of her life and she passed this belief to his son, who is my father, by setting up the model.

[1] My grandmother's name is originally in Chinese as 蔡桂蘭.

Chapter 1

INTRODUCTION

Nowadays, security of user information is a major concern for Internet based businesses, such as search engines, social networking, online banking and online shopping. On one hand, these companies can mine data to extract profitable information (such as consumer behaviour, users' usage pattern and their business focuses) through the data collected from the users (referred to as sample data sets, training data sets, training sets or samples hereafter). On the other hand, the user information, which contains sensitive data from individuals, attracts internet threats from hackers. Therefore, privacy preserving data mining (in short as PPDM) comes as a popular research area that "develop[s] algorithms for modifying the original data in some way, so that the private data and private knowledge remain private even after the mining process[1]". At the same time, these algorithms attempt to maintain the most utilities of the samples for data mining purpose. In most cases, there are some inevitable trade-offs between data privacy protection and data mining utility.

1.1 Research Scope, Facts and Assumptions

According to a security and risk management report[2], more than 70% of the worst data security breaches of the 21st century were caused by injection or hacking from untrustworthy parties. Although, hackers can steal only a small portion (less than 15% in the worst case scenario) of the samples stored by a large internet corporation through an

internet security hole, that small amount of information may contain millions pieces of sensitive data of some individuals[3]; therefore, these privacy and security threats have concerned public concerns at both the national and international level. As a result, those web giants have started to invest large amount of resources in the methods for data privacy safeguarding[4] and PPDM research.

Based on the facts given above, the scope of this paper focuses on privacy threats through online benching from unauthorized parties on large databases / data storage for data mining with the following assumptions:

(1) The breach is sourced from a client application / interface attack, which cannot access raw data in the physical storage directly beyond the usability of the application / interface itself.

(2) The sample size of the collected samples is very large.

(3) Even though the total number of stolen records can be large, they are considered to be a small fraction of the overall stored records when compared with the sample size.

(4) The possibility of a successful injection or hacking is low and finding the connection amongst the results from multiple attacks could be difficult; therefore, the cases of multiple attacks are ignored in this paper.

1.2 Previous Research and the Contribution of Current Work

In *Privacy Preserving Decision Tree Learning Using Unrealized Data Sets*[5], Fong and Weber introduced a novel privacy preserving approach (named Data Unrealization) for discretized data sets. This approach “converts the original data sets

into a group of unreal data sets" for privacy protection; meanwhile, the authors provided a decision tree mining algorithm that can retrieve an accurate decision tree "built directly from those unreal data sets." This research offers a foundation for PPDM without sacrificing the quality of the data mining results, which is the downside of many PPDM approaches. The authors limited their research scope to ID3 decision tree mining and later Williams expanded the scope to C4.5 decision tree mining[32]; however, the theory of Data Unrealization has not yet been testified on other classes of data mining methods.

The Data Unrealization algorithm takes the original training samples as input and outputs two sets of unrealized data sets (referred to as unrealized samples). The sample distribution of the unrealized samples is related to that of the original samples – the more likely a sample exists in the unrealized samples, the less likely it can be found in the original; therefore, any piece of data loss in the unrealized samples has a low possibility to match the information of the original. The theory still works even if we create some data sets has zero frequency in the original (which have the highest frequency in the unreal samples), so that we can decrease the risk of privacy loss of the original from the unrealized data sets further; however, adding those dummy data sets might raise a storage requirement concern.

This dissertation is the continuity of Fong and Weber's research. We are going to (1) extend the scope coverage for PPDM on discretized data sets, (2) resolve the storage concern of the previous work and (3) promote Data Unrealization to a higher privacy preservation level. From the scope coverage perspective, we will expand the usage of the Data Unrealization approach to all counting-based, possibility-based and entropy-based data mining algorithms, which include many common data mining models on discretized

data sets. From the storage requirement perspective, we will introduce a number sequence, named J-Sequence that can be applied to compress the sample storage size while improving query performance at the same time.

J-Sequence can be applied to support efficient and straightforward computations on the following terms: (1) the existence of a member from any product (named J-Product) of some members of that J-Sequence, (2) the number of members contained in that J-Product, (3) the order of multiple J-Products in terms of their number of containing members, (4) the union set of all members contained in multiple J-Products and (5) all common members of multiple J-Products. If we encrypt the unrealized samples by using a J-Sequence and apply run-length encoding compression on the encrypted samples, then the above properties provide the ground for $O(1)$ sample storage requirement and $O(1)$ statistical query performance. In addition, the encryption method itself is an extra privacy shelter of the samples.

1.3 Dissertation Organization

This dissertation consists of seven chapters. Chapter 1 introduces background, motivation and contribution of our research, so that readers can follow the overall scope and presentation of the research content. Chapter 2 explains the details of the previous research from Fong and Weber. Their work establishes the conceptual ground of the Data Unrealization approach that will be extended in later chapters. Chapter 3 describes some statistical theories based on Data Unrealization and applies them to counting-based, possibility-based and entropy-based data mining applications with examples. Chapter 4 introduces J-Sequence in general and explains its concepts. Chapter 5 applies J-

Sequence's contributions to our current studies. We use J-Sequence to compress the storage of the unrealized data sets so that the storage requirement is reduced, the query performance is increased and the privacy protection is enhanced. Chapter 6 provides an overall summary of this dissertation, and suggests directions for future research on this topic.

Chapter 2

FOUNDATION OF DATA UNREALIZATION APPROACH

In *Privacy Preserving Decision Tree Learning Using Unrealized Data Sets*[5] published in 2012, Fong and Weber introduced a novel PPDM approach, Data Unrealization, that preserves both the privacy and utility of the training samples. In this chapter, we will first discuss the motivation of authors by exploring some common PPDM techniques and their standpoints. After that, we will focus on the technical details of the Data Unrealization approach.

All terms, concepts, theories and notations covered in this chapter will be carried through in the rest of this dissertation. Please refer to the original paper if needed as we will skip the details of proofs in this chapter. As our discussion always involves data tables containing samples, let's define the sample table: a sample table $T = \{t_1, t_2, \ldots, t_n\}$ is a table containing samples associating with a set of attributes $A = \{a_1, a_2, \ldots, a_m\}$ where each t_i is a tuple of attribute values $< k_1, k_2, \ldots, k_m >$, which means $\{a_1 = k_i, a_2 = k_2, \ldots, a_m = k_m\}$.

2.1 Common PPDM Approaches

In *Privacy Preserving Data Mining: Models and Algorithms*[6], Aggarwal and Yu classify PPDM techniques, including data modification, cryptographic, statistical, query auditing and perturbation-based strategies. Statistical and query auditing techniques (such as random sampling[34, 35]) are related to inference control and security assurance, all of

which are subjects outside of the focus of our studies. Therefore, we will only explore data modification, perturbation-based and cryptographic approaches in this chapter.

2.2.1 Data Modification Approaches and *k*-anonymity

Data modification techniques maintain privacy by modifying attribute values of the sample data sets. Essentially, data sets are modified by eliminating or unifying uncommon elements among all data sets, such that each data set within the sanitized samples is guaranteed to pass the threshold of similarity with the other data sets. These similar data sets act as masks for the others within the group, because they cannot be distinguished from the others. In this way, privacy can be preserved by ensuring that every data set is loosely linked with a certain number of original data sets.

k-anonymity[7, 8, 9] is a typical data modification approach that intends to achieve effective data privacy preservation. The term "*k*-anonymity" implies that the quasi-identifier of each sanitized data set is the same as those of at least $(k+1)$ others. A quasi-identifier is defined as a set of attributes that can be used to identify an individual with a significant probability of accuracy. If each quasi-identifier is contained by at least k individuals, then the individuals cannot be distinguished from each other using this quasi-identifier. To achieve *k*-anonymity, suppression or aggregation techniques are used to "generalize" attribute values of data sets. After the generalization process, the domains of attributes are shrunk as attribute values are merged into groups.

Let's take Table 2-1[2] and the domain generalization hierarchy shown in Figure 2-1 as an example. For approaching 2-anonymity of all sample data sets, three generalization steps are needed to form the sanitized data table is shown in Table 2-2. The sanitized data sets guarantee that all sensitive information from the original will be hidden – but with loss of information of the generalized attributes. In this example, data utility is compromised by the removal of attributes {*Humidity*,*Wind*} from the original data, because it will lead to a significant loss of accuracy from the data mining result based on the sanitized data table.

The utility of the sanitized data table could be improved by using another domain generalization hierarchy, or even by applying another generalization rule. However, the *k*-anonymity strategy presents two potential problems: firstly, the privacy preservation and information usability factors are heavily dependent upon the selection of the number of anonymity, quasi-identifier and generalization rules, which make it NP-hard to find an optimal solution; secondly, no matter how good the generalization rule is, each generalization step downgrades the utility of the generalized attributes and the domain of the data sets, in addition to the codomain of the data mining result, will be different from those of the original data sets. Even although researchers promoted the studies of *k*-anonymity to *l*-diversity and *t*-closeness[33] in recent years, the problems mentioned in this section have not yet been solved.

[2] Please aware that the field Sample# in the table is added for reading convenient without actual usage. In later chapters, we may remove this field if needed.

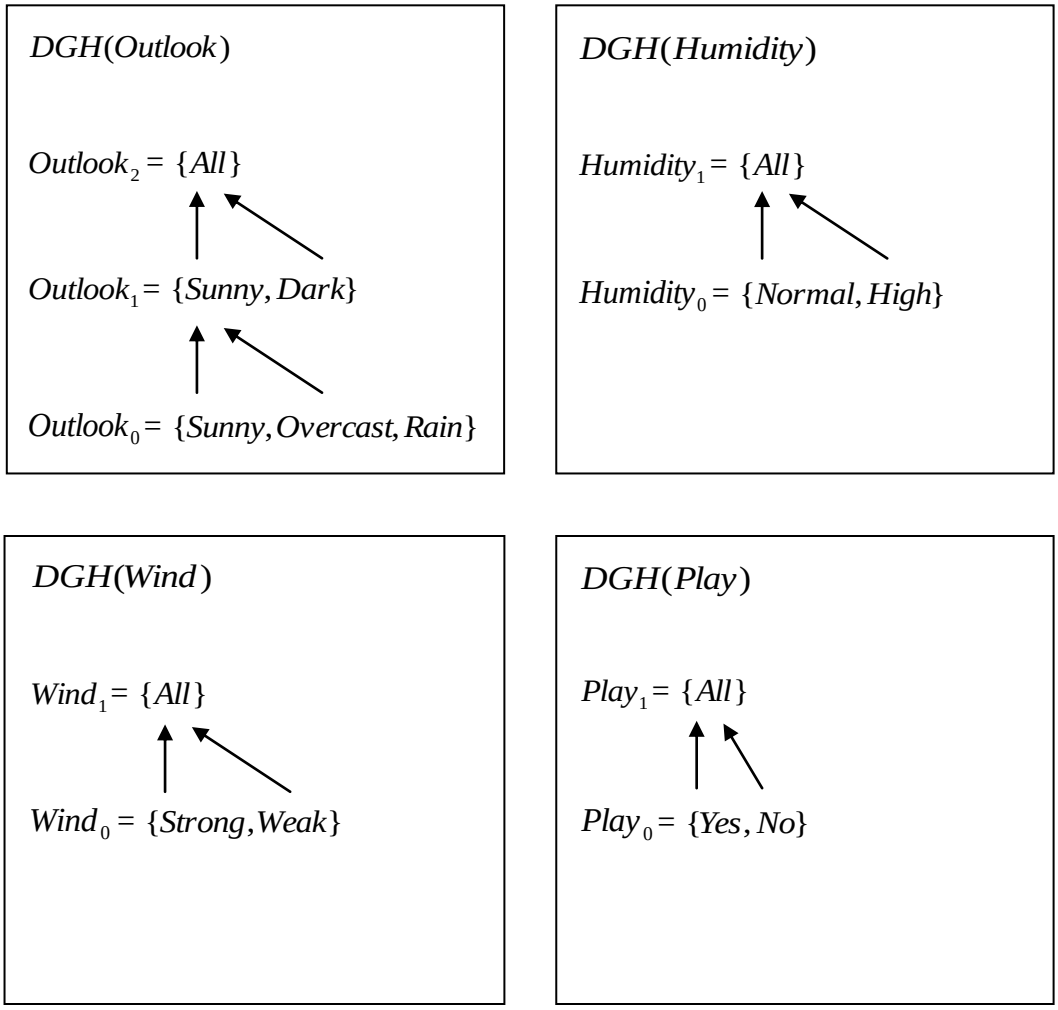

Figure 2-1 Domain generalization hierarchy of quasi-identifier $\{Outlook, Humidity, Wind, Play\}$ with generalization sequences $< Outlook_1, Humidity_1, Wind_1, Outlook_2, Play_1 >$.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	Yes
4	Rain	High	Weak	Yes
5	Rain	Normal	Weak	Yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	Yes
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	Yes
10	Rain	Normal	Weak	Yes
11	Sunny	Normal	Strong	Yes
12	Overcast	High	Strong	Yes
13	Overcast	Normal	Weak	Yes
14	Rain	High	Strong	No

Table 2-1 Samples taken from a study case.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	All	All	No
2	Sunny	All	All	No
3	Dark	All	All	Yes
4	Dark	All	All	Yes
5	Dark	All	All	Yes
6	Dark	All	All	No

7	Dark	All	All	Yes
8	Sunny	All	All	No
9	Sunny	All	All	Yes
10	Dark	All	All	Yes
11	Sunny	All	All	Yes
12	Dark	All	All	Yes
13	Dark	All	All	Yes
14	Dark	All	All	No

Table 2-2 Sanitized data table after 3 generalization steps.

2.2.2 Perturbation-based Approaches and Random Substitution

Perturbation-based approaches[10] attempt to achieve privacy protection by distorting the original data. By applying some data perturbation techniques, data sets are modified such that they are different from the originals. Meanwhile, the perturbed data sets still retain features of the originals, so that records derived from them can be used to perform data mining, directly or indirectly, via data reconstruction. Two common strategies for data perturbation are noise-adding and random-substitution[11]. Noise-adding adds noise v to each sample t such that the perturbed data set, which equals $(t+v)$, is similar but not equal to the original one where v is a random value within an acceptable range. This strategy is usually used for numeric values and has been proven to preserve little data privacy[10] because so it is not discussed in this dissertation.

Instead of adding noise, random substitution perturbs samples by randomly replacing values of attributes. For example, if the possible values $\{Sunny, Overcast, Rain\}$ of the attribute *Outlook* take the substitution rule as $\{Sunny \rightarrow Rain, Overcast \rightarrow Sunny, Rain \rightarrow Rain\}$, then data sets $< Sunny, Normal, Weak, Yes >$ and $< Overcast, Normal, Strong, No >$ will be replaced by $< Rain, Normal, Weak, Yes >$ and $< Sunny, Normal, Strong, No >$ respectively. Random substitution is attribute-based and substitution rule of each attribute is computed from a perturbation matrix and a number controlling the degree of privacy protection. Table 2-3 shows a possible outcome of perturbed data sets generated by random substitution.

After random substitution, the information related to a particular attribute in the perturbation data sets is irrelevant to that of the original samples. For data mining, the perturbation data sets must undergo data set reconstruction. The reconstructed data sets are an estimation of the originals, based on the reconstruction matrix calculated from the perturbation matrix. The reconstruction process is also attribute-based and it requires the perturbed data sets T' and the reconstruction matrix. By reconstructing the data sets in Table 2-3, we may produce the reconstructed data sets as shown in Table 2-4.

From the reconstructed data sets, we keep the domain as that of the original's. Therefore, the utility of the data mining results is better than the ones from *k*-anonymity because the codomain of the results, which follows the domain, remains the same as the original. However, the number used as the degree of privacy protection is also the degree of accuracy loss. As a result, choosing the value of this number is a difficult problem in a practical situation because we always lose some important factors of PPDM – privacy, accuracy or both.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	Yes
4	Overcast	High	Weak	Yes
5	Overcast	Normal	Weak	Yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	Yes
8	Sunny	High	Weak	No
9	Overcast	Normal	Weak	Yes
10	Rain	Normal	Weak	Yes
11	Overcast	Normal	Strong	Yes
12	Rain	High	Strong	Yes
13	Rain	Normal	Weak	Yes
14	Rain	High	Strong	No

Table 2-3 Sanitized data table after random substitution.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Weak	No
2	Overcast	High	Strong	No
3	Overcast	High	Weak	No
4	Overcast	High	Weak	Yes

5	Overcast	High	Weak	Yes
6	Overcast	Normal	Weak	Yes
7	Overcast	Normal	Strong	Yes
8	Overcast	Normal	Weak	Yes
9	Overcast	Normal	Strong	Yes
10	Rain	Normal	Strong	No
11	Rain	Normal	Weak	Yes
12	Rain	High	Strong	Yes
13	Rain	Normal	Weak	Yes
14	Rain	High	Strong	No

Table 2-4 Reconstructed data sets according to attribute *Outlook* .

2.2.3 Cryptographic Approaches and Monotone / Anti-monotone Framework

Cryptographic approaches mask the information of sample data sets by applying encryption functions with encryption keys (or encoding functions) to them such that the sample data sets are not meaningful (or are not related to the information providers) without the corresponding decryption keys and decryption functions (or decoding functions.) Cryptographic approaches are commonly applied to multi-party protocol scenarios[36, 37]; yet, there are some approaches in this class (such as (anti)monotonic function encoding[12], homomorphic encryption[38] and asymmetric encryption[39]) that can be used in single party cases , which is the focus of our studies. In this section, we will use (Anti)monotone Framework[12] as an example for reviewing cryptographic approaches from PPDM perspective.

(Anti)monotone Framework is designed for samples with numeric-value attributes. Breakpoints are introduced to break up the sample data sets into subgroups and an (anti)monotone function is assigned to each group. A series of (anti)monotone functions (also known as transformation functions) are applied to sanitize an attribute of the samples. The choices of breakpoints and encoding functions should satisfy the global-(anti)monotone invariant constraint. To define breakpoints and transformation functions that fulfill the global-(anti)monotone invariant constraint, samples are sorted according to the values of a particular attribute for sanitization. Breakpoints are defined as the average attribute values of each pair of adjacent samples according to the attribute used as the decision of the data mining results. Based on those subgroups derived from the breakpoints, a family of bijective functions that follow the constraint could be defined arbitrarily[3]. Let's take the samples in Table 2-5, which are sorted by attribute *Age*, to define breakpoints regarding decision attribute *Risk*, then the sample set will break down into subgroups δ_1 = {Sample#1, Sample#2, Sample#3}, δ_2 = {Sample#4}, δ_3 = {Sample#5} and δ_4 = {Sample#6}, as the breakpoints are 27.5, 37.5 and 55.5. If we assign transformation functions f_1: $Age = x + 5$ if $x < 27.5$, f_2: $Age = 1.5*x$ if $27.5 < x < 37.5$, f_3: $Age = 2*x + 3$ if $37.5 < x < 55.5$ and f_4: $Age = 2.5*x - 20$ if $55.5 < x$ to δ_1, δ_2, δ_3 and δ_4 respectively, then the samples will be sanitized as the data sets in Table 2-7, which satisfy the global-monotone invariant constraint.

The global-(anti)monotone invariant constraint promises precise outcomes by the following three factors: first, one and only one inverse function exists to recover each subgroup of data sets sanitized by a transformation function. For example, f_1^{-1}: $Age = y -$

[3] The original literature does not explain the selection of transformation functions in full details, but it provides permutation and polynomial functions as possible selections.

5 if $y < 32.5$, f_2^{-1}: $Age = y/1.5$ if $41.25 < y < 56.25$, f_3^{-1}: $Age = (y - 3)/2$ if $78 < y < 114$ and f_4^{-1}: $Age = (y + 20)/2.5$ if $118.75 < y$ are the inverse functions[4] to recover data sets of subgroups δ_1, δ_2, δ_3 and δ_4 in Table 2-6; second, the composition of data mining results remains the same after transformation, which means the original decision can be reconstructed by applying the inverse functions to the data mining results[5] according to the range of breakpoints; and third, the transformation and recovery process of each attribute are independent to the others, such that the assignment of transformation and inverse functions of each attribute preserves the conservation of the recovered data mining results.

Even though the application of (anti)monotone functions saves both the privacy and utility of the samples, it raises other security issues. The transformation functions are specifically assigned to preserve the data privacy and their unique inverse functions are the keys to preserve the data utility. Therefore, the inverse functions should be stored permanently to "decode" the data mining results, or the transformation functions should be kept to determine their inverse functions. Either way, it is possible for the privacy attackers to "crack" a subgroup of original data sets by "stealing" one of the stored functions. Furthermore, (anti)monotone functions are applicable for ranged-valued attributes only, and the original literature does not provide any solution for handling discrete-valued or symbolic-valued attributes such as *Gender* = <Male, Female>. We may enumerate any symbolic-valued attribute into numeric-valued attribute, such as changing *Gender* = <Male, Female> to *Gender* = <0, 1>. From the dimension of a particular discrete-valued attribute, transformed data sets having the same attribute value

[4] f^{-1} is denoted as the inverse function of f.

[5] From the original paper, it is focusing on decision tree mining.

belongs to the same subgroup, which implies they have the same original value. Therefore, for discrete-valued or symbolic-valued attributes, the effectiveness of privacy preservation by using (anti)monotone functions is doubted.

Sample#	Age	Salary	Risk
1	17	30k	High
2	20	20k	High
3	23	50k	High
4	32	70k	Low
5	43	40k	High
6	68	50k	Low

Table 2-5 6 samples with attributes [*Age*, *Salary*, *Risk*].

Sample#	Age	Salary	Risk
1	22	30k	High
2	25	20k	High
3	28	50k	High
4	48	70k	Low
5	89	40k	High
6	150	50k	Low

Table 2-6 Transformed data sets of samples in Table 2-6.

2.2 Data Unrealization Approach

Data Unrealization approach substitutes the original samples with a set of unreal data sets. Each unreal data set does not have any direct connection with any individual original data set. This approach is introduced to overcome the drawbacks of the traditional approaches mentioned in Section 2.1. First, it preserves both the privacy and utility of the training data sets – which means we don't need to trade the degree of privacy protection with that of data mining accuracy. Second, the safety of the protected data is not dependent on the security of other means, such as the decryption / decoding functions of a cryptographic approach. Third, this approach is designed for data mining on discrete-value samples, which is the focus of our research. In this section, we will explore the details of this approach.

2.2.1 Set Notations used by Data Unrealization

Data Unrealization extends many concepts from the Set[15] and Mutliset Theories[31]; therefore, this section explains some common set notations used by this approach before we dive into the research details. These set notations are given as:

1) A universal set (T^U) is the sample domain that contains a single instance of all possible data sets in data table *T*. If a sample is any tuple of attributes $< Wind, Play >$ with attribute values $Wind = \{Strong, Weak\}$ and $Play = \{Yes, No\}$, then $T^U = \{< Strong, Yes >, < Strong, No >, < Weak, Yes >, < Weak, No >\}$.

2) q-multiple-of T_D (qT_D) is a set of data sets containing q instances of each data set in T_D where T_D is a subset of T and q is a positive integer. If T_D = $\{<Weak, Yes>\}$, then $2T_D$ = $\{<Weak, Yes>, <Weak, Yes>\}$.

3) An absolute complement of T_D ($T_D{}^C$) equals $T^U - T_D$. If T_D = $\{<Weak, Yes>\}$, then $T_D{}^C$ = $\{<Strong, Yes>, <Strong, No>, <Weak, No>\}$.

4) A q-absolute-complement of T_D ($qT_D{}^C$) equals $qT^U - T_D$. If T_D = $\{<Weak, Yes>\}$, then $2T_D{}^C$ = $\{<Strong, Yes>, <Strong, No>, <Weak, Yes>, <Weak, No>, <Strong, Yes>, <Strong, No>, <Weak, No>\}$.

5) $T_{[t]}$ denotes the subset of T that contains t where t is a tuple of attributes with values. If t = $<Wind = Weak>$, then $T^U{}_{[t]}$ = $\{<Weak, Yes>, <Weak, No>\}$.

2.2.2 Unrealizing Training Data Set Algorithm

Traditionally, a training set T is constructed by inserting sample data sets into a data table. However, the Data Unrealization approach requires an extra data table T^P. T^P is a perturbing set that generates unreal data sets for converting the sample data sets as an unrealized training set T'. The pseudocode for unrealizing the training set is shown on Figure 2-2. To unrealize the samples, we initialize both T' and T^P as empty sets, i.e. UNREALIZING TRAINING-SET(T_S, T^U, {}, {}) is called.

The recursive function UNREALIZING TRAINING-SET takes one data set in T_S in a recursion without any special requirement; it then updates T^P and T' correspondent with the next recursion. Therefore, it is obvious that the unrealized

training set process can be executed at any point during the sample collection process. Let's take the samples in Table 2-1 as T_S, and the universal set T^U in Table 2-7, as the inputs of UNREALIZING TRAINING-SET(T_S, T^U, {}, {}). The function will return T' and T^P shown in Tables 2-8 and 2-9.

The principle concept of the Data Unrealization approach is collecting data sets that are not in T_S. If we examine the details of the function UNREALIZING TRAINING-SET, we can easily find that T_S + (T'+T^P) equals qT^U for a positive integer q. If $T^U = \{t_1, t_2, \ldots, t_n\}$ and $n = |T^U|$ where $t_i \in T^U$ and $t_j \in T^U$ are tuples of attribute values as $t_i \neq t_j$, then we can represent all data sets in T^U as Figure 2-3(a). Since T_S and (T'+T^P) are subsets of qT^U, if T_S can be represented as the data sets contained in rectangles shown in Figure 2-3(b), then data sets in (T'+T^P) are the ones not contained by the rectangles.

function UNREALIZING TRAINING-SET(T_S, T^U, T', T^P) **returns** <T', T^P>

inputs: T_S, a set of input sample data sets

T^U, a universal set

T', a set of output training data sets

T^P, a set of unreal data sets

if T_S is empty **then**

return <T', T^P>

$t_i \leftarrow$ a data set in T_S

if t_i is an element of T^P **and** $T^P \setminus \{t_i\}$ is not empty **then**

$T^P \leftarrow T^P - \{t_i\}$

else

$T^P \leftarrow T^P + T^U - \{t_i\}$

$t'_i \leftarrow$ the most frequent data set in T^P

return UNREALIZING TRAINING-SET($T_S - \{t_i\}$, T^U, $T' + \{t'_i\}$, $T^P - \{t'_i\}$)

Figure 2-2 Pseudocode of unrealizing training set algorithm.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Strong	Yes
2	Sunny	High	Strong	No
3	Sunny	High	Weak	Yes
4	Sunny	High	Weak	No
5	Sunny	Normal	Strong	Yes
6	Sunny	Normal	Strong	No
7	Sunny	Normal	Weak	Yes
8	Sunny	Normal	Weak	No
9	Overcast	High	Strong	Yes

10	Overcast	High	Strong	No
11	Overcast	High	Weak	Yes
12	Overcast	High	Weak	No
13	Overcast	Normal	Strong	Yes
14	Overcast	Normal	Strong	No
15	Overcast	Normal	Weak	Yes
16	Overcast	Normal	Weak	No
17	Rain	High	Strong	Yes
18	Rain	High	Strong	No
19	Rain	High	Weak	Yes
20	Rain	High	Weak	No
21	Rain	Normal	Strong	Yes
22	Rain	Normal	Strong	No
23	Rain	Normal	Weak	Yes
24	Rain	Normal	Weak	No

Table 2-7 A universal set T^U of data table T.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Strong	Yes
2	Sunny	High	Weak	Yes
3	Sunny	Normal	Strong	Yes
4	Sunny	Normal	Strong	No
5	Sunny	Normal	Weak	Yes
6	Sunny	Normal	Weak	No

7	Overcast	High	Strong	Yes
8	Overcast	High	Strong	No
9	Overcast	High	Weak	No
10	Overcast	Normal	Strong	No
11	Overcast	Normal	Weak	Yes
12	Overcast	Normal	Weak	No
13	Rain	High	Strong	Yes
14	Rain	High	Weak	No

Table 2-8 Training data sets T' returned by the function call UNREALIZING TRAINING-SET(T_S, T^U, {}, {}).

Sample#	Outlook	Humidity	Wind	Play
21	Rain	Normal	Strong	Yes
24	Rain	Normal	Weak	No
1	Sunny	High	Strong	Yes
2	Sunny	High	Strong	No
3	Sunny	High	Weak	Yes
6	Sunny	Normal	Strong	No
8	Sunny	Normal	Weak	No
11	Overcast	High	Strong	No
13	Overcast	High	Weak	Yes
19	Overcast	High	Weak	No
22	Overcast	Normal	Strong	Yes

23	Overcast	Normal	Strong	No
26	Overcast	Normal	Weak	No
27	Rain	High	Strong	Yes
28	Rain	High	Strong	No
29	Rain	High	Weak	Yes
30	Rain	High	Weak	No
31	Rain	Normal	Strong	Yes
32	Rain	Normal	Strong	No
34	Rain	Normal	Weak	No

Table 2-9 Perturbing data sets T^P returned by the function call UNREALIZING TRAINING-SET(T_S, T^U, {}, {}).

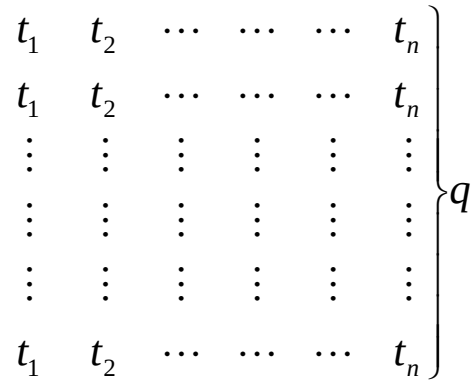

Figure 2-3(a) Distributing data sets in qT^U by data set value.

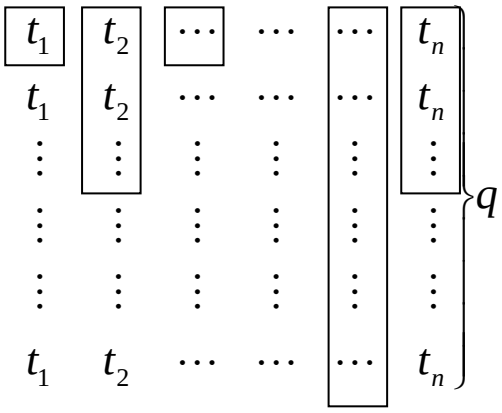

Figure 2-3(b) The rectangles contain data sets of T_S. The rest are in $(T'+T^P)$.

2.2.3 Data Set Reconstruction

Even each data set in the unrealized training set T' and perturbing set T^P is not related to any individual data set in the original sample data sets T_S, we can reconstruct the original sample data sets T_S from T' and T^P because Fong and Weber proved the following lemmas:

Lemma 2.1: $|T_S| = |T'|$

Lemma 2.2: $T_S = \frac{2*|T'|+|T^P|}{|T^U|} * T^U - T' - T^P$

The reconstruction process is dependent upon the full information of T' and T^P. As we claim from the scope of this paper, "the total number of stolen records can be large, [but] they are considered as a small fraction of the overall stored records when compared with the sample size", reconstruction of parts of T_S based on parts of T' and T^P is not possible.

2.2.4 Create Dummy Attribute Values

To increase the degree of privacy protection, dummy attribute values can be added to the domain of any attribute, for example, we can expand the possible values of the attribute *Wind* from $\{Strong, Weak\}$ to $\{Strong, Weak, Dummy\}$ where *Dummy* is a dummy attribute value that is not selectable by anyone during the data collection process. In Fong and Weber's research, they suggest to expand the domain of a sample data set to double the size of the universal set for obtaining good privacy protection outcomes. Table 2-10 to Table 2-12 shows the resulting tables of T^U, T^P and T^P after we double the size of the universal set by adding dummy attribute values on *Outlook* and *Wind* and then unrealizing the samples in Table 2-1.

Sample#	Outlook	Humidity	Wind	Play

1	Dummy1	High	Dummy2	Yes
2	Dummy1	High	Dummy2	No
3	Dummy1	High	Weak	Yes
4	Dummy1	High	Weak	No
5	Dummy1	High	Strong	Yes
6	Dummy1	High	Strong	No
7	Dummy1	Normal	Dummy2	Yes
8	Dummy1	Normal	Dummy2	No
9	Dummy1	Normal	Weak	Yes
10	Dummy1	Normal	Weak	No
11	Dummy1	Normal	Strong	Yes
12	Dummy1	Normal	Strong	No
13	Sunny	High	Dummy2	Yes
14	Sunny	High	Dummy2	No
15	Sunny	High	Weak	Yes
16	Sunny	High	Weak	No
17	Sunny	High	Strong	Yes
18	Sunny	High	Strong	No
19	Sunny	Normal	Dummy2	Yes
20	Sunny	Normal	Dummy2	No
21	Sunny	Normal	Weak	Yes
22	Sunny	Normal	Weak	No
23	Sunny	Normal	Strong	Yes
24	Sunny	Normal	Strong	No

25	Overcast	High	Dummy2	Yes
26	Overcast	High	Dummy2	No
27	Overcast	High	Weak	Yes
28	Overcast	High	Weak	No
29	Overcast	High	Strong	Yes
30	Overcast	High	Strong	No
31	Overcast	Normal	Dummy2	Yes
32	Overcast	Normal	Dummy2	No
33	Overcast	Normal	Weak	Yes
34	Overcast	Normal	Weak	No
35	Overcast	Normal	Strong	Yes
36	Overcast	Normal	Strong	No
37	Rain	High	Dummy2	Yes
38	Rain	High	Dummy2	No
39	Rain	High	Weak	Yes
40	Rain	High	Weak	No
41	Rain	High	Strong	Yes
42	Rain	High	Strong	No
43	Rain	Normal	Dummy2	Yes
44	Rain	Normal	Dummy2	No
45	Rain	Normal	Weak	Yes
46	Rain	Normal	Weak	No
47	Rain	Normal	Strong	Yes
48	Rain	Normal	Strong	No

Table 2-10 A universal set T^U of data table T with dummy attribute values on *Outlook* and *Wind* .

Sample#	Outlook	Humidity	Wind	Play
1	Dummy1	High	Dummy2	Yes
2	Dummy1	High	Dummy2	No
3	Dummy1	High	Weak	Yes
4	Dummy1	High	Weak	No
5	Dummy1	High	Strong	Yes
6	Dummy1	High	Strong	No
7	Dummy1	Normal	Dummy2	Yes
8	Dummy1	Normal	Dummy2	No
9	Dummy1	Normal	Weak	Yes
10	Dummy1	Normal	Weak	No
11	Dummy1	Normal	Strong	Yes
12	Dummy1	Normal	Strong	No
13	Sunny	High	Dummy2	Yes
14	Sunny	High	Dummy2	No

Table 2-11 Training data sets T' with dummy attribute values on *Outlook* and *Wind* .

Sample#	Outlook	Humidity	Wind	Play
15	Sunny	High	Weak	Yes
17	Sunny	High	Strong	Yes
19	Sunny	Normal	Dummy2	Yes

20	Sunny	Normal	Dummy2	No
22	Sunny	Normal	Weak	No
24	Sunny	Normal	Strong	No
25	Overcast	High	Dummy2	Yes
26	Overcast	High	Dummy2	No
28	Overcast	High	Weak	No
30	Overcast	High	Strong	No
31	Overcast	Normal	Dummy2	Yes
32	Overcast	Normal	Dummy2	No
34	Overcast	Normal	Weak	No
36	Overcast	Normal	Strong	No
37	Rain	High	Dummy2	Yes
38	Rain	High	Dummy2	No
40	Rain	High	Weak	No
41	Rain	High	Strong	Yes
43	Rain	Normal	Dummy2	Yes
44	Rain	Normal	Dummy2	No
46	Rain	Normal	Weak	No
47	Rain	Normal	Strong	Yes
1	Dummy1	High	Dummy2	Yes
2	Dummy1	High	Dummy2	No
3	Dummy1	High	Weak	Yes
4	Dummy1	High	Weak	No
5	Dummy1	High	Strong	Yes

6	Dummy1	High	Strong	No
7	Dummy1	Normal	Dummy2	Yes
8	Dummy1	Normal	Dummy2	No
9	Dummy1	Normal	Weak	Yes
10	Dummy1	Normal	Weak	No
11	Dummy1	Normal	Strong	Yes
12	Dummy1	Normal	Strong	No
13	Sunny	High	Dummy2	Yes
14	Sunny	High	Dummy2	No
15	Sunny	High	Weak	Yes
17	Sunny	High	Strong	Yes
18	Sunny	High	Strong	No
19	Sunny	Normal	Dummy2	Yes
20	Sunny	Normal	Dummy2	No
21	Sunny	Normal	Weak	Yes
22	Sunny	Normal	Weak	No
23	Sunny	Normal	Strong	Yes
24	Sunny	Normal	Strong	No
25	Overcast	High	Dummy2	Yes
26	Overcast	High	Dummy2	No
27	Overcast	High	Weak	Yes
28	Overcast	High	Weak	No
29	Overcast	High	Strong	Yes
30	Overcast	High	Strong	No

31	Overcast	Normal	Dummy2	Yes
32	Overcast	Normal	Dummy2	No
33	Overcast	Normal	Weak	Yes
34	Overcast	Normal	Weak	No
35	Overcast	Normal	Strong	Yes
36	Overcast	Normal	Strong	No
37	Rain	High	Dummy2	Yes
38	Rain	High	Dummy2	No
39	Rain	High	Weak	Yes
40	Rain	High	Weak	No
41	Rain	High	Strong	Yes
42	Rain	High	Strong	No
43	Rain	Normal	Dummy2	Yes
44	Rain	Normal	Dummy2	No
46	Rain	Normal	Weak	No
47	Rain	Normal	Strong	Yes
48	Rain	Normal	Strong	No

Table 2-12 Perturbing data sets T^P with dummy attribute value on *Outlook* and *Wind* .

2.3 Generating Decision Tree from Unrealized Training Set

In the previous section, we discussed an algorithm that generates an unrealized training set T' and a perturbing set T^P from the samples in T_S. In this section, we use

data tables T' and T^P as the means to calculate the information content and information gain of T_S, such that a modified decision tree learning of the original data sets can be performed.

2.3.1 Tuple Notations used by the Modified Data Mining Approach

The following sections will explore research details of the modified decision tree learning algorithm. In this section, we will explain some common tuple notations used by the algorithm:

1) $T_{(a=k)}$ denotes the subset of T that contains all data sets satisfying the condition $(a = k)$. If $T = \{< Strong, Yes >, < Strong, No >, < Weak, Yes >, < Weak, Yes >, < Weak, Yes >, < Weak, No >\}$, then $T_{(Wind=Weak)} = \{< Weak, Yes >, < Weak, Yes >, < Weak, Yes >, < Weak, No >\}$.
2) $T_{(a\neq k)}$ denotes $T - T_{(a=k)}$.
3) $T_{(a_i=k)\wedge...\wedge(a_j=l)}$ denotes the subset of T that contains all data sets satisfying the condition $(a_i = k) \wedge ... \wedge (a_j = l)$.

2.3.2 Traditional Decision Tree Generation Approach

The traditional ID3[13] decision tree generating algorithm (shown as Figure 2-4) establishes the foundation for many popular decision tree mining algorithms such as C4.5[14]. All of these algorithms use information gain as the selection criteria of the test attribute for the function CHOOSE-ATTRIBUTE. Based on the recursive calls in the

DECISION-TREE LEARNING algorithm, the outcome decision tree will be built from sub-trees with maximum information gain in every node. Information gain measures the change of uncertainty level after a classification from an attribute. Fundamentally, this measurement is rooted in information theory. The definition of information gain *Gain* is shown as following:

$$Gain(a_j) = H_{a_i}(T_S) - H_{a_i}(T_S \mid a_j)$$

where $H_{a_i}(T_S)$ is the information content of decision attribute a_i before the test, equals:

$$H_{a_i}(T_S) = -\sum_{j=1}^{m} P(a_i = v_i)\log_2 P(a_i = v_i) = -\sum_{j=1}^{m} \frac{|T_{S(a_i=v_i)}|}{|T_S|}\log_2\left(\frac{|T_{S(a_i=v_i)}|}{|T_S|}\right)$$

and $H_{a_i}(T \mid a_j)$ is the condition information content of a_i with a given attribute a_j, equals:

$$H_{a_i}(T_S \mid a_j) = \sum_{i=1}^{n} P(a_j = k_i) H_{a_i}(T_{S(a_j=k_i)}) = \sum_{i=1}^{n} \frac{|T_{S(a_j=k_i)}|}{|T_S|} H_{a_i}(T_{S(a_j=k_i)})$$

The higher the information gains of an attribute test, the lower the uncertainty contained in its decision. Therefore, by comparing the information gain among the attributes available as an internal node, we can find the best test attributes in the decision-tree learning process. By applying the regular ID3 method on the samples in Table 2-1, we will produce the decision tree shown in Figure 2-5.

```
function DECISION-TREE LEARNING(samples, attributes, default) returns a decision tree
    inputs:samples, set of samples
        attributes, set of attributes
        default, default value for the goal predicate
    if samples is empty then
        return default
    else if all samples have the same classification then
        return the classification
    else if attributes is empty then
        return MAJORITY-VALUE(samples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, samples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            samples_i ← {elements of samples with best = v_i}
            m ← MAJORITY-VALUE(samples_i)
            subtree ← DECISION-TREE-LEARNING(samples_i,
            attributes – best, m)
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Figure 2-4 Pseudocode of the decision tree learning algorithm.

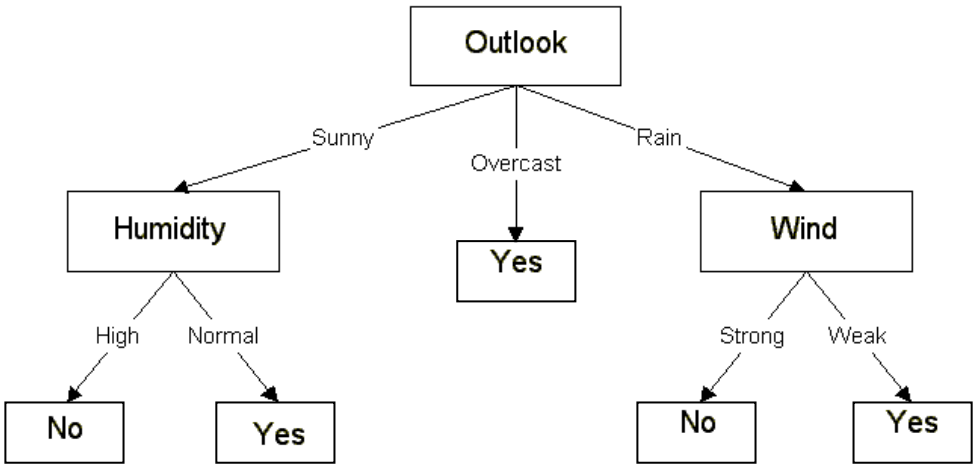

Figure 2-5 The final decision tree built from the training set in Table 2-1.

2.3.3 Fong and Weber's Approach

Fong and Weber introduced another approach that can generate the same decision tree with 100% accuracy directly from the unreal samples. They proved that information gain, the information content of decision attribute and the condition information content with a given attribute can be determined from the unreal samples without reconstructing any original data set. The new definition of information gain *Gain* is shown as following:

Lemma 2.3: $Gain(a_j) = H_{a_i}(q[T'+T^p]^C) - H_{a_i}(q[T'+T^p]^C \mid a_j)$

where the following lemmas hold given attribute a_i has n_i possible attribute values k_i:

Lemma 2.4: $H_{a_i}(q[T'+T^p]^C) = -\sum_{i=1}^{n_i} \frac{\frac{|qT^U|}{n_i} - |[T'+T^P]_{(a_i=k_i)}|}{|qT^U| - |[T'+T^P]|} \log_2 \frac{\frac{|qT^U|}{n_i} - |[T'+T^P]_{(a_i=k_i)}|}{|qT^U| - |[T'+T^P]|}$

Lemma 2.5: $$H_{a_i}(q[T'+T^P]^C \mid a_j) = -\sum_{j=1}^{n_j}\sum_{i=1}^{n_i}\frac{\frac{|qT^U|}{n_i * n_j}-|[T'+T^P]_{(a_i=k_i)\wedge(a_j=k_j)}|}{|qT^U|-|[T'+T^P]|}\log_2\frac{\frac{|qT^U|}{n_i * n_j}-|[T'+T^P]_{(a_i=k_i)\wedge(a_j=k_j)}|}{\frac{|qT^U|}{n_j}-|[T'+T^P]_{(a_j=k_j)}|}$$

Lemma 2.6: $$H_{a_i}(q[T'+T^P]^C{}_{(a_j=k_j)}) = -\sum_{i=1}^{n_i}\frac{\frac{|qT^U|}{n_i * n_j}-|[T'+T^P]_{(a_i=k_i)\wedge(a_j=k_j)}|}{\frac{|qT^U|}{n_j}-|[T'+T^P]_{(a_j=k_j)}|}\log_2\frac{\frac{|qT^U|}{n_i * n_j}-|[T'+T^P]_{(a_i=k_i)\wedge(a_j=k_j)}|}{\frac{|qT^U|}{n_j}-|[T'+T^P]_{(a_j=k_j)}|}$$

Lemma 2.7: $$|qT^U| = 2*|T'|+|T^P|$$

Lemma 2.8: $$|qT^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}| = \frac{2*|T'|+|T^P|}{n_i * \ldots * n_j}$$

These equations can be applied to the modified decision tree learning algorithm shown as Figure 2-6 for generating a decision tree with the unreal samples. It guarantees the result accuracy because it preserves the information gain on every subtree on every level. If we redo the decision tree mining process with the function DECISION-TREE LEARNING' and samples ($T'+T^P$) either in Table 2-8 and Table 2-9 or Table 2-11 and Table 2-12, then we will retrieve the decision tree shown as Figure 2-7, which is the same as the tree we built from applying the traditional decision tree generating method on the original samples.

function DECISION-TREE LEARNING'(*size*, T', T^P, *attributes*, *default*)
returns a decision tree
    inputs: *size*, size of the *q*-multiple-of universal set
        T', set of unreal training data sets
        T^P, set of perturbing data sets
        attributes, set of attributes
        default, default value for the goal predicate
    if $(T'+T^P)$ is empty **then**
        return *default*
    else if $H_{a_i}(q[T'+T^P]^C) = 0$ **then**
        return MINORITY-VALUE$(T'+T^P)$
    else if *attributes* is empty **then**
        return MINORITY-VALUE$(T'+T^P)$
    else
        best ← CHOOSE-ATTRIBUTE(*attributes*, *size*, $(T'+T^P)$)
        tree ← a new decision tree with root test *best*
        size ← *size* / number of possible values v_i in *best*
        for each value v_i of *best* **do**
            T'_i ← {data sets in T' with *best* = v_i}
            $T^P{}_i$ ← {data sets in T^P with *best* = v_i}
            m ← MINORITY-VALUE$(T'_i+T^P{}_i)$
            subtree ← DECISION-TREE-LEARNING'(*size*, T'_i, $T^P{}_i$, *attributes* – *best*, *m*)
            add a branch to tree with label v_i and subtree *subtree*
        return *tree*

Figure 2-6 Pseudocode of the modified decision tree learning algorithm using T' and T^P.

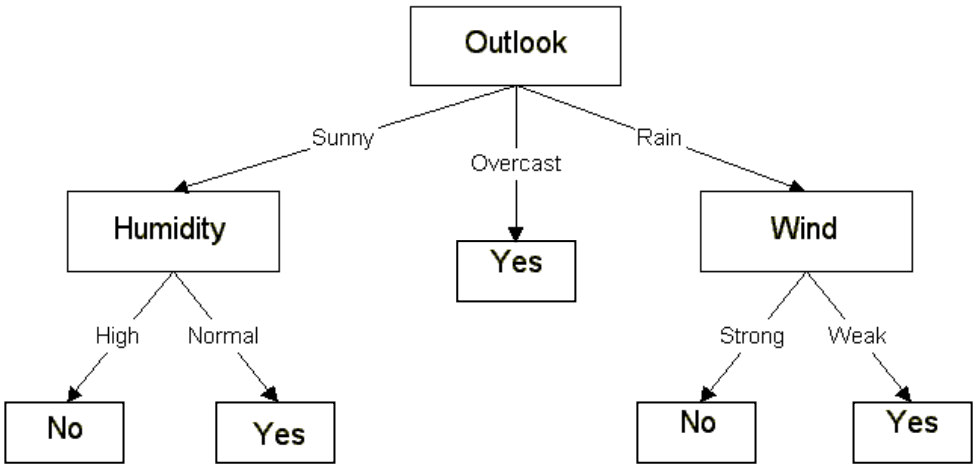

Figure 2-7 The final decision tree built from data sets in Table 2-12 and 2-13.

2.4 Evaluation and Limitations

In section 2.3, we show that the Data Unrealization approach keeps the full utility of data mining result from the usage of decision tree mining. In this section, we will discuss the evaluation works from two dimensions: privacy protection and storage complexity. The analysis shown in this section is based on Data Unrealization with doubling the sample domain.

From the privacy preservation aspect, Fong and Weber defines the privacy loss function as:

$$P_{loss}(T_S, T_D) = \frac{r}{|T_D|} * P(T_S, T_D)$$

where T_S and T_D are denoted as the data tables of the original sample and the sanitized database, r is the number of data sets lost, $|T_D|$ is the total number of data sets in the

sanitized database and $P(T_S, T_D)$ is the total amount of privacy information of the whole database T_D. Without privacy protection, the privacy loss is ranged from:

$$r * \frac{|T_S|}{|T^U|} \leq P_{loss}(T_S, T_S) \leq r^* |T_S|$$

With Data Unrealization, the privacy loss can be decreased to:

$$0 \leq P_{loss}(T_S, T' + T^P) \leq r * \frac{|T_S| * (|T^U| - \sqrt{2|T^U| - 1})}{2 * |T^U| * (|T^U| - 1)}$$

Consequently, the Data Unrealization approach protects privacy effectively. However, from the storage complexity aspect, the storage requirement is $(2*|T^U|-1)*|T_S|$ in the worst case scenario, which means it requires $(2*|T^U|-1)$ times of the storage of the original sample size. At the same point, a database query may take $(2*|T^U|-1)$ times longer to scan through the database for retrieving the count of a data set.

Other than the storage requirement, a downside of Fong and Weber's research is the scope limitation on different data mining applications. They proved that Data Unrealization is a powerful means to preserve both privacy and utility of the samples. However, if it is merely applicable for decision tree mining, then the usability of the approach itself would be too limited because people cannot use Data Unrealization to preserve privacy of training samples applied for other data mining algorithms.

Chapter 3

DATA UNREALIZATION – SCOPE OF APPLICATION

In Fong and Weber's research, they proved the concepts of Data Unrealization by using decision tree learning as an example. For our research, we are going to expand the application coverage of Data Unrealization for other data mining approaches. Indeed, Data Unrealization hides the data privacy of discrete-value samples but not their statistical information (i.e. counts, probability and information entropy); hence, those statistics is still retrievable for any data mining algorithm that is designed for this type of samples. In this dissertation, we will explore the research from the angle of data mining; even so, our research finding is also applicable for extracting statistical information from those unrealized samples for other purposes, such as statistical analysis.

3.1 Data Unrealization and Set Theory

The concept of Data Unrealization is closely related to Set Theory[15]. If there is a space U and it holds a subset A, then its complement set A' under the same space should contain the items not in A (see Figure 3-1(a)). Hence, Set Theory ensures that:

$$|A| + |A'| = |U|$$

Section 2.3.1 states the relationship between the original training sets T_S and the unrealized training sets $(T'+T^P)$. $(T'+T^P)$ contains all data sets not in T_S under the space of qT^U, which means T_S and $(T'+T^P)$ are the complement set of each other

under the space of qT^U. From Figure 3-1(b), we can find that the same rule also applies for each individual tuple t. There are q units of t in the space of qT^U, in which some of them are in T_S and all the rest are in $(T'+T^P)$; therefore, the following lemma holds:

Lemma 3.1: $|T_S| + |(T'+T^P)| = |qT^U| = q^*|T^U|$

If we narrow the scope down to a specific tuple, say t_1 (see Figure 3-1(b)), then we can define:

$$|T_{S[t_1]}| + |(T'+T^P)_{[t_1]}| = |qT^U{}_{[t_1]}| = q$$

Indeed, if $t \subseteq T^U$ (which implies t is the union of some independent tuples in T^U), then the following lemma applies in general:

Lemma 3.2: $|T_{S[t]}| + |(T'+T^P)_{[t]}| = |qT^U{}_{[t]}| = q^*|T^U{}_{[t]}|$

The above complement set theories are very crucial for the discussion in section 3.2 – 3.4.

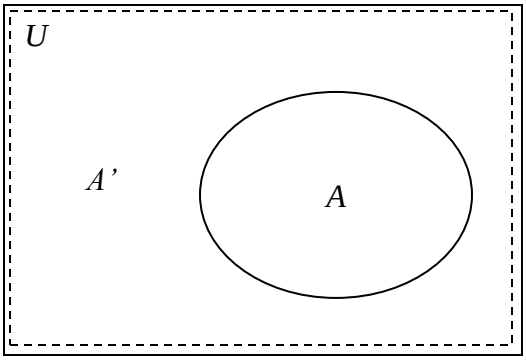

Figure 3-9(a) The illustration of space U and two compliment subsets in U : A and A'.

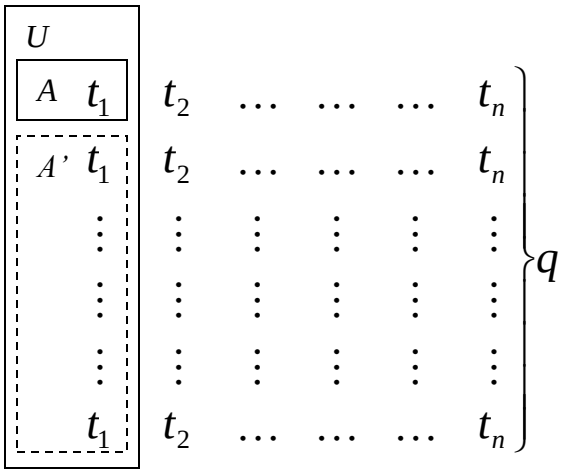

Figure 3-9(b) The illustration of space U as qT^U and two compliment subsets in the space: A as T_S and A' as $(T'+T^P)$.

3.2 Counting-based Data Mining Approaches

Most association rule learning and itemset mining[16] algorithms are counting based – which means the frequency of t in T_S is required for the data mining process. In

a later section, we will demonstrate the way to perform counting-based data mining directly from $(T'+T^P)$ by using Apriori algorithm[17] as an example.

3.2.1 Counting Lemmas

By Lemma 2.1, Lemma 2.8, Lemma 3.1 and Lemma 3.2, we can compute the counts (size or frequency) of any tuple *t* in the original training set from the unreal training set. According to Lemma 2.1, the size of T_S can be determined as:

Lemma 2.1: $|T_S| = |T'|$

Based on the space of $qT^U{}_{[t]}$, the counts of $T_{S[t]}$ can be determined from Lemma 3.1 and Lemma 3.2 as:

Lemma 3.3: $$|T_{S[t]}| = |qT^U{}_{[t]}| - |(T'+T^P)_{[t]}| = q^*|T^U{}_{[t]}| - |(T'+T^P)_{[t]}|$$

$$\because q^*|T^U| = |T_S| + |(T'+T^P)|$$

$$\therefore |T_{S[t]}| = \frac{2^*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[t]}|}} - |T'_{[t]}| - |T^P{}_{[t]}|$$

Derived from Lemma 2.8 with the space of $qT^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}$, the counts of T_S under the condition of $(a_i = k_i)\wedge\ldots\wedge(a_j = k_j)$, which is denoted as $T_{S(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}$, can be defined as:

Lemma 3.4: $$|T_{S(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}| = |qT^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}| - |(T'+T^P)_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}|$$

$$\because |qT^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}| = \frac{2^*|T'|+|T^P|}{n_i * \ldots * n_j}$$

$$\therefore |T_{S(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}| = \frac{2*|T'|+|T^P|}{n_i*\ldots*n_j} - |T'_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}| - |T^P{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}|$$

where attribute a_i has n_i possible attribute values k_i. Lemma 3.3 and Lemmas 3.4 are equivalent, in which Lemma 3.3 is based on set notation and Lemma 3.4 is based on tuple notation. For the rest of this dissertation, we will use the set notation for our writing consistency.

3.2.2 Apriori Algorithm

Figure 3-2 illustrates the original Aprior algorithm that returns the frequent itemsets in T_S with supports larger than ε. In the function APRIOR, we need to compute the frequency of an itemset f in T_S (i.e. $C_f \leftarrow$ Count the number of t in T_S that contains f, which is $|T_{S[f]}|$) and the size of the samples T_S.

To prove our work, we need to determine these two pieces of information from the unrealized samples. After Data Unrealization, T_S and $(T'+T^P)$ become the complement set of each other in the space of qT^U; therefore, Lemma 2.1, Lemma 3.3 and Lemma 3.4 can be applied to extract the counting information $|T_{S[f]}|$ and $|T_S|$ directly from T' and T^P, which implies:

$$|T_S| = |T'|$$

and

$$|T_{S[f]}| = \frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[f]}|}} - |T'_{[f]}| - |T^P{}_{[f]}|$$

Figure 3-3 illustrates the modified Aprior algorithm based on the inputs of the unrealized training samples. In the function APRIOR', $|T_S|$ is replaced by $|T'|$ according to Lemma 2.1 and $|T_{S[f]}|$ can be derived from $\frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[f]}|}} - |T'_{[f]}| - |T^P{}_{[f]}|$ according to Lemma 3.3. If the universal set is the power set of all possible attribute values, we can optimize the function (see Figure 3-4) by introducing a variable *size* that stores the pre-computed value of $\frac{|T^U|}{|T^U{}_{[f]}|}$ dynamically. Since all tuples in T^U are evenly distributed, $\frac{|T^U|}{|T^U{}_{[f]}|}$ equals to $2^{length(f)}$ (for example, if there are three items a, b and c to form all itemsets, which imply $T^U = \{\phi, a, b, c, ab, ac, bc, abc\}$, then $\frac{|T^U|}{|T^U{}_{[a]}|} = \frac{|T^U|}{|T^U{}_{[b]}|} = \frac{|T^U|}{|T^U{}_{[c]}|} = 2$, $\frac{|T^U|}{|T^U{}_{[ab]}|} = \frac{|T^U|}{|T^U{}_{[ac]}|} = \frac{|T^U|}{|T^U{}_{[bc]}|} = 4$ and $\frac{|T^U|}{|T^U{}_{[abc]}|} = 8$). Hence, $|T_{S[f]}|$ is equivalent to $\frac{2*|T'|+|T^P|}{size} - |T'_{[f]}| - |T^P{}_{[f]}|$ because $size = 2^{length(f)} = \frac{|T^U|}{|T^U{}_{[f]}|}$.

For example, if we use the original samples in Table 2-1 as the input T_S of function APRIOR with support threshold ε = 30%, then the resulting frequent itemset table will be found in three iterations where the content of T_O after the first, second and third iterations is shown as Table 3-1, Table 3-2 and Table 3-3 respectively. Meanwhile, if we use the unrealized training samples in Table 2-11 and Table 2-12 as the inputs T'

and T^P of function APRIOR' with support threshold ε = 30%, we can get the resulting frequent itemset table T_O (shown as Table 3-4), which is the same as the original finding, in three iterations.

```
function APRIOR(T_S, ε) returns a set of frequent itemsets with supports
    input: T_S, set of training data sets
           ε, a support threshold
    k ← 1
    T_O ← {}
    do
        next ← false
        if k = 1 then
            F ←Generate itemsets of length k
        else
            F ← Generate itemsets of length k
            from itemsets of length k−1 in T_O
        for each f in F
            C_f ← Count the number of t in T_S that contains f
            if C_f ≥ ε*|T_S| then
                T_O ← T_O + {< f , C_f /|T_S| >}
                next ← true
        k ← k+1
    while next = true
    return T_O
```

Figure 3-10 Pseudocode of the classic Aprior algorithm.

function APRIOR'(T', T^P, ε) **returns** a set of frequent itemsets with supports

input: T', set of unreal training data sets
T^P, set of perturbing data sets
ε, a support threshold

$k \leftarrow 1$
$T_O \leftarrow \{\}$
do
next ← false
if $k = 1$ **then**
$F \leftarrow$ Generate itemsets of length k
else
$F \leftarrow$ Generate itemsets of length k from itemsets of length $k-1$ in T_O
for each f in F
$C'_f \leftarrow$ Count the number of t in T' that contains f
$C^P{}_f \leftarrow$ Count the number of t in T^P that contains f
$C_f \leftarrow \frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[f]}|}} - C'_f - C^P{}_f$
if $C_f \geq \varepsilon * |T'|$ **then**
$T_O \leftarrow T_O + \{< f , C_f / |T'| >\}$
next ← true
$k \leftarrow k+1$
while *next* = true
return T_O

Figure 3-11 Pseudocode of the modified Aprior algorithm applied for unrealized training samples.

function APRIOR2'(T', T^P, ε) **returns** a set of frequent itemsets with supports
input: T', set of unreal training data sets
T^P, set of perturbing data sets
ε, a support threshold
$k \leftarrow 1$
size ←2
$T_O \leftarrow \{\}$
do
next ← false
if $k = 1$ **then**
$F \leftarrow$Generate itemsets of length k
else
$F \leftarrow$ Generate itemsets of length k
from itemsets of length $k-1$ in T_O
for each f in F
$C'_f \leftarrow$ Count the number of t in T' that contains f
$C^P{}_f \leftarrow$ Count the number of t in T^P that contains f
$C_f \leftarrow \frac{2*|T'|+|T^P|}{size} - C'_f - C^P{}_f$
if $C_f \geq \varepsilon * |T'|$ **then**
$T_O \leftarrow T_O + \{< f , C_f / |T'| >\}$
next ← true
$k \leftarrow k+1$
size←*size**2
while *next* = true
return T_O

Figure 3-12 Pseudocode of the modified Aprior algorithm applied for unrealized training samples with optimization.

Itemset	Supports
{ *Outlook* = *Sunny* }	5
{ *Outlook* = *Rain* }	5
{ *Humidity* = *High* }	7

$\{ Humidity = Normal \}$	7
$\{ Wind = Strong \}$	6
$\{ Wind = Weak \}$	8
$\{ Play = Yes \}$	9
$\{ Play = No \}$	5

Table 3-13 The frequent itemset table T_O after the first iteration with support threshold = 30%.

Itemset	Supports
$\{ Outlook = Sunny \}$	5
$\{ Outlook = Rain \}$	5
$\{ Humidity = High \}$	7
$\{ Humidity = Normal \}$	7
$\{ Wind = Strong \}$	6
$\{ Wind = Weak \}$	8
$\{ Play = Yes \}$	9
$\{ Play = No \}$	5
$\{ Humidity = Normal, Play = Yes \}$	6
$\{ Wind = Weak, Play = Yes \}$	6

Table 3-14 The frequent itemset table T_O after the second iteration with support threshold = 30%.

Itemset	Supports
$\{ Outlook = Sunny \}$	5
$\{ Outlook = Rain \}$	5
$\{ Humidity = High \}$	7
$\{ Humidity = Normal \}$	7
$\{ Wind = Strong \}$	6
$\{ Wind = Weak \}$	8
$\{ Play = Yes \}$	9
$\{ Play = No \}$	5
$\{ Humidity = Normal , Play = Yes \}$	6
$\{ Wind = Weak , Play = Yes \}$	6

Table 3-15 The resulting frequent itemset table T_O after the third iteration with support threshold = 30%.

Itemset	Supports
$\{ Outlook = Sunny \}$	5
$\{ Outlook = Rain \}$	5
$\{ Humidity = High \}$	7
$\{ Humidity = Normal \}$	7
$\{ Wind = Strong \}$	6
$\{ Wind = Weak \}$	8
$\{ Play = Yes \}$	9

$\{ Play = No \}$	5
$\{ Humidity = Normal,\ Play = Yes \}$	6
$\{ Wind = Weak,\ Play = Yes \}$	6

Table 3-16 The resulting frequent itemset table T_O by applying the APRIOR' function with the unrealized training sets.

3.3 Probability-based Data Mining Approaches

Other than the counts of each data set, probability is another piece of information required by many data mining classification algorithms. These probability-based approaches categorize the evidences / results based on the probability of some events / facts of t in T_S. In this section, we will use Naïve Bayes classifier[18] as an example to illustrate probability-based data mining from $(T'+T^P)$.

3.3.1 Probability Lemmas

If x is a random event, t is a resulting event and c is a known conditional event in T_S, then $P(x \in T_{S[t]})$ (which means the probability of "t contains x") and $P(x \in T_{S[t]} \mid x \in T_{S[c]})$ (which means the probability of "t contains x under the condition of c contains x") can be determined as the following lemmas based on Lemma 3.3:

Lemma 3.5: $$P(x \in T_{S[t]}) = \frac{|T_{S[t]}|}{|T_S|} = \left(\frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[t]}|}} - |T'_{[t]}| - |T^P{}_{[t]}|\right) * \frac{1}{|T_S|}$$

$$\because |T_S| = |T'|$$

$$\therefore \frac{|T_{S[t]}|}{|T_S|} = \left(\frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[t]}|}} - |T'_{[t]}| - |T^P{}_{[t]}|\right) * \frac{1}{|T'|}$$

Lemma 3.6: $P(x \in T_{S[t]} \mid x \in T_{S[c]}) = \frac{|T_{S[t\cap c]}|}{|T_{S[c]}|}$

$$= \frac{\frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[t\cap c]}|}} - |T'_{[t\cap c]}| - |T^P{}_{[t\cap c]}|}{\frac{2*|T'|+|T^P|}{\frac{|T^U|}{|T^U{}_{[c]}|}} - |T'_{[c]}| - |T^P{}_{[c]}|}$$

$$= \frac{(2*|T'|+|T^P|)*|T^U{}_{[t\cap c]}| - (|T'_{[t\cap c]}| + |T^P{}_{[t\cap c]}|)*|T^U|}{(2*|T'|+|T^P|)*|T^U{}_{[c]}| - (|T'_{[c]}| + |T^P{}_{[c]}|)*|T^U|}$$

3.3.2 Naïve Bayes Probabilistic Model

The Naïve Bayes Probabilistic model is a conditional model that determines the possibility of "a random event x belongs to a known event t given the conditions that x belongs to c_1 through c_n", which is $P(x \in T_{S[t]} \mid x \in T_{S[c_1]} \wedge \ldots \wedge x \in T_{S[c_n]})$. As Bayes' theorem assumes all events are independent to the others, it defines:

$$P(x \in T_{S[t]} \mid x \in T_{S[c_1]} \wedge \ldots \wedge x \in T_{S[c_n]})$$

$$= \frac{P(x \in T_{S[c_1]} \wedge \ldots \wedge x \in T_{S[c_n]} \mid x \in T_{S[t]}) * P(x \in T_{S[t]})}{P(x \in T_{S[c_1]} \wedge \ldots \wedge x \in T_{S[c_n]})}$$

and

$$P(x \in T_{S[c_1]} \wedge \ldots \wedge x \in T_{S[c_n]} \mid x \in T_{S[t]})$$

$$= P(x \in T_{S[c_1]} \mid x \in T_{S[t]}) * \ldots * P(x \in T_{S[c_n]} \mid x \in T_{S[t]})$$

The above theorems are the fundamental equations supporting the entire probabilistic model with samples T_S. By applying Lemma 3.5 and Lemma 3.6, we can determine $P(x \in T_{S[c]} \mid x \in T_{S[t]})$, $P(x \in T_{S[t]})$ and $P(x \in T_{S[c_1]} \wedge \ldots \wedge x \in T_{S[c_n]})$ from the unrealized samples T' and T^P. The equations are given as the following:

$$P(x \in T_{S[c]} \mid x \in T_{S[t]}) = \frac{(2*|T'| + |T^P|)*|T^U{}_{[t \cap c]}| - (|T'_{[t \cap c]}| + |T^P{}_{[t \cap c]}|)*|T^U|}{(2*|T'| + |T^P|)*|T^U{}_{[t]}| - (|T'_{[t]}| + |T^P{}_{[t]}|)*|T^U|},$$

$$P(x \in T_{S[t]}) = (\frac{2*|T'| + |T^P|}{\frac{|T^U|}{|T^U{}_{[t]}|}} - |T'_{[t]}| - |T^P{}_{[t]}|) * \frac{1}{|T'|}$$

and

$$P(x \in T_{S[c]}) = (\frac{2*|T'| + |T^P|}{\frac{|T^U|}{|T^U{}_{[c]}|}} - |T'_{[c]}| - |T^P{}_{[c]}|) * \frac{1}{|T'|}.$$

For example, if we want to determine whether $Play = Yes$ or $Play = No$ given the condition as $<Outlook = Sunny,\ Humidity = High$ and $Wind = Strong>$ from the original samples in Table 2-1, then the probability of $Play = Yes$ can be computed as follows:

$$P(Outlook = Sunny \mid Play = Yes) = \frac{2}{9}$$

$$P(Humidity = High \mid Play = Yes) = \frac{3}{9}$$

$$P(Wind = Strong \mid Play = Yes) = \frac{3}{9}$$

$$P(Play = Yes) = \frac{9}{14}$$

and let's assume:

$$P(Outlook = Sunny \wedge Humidity = High \wedge Wind = Strong) = \frac{1}{\alpha};$$

then,

$$P(Play = Yes \mid Outlook = Sunny \wedge Humidity = High \wedge Wind = Strong)$$
$$= (\frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{9}{14}) * \alpha$$
$$= 0.0159 * \alpha$$

For the probability of $Play = No$, we can compute it as follows:

$$P(Outlook = Sunny \mid Play = No) = \frac{3}{5}$$

$$P(Humidity = High \mid Play = No) = \frac{4}{5}$$

$$P(Wind = Strong \mid Play = No) = \frac{3}{5}$$

$$P(Play = No) = \frac{5}{14};$$

therefore,

$$P(Play = No \mid Outlook = Sunny \wedge Humidity = High \wedge Wind = Strong)$$
$$= (\frac{3}{5} * \frac{4}{5} * \frac{3}{5} * \frac{5}{14}) * \alpha$$
$$= 0.1026 * \alpha$$

As a result, we can conclude that $Play = No$ if $Outlook = Sunny$, $Humidity = High$ and $Wind = Strong$ because $0.1026 * \alpha$ is larger than $0.0159 * \alpha$.

To verify the result by using the unrealized samples in Table 2-11 and Table 2-12, we re-compute the probability of $Play = No$ as the following:

$$P(Outlook = Sunny \mid Play = No) = \frac{(2*14+68)*6-(1+8)*48}{(2*14+68)*24-(7+36)*48} = \frac{3}{5}$$

$$P(Humidity = High \mid Play = No) = \frac{(2*14+68)*12-(4+16)*48}{(2*14+68)*24-(7+36)*48} = \frac{4}{5}$$

$$P(Wind = Strong \mid Play = No) = \frac{(2*14+68)*8-(2+11)*48}{(2*14+68)*24-(7+36)*48} = \frac{3}{5}$$

$$P(Play = No) = (\frac{2*14+68}{\frac{48}{24}} - 7 - 36)*\frac{1}{14} = \frac{5}{14};$$

hence,

$$P(Play = No \mid Outlook = Sunny \wedge Humidity = High \wedge Wind = Strong)$$
$$= (\frac{3}{5}*\frac{4}{5}*\frac{3}{5}*\frac{5}{14})*\alpha$$
$$= 0.1026*\alpha$$

All of the above calculated results are the same as the results from the original samples, therefore validating the probability lemmas provide the same conclusion as the original by using the unrealized samples.

3.4 Entropy-based Data Mining Approaches

Entropy-based methods seek to iteratively minimize uncertainty when an outcome needs to be predicted from some known information. Some classifying or clustering algorithms utilize entropy for data mining purpose. Entropy-based data mining approaches select an outcome with the least uncertainty based on the information gain from T_S. In Fong and Weber's research, they developed the formulae that calculate information gain from the unrealized samples and gave the decision tree algorithm as a typical example. They have already provided the lemmas (which are referenced as

Lemma 2.3 through Lemma 2.8 in this dissertation) with proofs and explained the unrealized data mining algorithm with examples.

As their research successfully validates the application of Data Unrealizaion on entropy-based data mining, we will not duplicate the contents in this dissertation. For the full details and analysis, please refer to Section 2.2 and Section 2.3 of this dissertation and the original papers[5, 32].

Chapter 4

JOIN INFORMATION SEQUENCE[6][7]

One of the main objectives of this dissertation is eliminating the storage requirement by applying data compression to the unrealized training sets. To achieve this purpose, we invent a number sequence – named Join Information Sequence (referenced to as J-Sequence in short) – that establishes a math space for representing object relations in graphs. By applying J-Sequence, we can represent the data sets in the unreal training samples and further data compress them. In this chapter, we will discuss the principles of J-Sequence in details. In later chapters, these principles will be used as the basis for explaining the application of unrealized training sets compression.

4.1 Introducing J-Sequence

Computer scientists always transform typical real life problems into logical questions for systematic analysis. This requires them to represent the information of the original problem as a mathematical model or with some data structures. Bitmap[23] (or binary array with fixed size) and linked list[24] are two common data structures used to representing the relation map among a group of objects with close relation. For example, they are applied as the main components in adjacency matrices[25] and adjacency lists[25] to store the information of "relationship-existence" between the source object and each of

[6] Some parts of this chapter have been submitted as a journal paper titled "Join Information Sequence – a Novel Way of Information Representation" in early 2013.

[7] For someone who references any content of this chapter, please put the QR code shown in the acknowledgement page in his / her paper.

the other objects in the neighborhood map[27]. The bitmap representation reserves a bit to store for relationship-existence information for each pair of objects. Therefore, it supports fast and simple lookup operation when we want to verify the relationship between two objects. However, it is not practical when we need to determine the "relationship-size", which is the number of connections from a particular object to other objects, especially for large maps with sparse relation. Meanwhile, the linked list representation needs to store the information of the connected objects from the source object only, such that the counting operation can be performed efficiently when the "relationship-size" is required; however, its code implementation is more complicated and its lookup operation is comparatively slow, especially for dense graphs.

The J-Sequence representation combines the benefits of the two methods. It can be applied to compress the relationship information into a number (which is the product of members of a J-Sequence, referred as "J-Product" hereafter). Furthermore, we can determine some useful information among multiple objects (such as their size comparison, information of their union set and information of their intersection set) by applying simple math operations on their J-Products. J_N denotes a J-Sequence with N members and the members can be assigned in any order. For the convention of this dissertation, we create a function *Order* such that the members of *Order*(J_N) are in ascending order.

4.2 Properties of J-Sequence and the J-Sequence Representation

A J-Sequence with size N is a sequence of N unique numbers having the following five properties whereas $J_k = \{j_1, j_2, \dots, j_{k-1}, j_k\} \subseteq J_N$ *and* $P_k = \prod_{i=1}^{k} J_i$:

1) $If\ j_c \in J_N, then\ P_k\ mod\ j_c \neq 0 \Leftrightarrow j_c \notin J_k\ and\ P_k\ mod\ j_c = 0 \Leftrightarrow j_c \in J_k.$
2) $If\ J_m = \{j_1, j_2, \dots, j_{m-1}, j_m\} \subseteq J_N\ with\ P_m = \prod_{i=1}^{m} J_i\ , then\ P_k\ mod\ P_m = 0 \Leftrightarrow J_m \subseteq J_k; otherwise\ there\ exists\ a\ number\ j_c\ such\ that\ j_c \in J_m\ and\ j_c \notin J_k.$
3) $If\ J_m = \{j_1, j_2, \dots, j_{m-1}, j_m\} \subseteq J_N\ with\ P_m = \prod_{i=1}^{m} J_i\ , then\ P_k \geq\ P_m \Longrightarrow k \geq m\ and\ k > m \Longrightarrow P_k >\ P_m.$
4) $If\ P_N = \prod_{i=1}^{N} J_i\ , then\ \left|\frac{\log(P_k)}{\log({P_N}^{1/N})} - k\right| < 0.5.$
5) $J_k\ is\ a\ J\text{-}Sequence\ with\ size\ k\ such\ that\ all\ properties\ of\ J\text{-}Sequence\ applies\ on\ J_k.$

Based on the above definitions of J_N, P_N, J_k and P_k, we can symbolize each object j_c within a relation map with a unique number in J_N and refer each relation of the map to an individual J_k. If the object j_c is a part of the relation J_k, then j_c will be considered as a member of J_k. In such a way, we can represent the relationship information of the relation J_k as its J-Product P_k, which is a number resulting from the product of some numbers. For example, if we have a group of four objects, say object1, object2, object3 and object4 with index 1, 2, 3 and 4 correspondingly and the bitmap representation of one of their relations is 0110 (which means that the objects with index 2 and 3 – object2 and object3 – exist as part of the relationship,) then we can select a J_4, says {5, 6, 7, 8}, to stand for object1, object2, object3 and object4 correspondingly and represent the relationship information as P_k = 6*7 = 42 (the illustration is shown as Figure 4-1.)

From the traditional approaches, we use a bitmap or a linked list data structure to represent relationship information and they require tailored computer algorithms for computing the relationship-existence and relationship-size information. By using the J-

Sequence approach, we compress the required relationship information as a number such that we can retrieve any useful piece of information by applying simple math operations such as rounding (referred to as function *round* in this dissertation,) modulus (referred to as operator % or *mod* in this dissertation) and logarithm (referred to as function *log* in this dissertation) that are usually provided by the standard math library of most programming languages. By Property 1, we can perform the relationship-existence lookup by calculating $P_k \ mod \ j_c$. By Property 4, we can determine the relationship-size by computing $|N * \log(P_k)/\log(P_N)|$ where N and $\log(P_N)$ are constants once a J_N is chosen for a group with a specific number of objects (please refer to Figure 1 for examples.)

The code implementation of the J-Sequence representation could be simple because it relies on basic mathematic calculations. Moreover, by Property 5, all subsets of a J-Sequence form new J-Sequences and all J-Sequence properties still applies on the subsets; thus, the formation of a J-Sequence with size N stands for a J-Sequence family, in which the members share common properties, with size less than or equal to N. However, one may question whether such a powerful sequence exists? In the following two sections, we are going to prove the existence of J-Sequence by showing: first, an algorithm returns a sequence of N unique numbers that guarantees all the five properties of J-Sequence (which proves the existence of J-Sequence except when an overflow exception is caught.) Second, a mathematical method generates a special J_N formed by any N prime numbers (as there are infinite numbers of prime numbers, so that a J-Sequence of any size exists theoretically.)

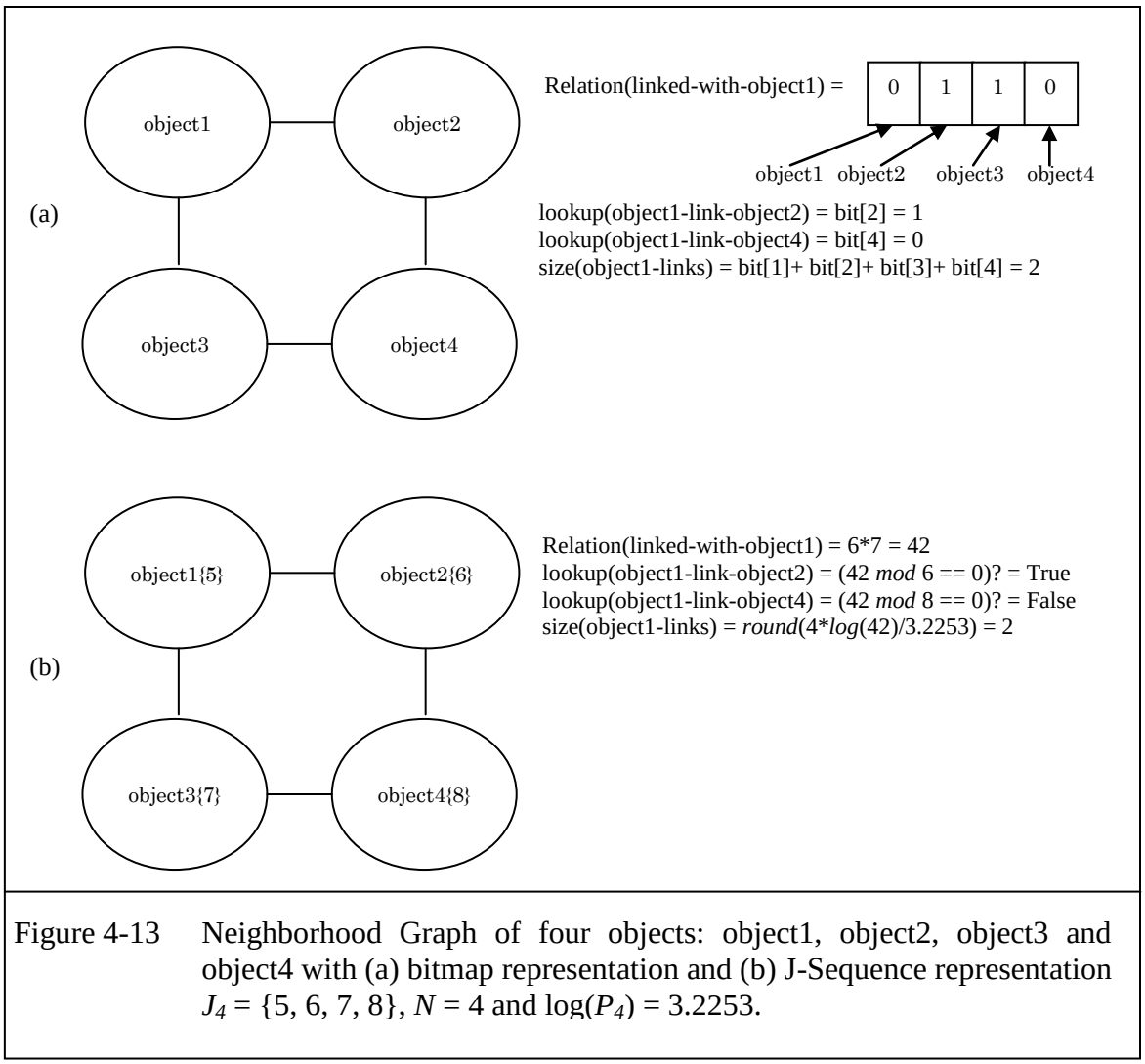

Figure 4-13 Neighborhood Graph of four objects: object1, object2, object3 and object4 with (a) bitmap representation and (b) J-Sequence representation $J_4 = \{5, 6, 7, 8\}$, $N = 4$ and $\log(P_4) = 3.2253$.

4.3 Generating J-Sequence by a Brute Force Approach

The GENERATE-J-SEQUENCE algorithm (shown as Figure 4-2) takes an input integer N as the size of the J-Sequence and returns a sequence of N unique numbers that follows the "Modulus-Rule", "Product-Rule" and "Size-Rule" (shown as Figure 4-5, Figure 4-6 and Figure 4-7 respectively.) If GENERATE-J-SEQUENCE (N) generates a *Order*(J_N) with J-Product c, geometric mean X and elements j_k (where $j_m \leq X <$

j_{m+1},) then these three rules are corresponding to the following mathematic equations respectively:

$$\frac{P_N}{j_k} \; mod \; j_k \neq 0$$

$$\prod_{k=1}^{\left\lceil \frac{N}{2} \right\rceil} j_k > \prod_{k=\left\lfloor \frac{N+1}{2} \right\rfloor + 1}^{N} j_k$$

$$\max\left(\left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log(X)} - m \right\|, \left\| \frac{\log(\prod_{k=m+1}^{N} j_k)}{\log(X)} - (N-m) \right\| \right) = 0$$

When a sequence satisfies all these three rules, it also holds the five properties of J-Sequence. We will prove that by the following theorems in this section:

Theorem 4.1: $If \; 1 \leq k \leq m \leq N \; and \; j_i, j_k \in J_N, then \; \frac{\prod_{i=1}^{N} j_i}{j_k} \; mod \; j_k \neq 0 \Longrightarrow$ $\frac{\prod_{i=1}^{m} j_i}{j_k} \; mod \; j_k \neq 0 \; and \; \prod_{i=1}^{m} j_i \; mod \; j_k = 0.$

Proof: $\prod_{i=1}^{m} j_i \; mod \; j_k = 0$

($\because 1 \leq k \leq m, trival$)

$$\frac{\prod_{i=1}^{N} j_i}{j_k} \; mod \; j_k \neq 0 \Longrightarrow \frac{(\prod_{i=1}^{m} j_i) * (\prod_{i=m+1}^{N} j_i)}{j_k} \; mod \; j_k \neq 0$$

$$If \; \frac{\prod_{i=1}^{m} j_i}{j_k} \; mod \; j_k = 0, then \; \frac{\prod_{i=1}^{N} j_i}{j_k} \; mod \; j_k = 0$$

$(Contradiction)$

$$Therefore, \frac{\prod_{i=1}^{N} j_i}{j_k} \; mod \; j_k \neq 0 \Longrightarrow \frac{\prod_{i=1}^{m} j_i}{j_k} \; mod \; j_k \neq 0$$

Q.E.D.

Theorem 4.2: $If \; 1 \leq k, p, q \leq m \leq N \; and \; j_i, j_k \in J_N, then \; \frac{\prod_{i=1}^{N} j_i}{j_k} \; mod \; j_k \neq$ $0 \Longrightarrow \frac{\prod_{i=1}^{m} j_i}{\prod_{i=1}^{q} j_i} \; mod \; \prod_{i=1}^{p} j_i \neq 0 \; and \; \prod_{i=1}^{m} j_i \; mod \; \prod_{i=1}^{p} j_i = 0$.

Proof: $\prod_{i=1}^{m} j_i \, mod \, \prod_{i=1}^{p} j_i = 0$

$(\because 1 \le p \le m, trival)$

$\frac{\prod_{i=1}^{N} j_i}{j_1} \, mod \, j_1 \neq 0 \Longrightarrow \frac{\prod_{i=1}^{m} j_i}{j_1} \, mod \, j_1 \neq 0 \Longrightarrow \frac{\prod_{i=1}^{m} j_i}{j_1} \, mod \, \prod_{i=1}^{p} j_i \neq 0 \Longrightarrow$

$\frac{\prod_{i=1}^{m} j_i}{\prod_{i=1}^{q} j_i} \, mod \, \prod_{i=1}^{p} j_i \neq 0$

Q.E.D.

Theorem 4.3: $If \; 1 \le p < m \le N, j_k \in Order(J_N), j_l \in J_l \subseteq J_N \; and \; j_r \in J_r \subseteq J_N, \; then \; \prod_{k=1}^{\left\lceil \frac{N}{2} \right\rceil} j_k > \prod_{k=\left\lfloor \frac{N+1}{2} \right\rfloor + 1}^{N} j_k \Longrightarrow \prod_{l=1}^{m} j_l > \prod_{r=1}^{p} j_r.$

Proof: $\because \prod_{l=1}^{m} j_l \; \ge \prod_{k=1}^{m} j_k \;\; and \; \prod_{r=1}^{p} j_r \; \le \prod_{k=N-p+1}^{N} j_k$

$\therefore \frac{\prod_{l=1}^{m} j_l}{\prod_{r=1}^{p} j_r} \ge \frac{\prod_{k=1}^{m} j_k}{\prod_{r=1}^{p} j_r} \ge \frac{\prod_{k=1}^{m} j_k}{\prod_{k=N-p+1}^{N} j_k} > 1$

$(\because \prod_{k=1}^{\left\lceil \frac{N}{2} \right\rceil} j_k > \prod_{k=\left\lfloor \frac{N+1}{2} \right\rfloor + 1}^{N} j_k \; and \; p < m)$

Q.E.D.

Theorem 4.4: $If \; 1 \le m, p \le N, j_k \in J_k \subseteq J_N, j_i \in J_i \subseteq J_N, j_l \in Order(J_N) \; and \; \prod_{l=1}^{\left\lceil \frac{N}{2} \right\rceil} j_l > \prod_{l=\left\lfloor \frac{N+1}{2} \right\rfloor + 1}^{N} j_l, then \; m > p \Longrightarrow \prod_{k=1}^{m} j_k > \prod_{i=1}^{p} j_i$

Proof: $m > p \Longrightarrow \prod_{l=1}^{m} j_l > \prod_{l=N-p+1}^{N} j_l \Longrightarrow \prod_{k=1}^{m} j_k > \prod_{i=1}^{p} j_i$

$(\because \prod_{l=1}^{\left\lceil \frac{N}{2} \right\rceil} j_l > \prod_{k=\left\lfloor \frac{N+1}{2} \right\rfloor + 1}^{N} j_l \,)$

Q.E.D.

Theorem 4.5: $If \; 1 \le m, p \le N, j_k \in J_k \subseteq J_N, j_i \in J_i \subseteq J_N, j_l \in Order(J_N) \; and \; \prod_{l=1}^{\left\lceil \frac{N}{2} \right\rceil} j_l > \prod_{l=\left\lfloor \frac{N+1}{2} \right\rfloor + 1}^{N} j_l \,, then \; \prod_{k=1}^{m} j_k \ge \prod_{i=1}^{p} j_i \Longrightarrow m \ge p.$

Proof: $Case\ 1$: $\prod_{k=1}^{m} j_k = \prod_{i=1}^{p} j_i$,

$If\ m \neq p, then\ m > p\ or\ m < p \Longrightarrow \prod_{k=1}^{m} j_k > \prod_{i=1}^{p} j_i\ \ or\ \prod_{k=1}^{m} j_k < \prod_{i=1}^{p} j_i$

$(Contradiction)$

$\therefore m = p$

$Case\ 2$: $\prod_{k=1}^{m} j_k > \prod_{i=1}^{p} j_i$,

$If\ m < p, then\ \prod_{k=1}^{m} j_k < \prod_{i=1}^{p} j_i$

$(Contradiction)$

$\therefore m \geq p$

Q.E.D.

Theorem 4.6: $If\ 1 \leq p \leq N, 1 \leq m \leq N-1, j_m \leq \mathrm{X} < j_{m+1}, j_i \in J_N\ and\ j_k, j_m, j_{m+1} \in Order(J_N), then\ \left\| \frac{\log(\prod_{i=1}^{p} j_i)}{\log(\mathrm{X})} - p \right\| \leq \max\left(\left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log(\mathrm{X})} - m \right\|, \left\| \frac{\log(\prod_{k=m+1}^{N} j_k)}{\log(\mathrm{X})} - (N-m) \right\| \right)$.

Proof: $\left\| \frac{\log(\prod_{i=1}^{p} j_i)}{\log(\mathrm{X})} - p \right\| = \left\| \sum_{i=1}^{p} \left(\frac{\log(j_i)}{\log(\mathrm{X})} - 1 \right) \right\|$

$\leq \max\left(\left\| \sum_{\forall j_i \leq \mathrm{X}} \left(\frac{\log(j_i)}{\log(\mathrm{X})} - 1 \right) \right\|, \left\| \sum_{\forall j_i > \mathrm{X}} \left(\frac{\log(j_i)}{\log(\mathrm{X})} - 1 \right) \right\| \right)$

$\leq \max\left(\left\| \sum_{\forall j_k \leq \mathrm{X}} \left(\frac{\log(j_k)}{\log(\mathrm{X})} - 1 \right) \right\|, \left\| \sum_{\forall j_k > \mathrm{X}} \left(\frac{\log(j_k)}{\log(\mathrm{X})} - 1 \right) \right\| \right)$

$= \max\left(\left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log(\mathrm{X})} - m \right\|, \left\| \frac{\log(\prod_{k=m+1}^{N} j_k)}{\log(\mathrm{X})} - (N-m) \right\| \right)$

Q.E.D.

Theorem 4.7: $If\ 1 \leq p \leq N, j_m \leq (\prod_{i=1}^{N} j_i)^{\frac{1}{N}} < j_{m+1},\ j_i \in J_N\ and\ j_k, j_m, j_{m+1} \in Order(J_N), then\ \left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log\left((\prod_{i=1}^{N} j_i)^{\frac{1}{N}} \right)} \right\| = m \Longrightarrow \left\| \frac{\log(\prod_{i=1}^{p} j_i)}{\log\left((\prod_{i=1}^{N} j_i)^{\frac{1}{N}} \right)} \right\| = p.$

Proof: $Let\ \mathrm{X} = (\prod_{k=1}^{N} j_k)^{\frac{1}{N}}\ where\ j_m \le \mathrm{X} < j_{m+1}, then$:

$$N = \frac{\log(\prod_{k=1}^{N} j_k)}{\log\left((\prod_{k=1}^{N} j_k)^{\frac{1}{N}}\right)} \Leftrightarrow N = \frac{\log(\prod_{k=1}^{N} j_k)}{\log(\mathrm{X})} \Leftrightarrow N = \frac{\log(\prod_{k=1}^{m} j_k) + \log(\prod_{k=m+1}^{N} j_k)}{\log(\mathrm{X})}$$

$$\Leftrightarrow m - \frac{\log(\prod_{k=1}^{m} j_k)}{\log(\mathrm{X})} = \frac{\log(\prod_{k=m+1}^{N} j_k)}{\log(\mathrm{X})} - (N-m) \Leftrightarrow \left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log(\mathrm{X})} - m \right\| =$$

$$\left\| \frac{\log(\prod_{k=m+1}^{N} j_k)}{\log(\mathrm{X})} - (N-m) \right\|$$

$$\therefore \max\left(\left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log\left((\prod_{k=1}^{N} j_k)^{\frac{1}{N}}\right)} - m \right\|, \left\| \frac{\log(\prod_{k=m+1}^{N} j_k)}{\log\left((\prod_{k=1}^{N} j_k)^{\frac{1}{N}}\right)} - (N-m) \right\| \right) = \left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log\left((\prod_{k=1}^{N} j_k)^{\frac{1}{N}}\right)} - m \right\|$$

$Therefore,$

$$\left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log\left((\prod_{i=1}^{N} j_i)^{\frac{1}{N}}\right)} \right\| = m \Longrightarrow \left\| \frac{\log(\prod_{k=1}^{m} j_k)}{\log\left((\prod_{k=1}^{N} j_k)^{\frac{1}{N}}\right)} - m \right\| = 0 \Longrightarrow \left\| \frac{\log(\prod_{i=1}^{p} j_i)}{\log\left((\prod_{i=1}^{N} j_i)^{\frac{1}{N}}\right)} - p \right\| = 0 \Longrightarrow$$

$$\left\| \frac{\log\left(\prod_{j=1}^{p} j_i\right)}{\log\left(\left(\prod_{j=1}^{N} j_i\right)^{\frac{1}{N}}\right)} \right\| = p$$

Q.E.D.

Theorem 4.8: $If\ 1 \le k \le m+1 \le N\ and\ j_i, j_k, j_m, j_{m+1} \in I_N, then$

$$\left\| \frac{\log(\prod_{i=1}^{m+1} j_i)}{\log\left((\prod_{i=1}^{N} j_i)^{\frac{1}{N}}\right)} \right\| = m+1 \Longrightarrow \left\| \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{\prod_{i=1}^{N} j_i}{j_k}\right)^{\frac{1}{N-1}}\right)} \right\| = m.$$

Proof: $Let\ P = \prod_{i=1}^{N} j_i \ge 1\ where\ j_i \in J_N, then$:

$$\frac{\log\left(\prod_{i=1}^{m+1} j_i\right)}{\log\left(\prod_{i=1}^{N} j_i\right)^{\frac{1}{N}}} = \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N}}\right)+\frac{1}{N}*\log(j_k)} = \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{N-1}{(N-1)*N}}\right)+\frac{1}{N}*\log(j_k)} =$$

$$\frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\frac{N-1}{N}*\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N}*\log(j_k)} = \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N}*\left(\log(j_k)-\frac{1}{N-1}*\log\left(\frac{P}{j_k}\right)\right)} =$$

$$\frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N}*\left(\frac{N*\log(j_k)-\log(j_k)-\log\left(\frac{P}{j_k}\right)}{N-1}\right)} = \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N}*\left(\frac{\log\left(\frac{{j_k}^N}{P}\right)}{N-1}\right)} =$$

$$\frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)+\log(j_k)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N-1}*\left(\log(j_k)-\log\left(P^{\frac{1}{N}}\right)\right)} = \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N-1}*\left(\log(j_k)-\log\left(P^{\frac{1}{N}}\right)\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)}$$

$Case\ 1: j_k \geq P^{\frac{1}{N}}$,

$$\frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N-1}*\left(\log(j_k)-\log\left(P^{\frac{1}{N}}\right)\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)} \leq \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{i_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)}$$

$$\Longrightarrow m+1 \leq \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)} < m+1.5$$

$$\Longrightarrow m-0.5 < \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)} < m+0.5\ (\because 1 \leq \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)} < 1.5)$$

$Case\ 2: j_k < P^{\frac{1}{N}}$,

$$\frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)+\frac{1}{N-1}*\left(\log(j_k)-\log\left(P^{\frac{1}{N}}\right)\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)} > \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)}$$

$$\Longrightarrow m + 0.5 < \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)} + \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)} < m + 1$$

$$\Longrightarrow m - 0.5 < \frac{\log\left(\frac{\prod_{i=1}^{m+1} j_i}{j_k}\right)}{\log\left(\left(\frac{P}{j_k}\right)^{\frac{1}{N-1}}\right)} < m + 0.5 \ (\because 0.5 < \frac{\log(j_k)}{\log\left(P^{\frac{1}{N}}\right)} < 1)$$

Q.E.D.

```
function GENERATE-J-SEQUENCE (N) returns a sorted J-Sequence with size N
        input: N, a positive integer
        J_N ← {1}
        current-size ← 1
        while current-size < N
                try
                        J_N ← Generate-Next-J-Sequence (J_N)
                catch
                        overflow-exception
                current-size ← current-size + 1
return J_N
```

Figure 4-2 Pseudocode of the algorithm that generates a sorted J_N.

```
function GENERATE-NEXT-J-SEQUENCE (J_N) returns a sorted J-Sequence with size N+1
    input: J_N, a sorted J-Sequence with size N
    current ← the last element of J_N
    while true
        current ←Get-Next-Number(current, J_N)
        J_N+1 ← append current to the end of J_N
        if the product of J_N+1 is overflow
            throw overflow-exception
        else if Verify-Modulus-Rule(J_N+1) = false
            J_N ← remove the first element of J_N+1
        else if Verify-Product-Rule(J_N+1) = false
            J_N ← remove the first element of J_N+1
        else if Verify-Size-Rule(J_N+1) = false
            J_N ← remove the first element of J_N+1
        else
            break
    return J_N+1
```

Figure 4-3 Pseudocode of the GENERATE-NEXT-J-SEQUENCE algorithm.

```
function GET-NEXT-NUMBER (current, J_N) returns an integer
    input: current, an integer
           J_N, a sorted J-Sequence with size N
    next ← current + 1
    product ← the product of J_N
    while product % next = 0
        next ← next + 1
    return next
```

Figure 4-4 Pseudocode of the GET-NEXT-NUMBER algorithm.

```
function VERIFY-MODULUS-RULE (J_N) returns a boolean
        input: J_N, a sorted J-Sequence with size N
        i ← 0
        size ← the size of J_N
        product ← the product of J_N
        while i < size
                i ← i + 1
                current ← the i-th element of J_N
                if (product / current) % current = 0
                        return false
        return true
```

Figure 4-5 Pseudocode of the VERIFY-MODULUS-RULE algorithm.

```
function VERIFY-PRODUCT-RULE (J_N) returns a boolean
        input: J_N, a sorted J-Sequence with size N
        size ← the size of J_N
        half-size ← round(size / 2)
        left-product ← the product of the first (half-size) elements of J_N
        right-product ← the product of the last (half-size – 1) elements of J_N
        if left-product > right-product
                return true
        else
return false
function GENERATE-J-SEQUENCE (N) returns a sorted J-Sequence with size N
        input: N, a positive integer
```

Figure 4-6 Pseudocode of the VERIFY-PRODUCT-RULE algorithm.

J_N

```
function VERIFY-SIZE-RULE (J_N) returns a boolean
    input: J_N, a sorted J-Sequence with size N
    mean ← geometric mean of J_N
    J_left ← the set of elements of J_N that smaller than mean
    left-size ← the size of J_left
    left-product ← the product of J_left
    if round(log(left-product)/log(mean)) ≠ left-size
        return false
    J_right ← the set of elements of J_N that larger than mean
    right-size ← the size of J_right
    right-product ← the product of J_right
    if round(log(right-product)/log(mean)) ≠ right-size
        return false
    return true
```

Figure 4-7 Pseudocode of the VERIFY-SIZE-RULE algorithm.

4.4 Creating J-Sequence with Prime Numbers

The Generate-J-Sequence Algorithm guarantees to generate a J-Sequence only if it can find one sequence that assures the Modulus-Rule, Product-Rule and Size-Rule. It is not sufficient to prove that such a sequence exists for any positive integer N, even though I tried to program the algorithm and it can produce a J-Sequence for every $N <$ 112 until the program overflow[8]. Hence, a theoretical approach is introduced to ensure the logical completeness of my work. In fact, a J-Sequence with size N can be created by N arbitrary unique prime numbers and another prime number that is greater than their

[8]The program is written in Python and it can produce a J-Sequence up to size 111, which is {518, 519, 521, 522, 523, 524, 526, 527, 529, 530, 531, 534, 535, 536, 539, 540, 541, 542, 543, 545, 546, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 562, 564, 565, 567, 568, 569, 570, 571, 572, 573, 574, 575, 577, 578, 579, 580, 581, 582, 584, 585, 586, 587, 588, 589, 590, 591, 593, 594, 595, 596, 598, 599, 600, 601, 602, 603, 606, 607, 608, 609, 611, 612, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652} with P_{111} = 2.48127056919e+307 and geometric mean = 587.924269882.

product. In this section, let's assume the N arbitrary unique prime numbers as { X_1, X_2, ..., X_{N-1}, X_N } with product P and the other prime number as Y. By using these prime numbers, we can form a sequence of N unique numbers with the following definition:

$$S_N = \{s_1, s_2, \dots, s_i, \dots, s_{N-1}, s_N\} \; where \; Y > P = \prod_{i=1}^{N} X_i > 1 \; and \; s_i = X_i Y$$

This special sequence S_N is actually a J_N because it satisfies the conditions of the Modulus-Rule, Product-Rule and Size-Rule. We will prove this by the following theorems in this section.

Theorem 4.9: $If \; 1 \le k \le N \; and \; s_j, s_k \in S_N, then \; \frac{\prod_{i=1}^{N} s_i}{s_k} \; mod \; s_k \neq 0.$

Proof:$If \; \frac{\prod_{i=1}^{N} s_j}{s_k} \; mod \; s_k = 0, then \; \frac{\prod_{i=1}^{N} s_i}{s_k * s_k} = \frac{\prod_{i=1}^{N} X_j Y}{X_k Y * X_k Y} = D \; where \; D \; is \; an \; integer$

$\frac{\prod_{i=1}^{N} s_j}{s_k * s_k} = \frac{\prod_{i=1}^{N} X_i Y}{X_k Y * X_k Y} = Y^{N-2} * \frac{X_1 * \dots * X_{k-1} * X_{k+1} * \dots * X_N}{X_k}$

$(Contradition, \because Y^{N-2} * \frac{X_1 * \dots * X_{k-1} * X_{k+1} * \dots * X_N}{X_k} \; is \; not \; an \; integer)$

$Therefore, \frac{\prod_{i=1}^{N} s_i}{s_k} \; mod \; s_k \neq 0$

Q.E.D.

Theorem 4.10: $If \; s_k \in S_k = Order(S_N), then \; \prod_{k=1}^{\lceil \frac{N}{2} \rceil} s_k > \prod_{k=\lfloor \frac{N+1}{2} \rfloor + 1}^{N} s_k.$

Proof: $\frac{\prod_{k=1}^{\lceil \frac{N}{2} \rceil} s_k}{\prod_{k=\lfloor \frac{N+1}{2} \rfloor + 1}^{N} s_k} = \frac{\prod_{k=1}^{\lceil \frac{N}{2} \rceil} X_k Y}{\prod_{k=\lfloor \frac{N+1}{2} \rfloor + 1}^{N} X_k Y} = \frac{Y * \prod_{k=1}^{\lceil \frac{N}{2} \rceil} X_k}{\prod_{k=\lfloor \frac{N+1}{2} \rfloor + 1}^{N} X_k} > \prod_{k=1}^{\lceil \frac{N}{2} \rceil} X_k > 1$

$(\because Y > P = \prod_{k=1}^{N} X_k)$

Q.E.D.

Theorem 4.11: $If\ s_k, s_m, s_{m+1} \in S_k = Order(S_N)\ and\ s_m \leq (\prod_{i=1}^{N} s_i)^{\frac{1}{N}} < s_{m+1}, then\ \left\| \frac{\log(\prod_{k=1}^{m} s_k)}{\log\left((\prod_{i=1}^{N} s_i)^{\frac{1}{N}}\right)} \right\| = m.$

Proof: $m + 0.5 - \frac{\log(\prod_{k=1}^{m} s_k)}{\log\left((\prod_{i=1}^{N} s_i)^{\frac{1}{N}}\right)} = m + 0.5 - \frac{\log(\prod_{k=1}^{m} X_k Y)}{\log\left((\prod_{k=1}^{N} X_k Y)^{\frac{1}{N}}\right)}$

$= m + 0.5 - \frac{\log(Y^m * \prod_{k=1}^{m} X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = m + 0.5 - \frac{m*\log(Y) + \sum_{k=1}^{m} \log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$

$= \frac{(m+0.5)*\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right) - m*\log(Y) - \sum_{k=1}^{m} \log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$

$= \frac{(m+0.5)*\log(Y) + (m+0.5)*\sum_{k=1}^{N} \log\left(X_k{}^{\frac{1}{N}}\right) - m*\log(Y) - \sum_{k=1}^{m} \log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$

$= \frac{0.5*\log(Y) + (m+0.5)*\sum_{k=1}^{N} \log\left(X_k{}^{\frac{1}{N}}\right) - \sum_{k=1}^{m} \log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = \frac{\log\left(Y^{\frac{1}{2}}\right) + \sum_{k=1}^{N} \log\left(X_k{}^{\frac{2m+1}{2N}}\right) - \sum_{k=1}^{m} \log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$

$> \frac{\log\left(P^{\frac{1}{2}}\right) + \sum_{k=1}^{N} \log\left(X_k{}^{\frac{2m+1}{2N}}\right) - \sum_{k=1}^{m} \log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = \frac{\sum_{k=1}^{N} \log\left(X_k{}^{\frac{N+2m+1}{2N}}\right) - 2N*\sum_{k=1}^{m} \log\left(X_k{}^{\frac{1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$

$= \frac{(N+2m+1)*\sum_{k=1}^{N} \log\left(X_k{}^{\frac{1}{2N}}\right) - 2N*\sum_{k=1}^{m} \log\left(X_k{}^{\frac{1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = \frac{(N+2m+1)*\log\left(P^{\frac{1}{2N}}\right) - 2N*\sum_{k=1}^{m} \log\left(X_k{}^{\frac{1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$

$> \frac{(N+2m+1)*\log\left(P^{\frac{1}{2N}}\right) - 2*\sum_{k=1}^{m} \log\left(P^{\frac{1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = \frac{(N+m+1)*\log\left(P^{\frac{1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} > 0$

$\therefore \frac{\log(\prod_{k=1}^{m} s_k)}{\log\left((\prod_{i=1}^{N} s_i)^{\frac{1}{N}}\right)} < m + 0.5$

$$\frac{\log\left(\prod_{k=1}^{m} s_k\right)}{\log\left(\left(\prod_{i=1}^{N} s_i\right)^{\frac{1}{N}}\right)} - (m-0.5) = \frac{\log\left(\prod_{k=1}^{m} X_k Y\right)}{\log\left(\left(\prod_{k=1}^{N} X_k Y\right)^{\frac{1}{N}}\right)} - (m-0.5)$$

$$= \frac{\log\left(Y^m * \prod_{k=1}^{m} X_k\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} - (m-0.5) = \frac{m*\log(Y) + \sum_{k=1}^{m}\log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} - (m-0.5)$$

$$= \frac{m*\log(Y) + \sum_{k=1}^{m}\log(X_k) - (m-0.5)*\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$$

$$= \frac{0.5*\log(Y) + \sum_{k=1}^{m}\log(X_k) - (m-0.5)*\sum_{k=1}^{N}\log\left(X_k{}^{\frac{1}{N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = \frac{\log\left(Y^{\frac{1}{2}}\right) + \sum_{k=1}^{m}\log(X_k) - \sum_{k=1}^{N}\log\left(X_k{}^{\frac{2m-1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$$

$$> \frac{\log\left(P^{\frac{1}{2}}\right) + \sum_{k=1}^{m}\log(X_k) - \sum_{k=1}^{N}\log\left(X_k{}^{\frac{2m-1}{2N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)} = \frac{\log\left(P^{\frac{N+1}{2N}}\right) + \sum_{k=1}^{m}\log(X_k) - \sum_{k=1}^{N}\log\left(X_k{}^{\frac{m}{N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$$

$$= \frac{\log\left(P^{\frac{N+1}{2N}}\right) + \frac{N-m}{N} * \sum_{k=1}^{m}\log(X_k) - \frac{m}{N} * \sum_{k=m+1}^{N}\log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$$

$$> \frac{\log\left(P^{\frac{N+1}{2N}}\right) + \frac{N-m}{N} * \sum_{k=1}^{m}\log(X_k) - \frac{m}{N} * \sum_{k=1}^{N-m+1}\log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$$

$$> \frac{\log\left(P^{\frac{N+1}{2N}}\right) + \frac{N-m}{N} * \sum_{k=1}^{m}\log(X_k) - \frac{m}{N} * \sum_{k=1}^{N-m+1}\log\left(P^{\frac{1}{N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k{}^{\frac{1}{N}}\right)}$$

$$= \frac{\log\left(P^{\frac{N+1}{2}}\right) + (N-m) * \sum_{k=1}^{m}\log(X_k) - m * \sum_{k=1}^{N-m+1}\log\left(P^{\frac{1}{N}}\right)}{\log\left(Y * \prod_{k=1}^{N} X_k\right)}$$

$$= \frac{(N^2+N-2Nm+2m^2-2m) * \log\left(P^{\frac{1}{2N}}\right) + (N-m) * \sum_{k=1}^{m}\log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k\right)}$$

$$= \frac{\left((N-m)^2+(N-1)+(m-1)^2\right) * \log\left(P^{\frac{1}{2N}}\right) + (N-m) * \sum_{k=1}^{m}\log(X_k)}{\log\left(Y * \prod_{k=1}^{N} X_k\right)} > 0$$

$$\therefore \frac{\log\left(\prod_{k=1}^{m} s_k\right)}{\log\left(\left(\prod_{i=1}^{N} s_i\right)^{\frac{1}{N}}\right)} > m - 0.5$$

Q.E.D.

4.5 J-Union, J-Intersection and J-Comparison

When a J-Sequence is used alone to represent one relation, it simplifies the coding effort by making the lookup and counting operations based on a constant number of basic math functions but independent to the number of involved objects. Furthermore, it will be more powerful when a J-Sequence is used as a whole J-Sequence family. There are three main kinds of information that we can get from the family members, which are (1) J-Comparison – the comparison information of multiple members, (2) J-Union – the union information of multiple members and (3) J-Intersection – the intersection information of multiple members.

4.5.1 J-Comparison

When we want to compare the relationship-size of multiple relations under the same J-Sequence family, we can directly compare the values of their J-Products. In another word, the order of the values of some J-Products, which represent the relationship information of some relations, rooted from the same J-Sequence family is also the order of their relationship-size (given by Property 3 of J-Sequence.)

For example, if we want to select a location to build the first wireless server from four qualified spots and our purpose is figuring out a spot that covers the highest number

of cities among the ten cities in the province (illustrated as Figure 4-8,) then a J_{10} (e.g. {518, 519, 521, 522, 523, 523, 524, 526, 527, 529, 530}, which is a subset of J_{111}) can be applied to represent the relation map for solving this problem. By ordering the J-Products of all the spot scoping relations (i.e. *P*(Spot *A*) = 519*521 = 130269, *P*(Spot *B*) = 75817566360, *P*(Spot *C*) = 143039484 and *P*(Spot *D*) = 274052,) then we know that Spot *B* is the best choice, followed by Spot *C*, Spot *D* and Spot *A*. If we expand the question as finding the best server spot to serve the most population where every city has one million people except City1 has three million and City3 has two million, then we can expand the same J_{10} into one of its superset J_{13} in the family, e.g. {518, 519, 521, 522, 523, 523, 524, 526, 527, 529, 530, 531, 534, 535} and reassign the representation of each cities according to the population ratio. As a result, the representations of City1 and City3 will become {518*531*534} and {521*535}, and the others remain the same. By Property 3 and Property 5 of J-Sequence, the logic of J-Comparison still applies. If we re-compute the J-Products and reorder them, the new selection order will become as Spot *B*, Spot C, Spot *A* and Spot *D*.

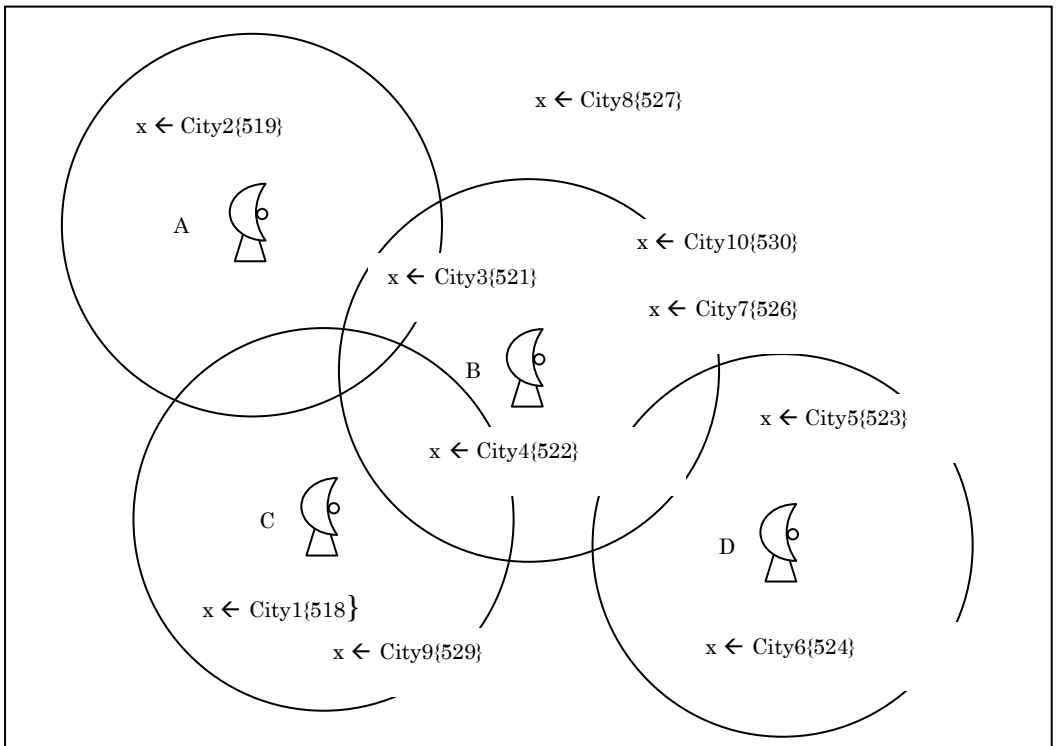

Figure 4-8 Illustration of a province with ten cities and four selections of first server location. The server coverage scopes from spot A, B, C and D are shown as the circles centered by the spots.

4.5.2 J-Union

When we want to combine multiple relations of the same family into a parent relation and preserve the information of the children, we can calculate the parent's J-Product as the least common multiple (*LCM*) of the children's J-Products. The conceptual meaning of this operation is to determine the union superset of multiple J-Sequences. The resulting superset is a J-Sequence that keeps all of the features of its children, so that we can retrieve each piece of the combined information from the parent's J-Product directly.

For example, if the province finally has two servers built at Spot *B* and Spot *C*, then we can determine their coverage (in number of cities) by apply the counting operation on their J-Products directly as *round*(*log*(*LCM*(75817566360, 143039484))/*log*(587.924269882)) = 6. By Property 4 and 5 of J-Sequence, the counting operation returns the number of individual pieces of information contained by the J-Products (referred to as J-Size.)

4.5.3 J-Intersection

Similar to J-Union, J-Intersection is the intersection subset of multiple J-Sequences of the same family. The subset is a child J-Sequence that contains the common features of its parents. The J-Product of the child can be found as the highest common factor (*HCF*) of the parents' J-Products. Therefore, the common information of the parent J-Sequences can be retrieved from the J-Product of their child.

For example, if we want to determine whether City4 is supported by both the servers at Spot B and Spot C from their J-Products, then we can imply it as a mathematic question, i.e. "*HCF*(75817566360, 143039484)%522 == 0?", and its answer is Yes. To develop the question further, we can verify whether both of these two servers support City4 and City7 by a mathematic equation, i.e. "*HCF*(75817566360, 143039484)%*LCM*(522, 526) == 0?" (or simply "*HCF*(75817566360, 143039484)%(522*526) == 0?" if 522 and 526 are known as P_1,) because Property 2 and Property 5 of J-Sequence promises to check the complete information on a lookup operation (referred to as J-Lookup in this paper). Overall, the properties of the J-Sequence establish a common space of any J-Sequence family such that we can

manipulate its information and apply the J-Lookup, J-Size, J-Union, J-Intersection and J-Comparison on its members from their J-Products straightforwardly. Therefore, we can narrow down the question into the scope of a city or expand it to the country level in a similar way with a shared space.

Chapter 5

COMPRESSING UNREALIZED TRAINING DATA SETS[9]

In chapter 4, we discussed the way to represent object relations by applying J-Sequence. In this chapter, we are going to explain (1) how we can numerate the realized training samples by using J-Sequence representation, (2) how we can data compress the numerated samples and (3) some other research details related to this representation method.

5.1 Numerating Unrealized Training Samples

To represent the unrealized training samples, we need a J-Sequence with size N where N is larger or equal to the total number, says M, of possible attribute values in the sample domain, which means:

$$N \geq M = \sum_{\forall a \in attribute\ domain\ of\ the\ samples} \sum_{\forall k \in attribute\ values\ of\ a} 1$$

If we take the samples in Table 2-10 to Table 2-12 as an example, then M is equal to $number\ of\ attribute\ values\ of\ Outlook$ + $number\ of\ attribute\ values\ of\ Humidity$ + $number\ of\ attribute\ values\ of\ Wind$ + $number\ of\ attribute\ values\ of\ Play = 4 + 2 + 3 + 2 = 11$.

By mapping each attribute value k_i (where $1 \leq i \leq M$) with an individual element $j \in J_N$, each data set t in the unrealized training samples can be represented as a

[9] For someone who references any content of this chapter, please put the QR code shown in the acknowledgement page in his / her paper.

number, which is the J-Product of the mapped numbers associating with the attribute values of t. For example, if we use a J_{12} = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39} to numerate the attribute values of the data sets in Table 2-11, Table 2-12 and Table 2-13, then a possible mapping of the sample domain could be {16 → $Outlook = Dummy1$, 17 → $Outlook = Sunny$, 19 → $Outlook = Overcast$, 23 → $Outlook = Rain$, 25 → $Humidity = High$, 27 → $Humidity = Normal$, 28 → $Wind = Dummy2$, 29 → $Wind = Weak$, 31 → $Wind = Strong$, 33 → $Play = Yes$, 37 → $Play = No$}. Hence, the J-Product of all samples can be found (such as the J-Product of Sample#1 < $Outlook = Dummy1$, $Humidity = High$, $Wind = Dummy2$, $Play = Yes$ > is $16 * 25 * 28 * 33 = 369600$ and the resulting tables of Table 2-11, Table 2-12 and Table 2-13 will become as Table 5-1, Table 5-2 and Table 5-3 respectively.

According to the five properties of J-Sequence, we can perform the following operations straightforwardly: (1) determining the frequency of any data set *t* in the sample tables (the details will be provided in Section 5-2) for data mining or getting the statistical information, (2) determining the length of any data set t in the sample tables or the sample domain (which is useful for some data mining methods that counts on the length of t; however, we will not discuss in this dissertation because this topic is out of our research focus) and (3) switching the mapping keys for security purposes (the details will be provided in Section 5.4.)

Sample#	J-Product
1	369600
2	414400

3	382800
4	429200
5	409200
6	458800
7	399168
8	447552
9	413424
10	463536
11	441936
12	495504
13	392700
14	440300
15	406725
16	456025
17	434775
18	487475
19	424116
20	475524
21	439263
22	492507
23	469557
24	526473
25	438900
26	492100

27	454575
28	509675
29	485925
30	544825
31	474012
32	531468
33	490941
34	550449
35	524799
36	588411
37	531300
38	595700
39	550275
40	616975
41	588225
42	659525
43	573804
44	643356
45	594297
46	666333
47	635283
48	712287

Table 5-17 A universal set J_N in Table 2-11 after mapping to J_N = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39}.

Sample#	J-Product
1	369600
2	414400
3	382800
4	429200
5	409200
6	458800
7	399168
8	447552
9	413424
10	463536
11	441936
12	495504
13	392700
14	440300

Table 5-18 Training data sets T' in Table 2-12 after mapping to J_{12} = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39}.

Sample#	J-Product
15	406725
17	434775
19	424116

20	475524
22	492507
24	526473
25	438900
26	492100
28	509675
30	544825
31	474012
32	531468
34	550449
36	588411
37	531300
38	595700
40	616975
41	588225
43	573804
44	643356
46	666333
47	635283
1	369600
2	414400
3	382800
4	429200
5	409200

6	458800
7	399168
8	447552
9	413424
10	463536
11	441936
12	495504
13	392700
14	440300
15	406725
17	434775
18	487475
19	424116
20	475524
21	439263
22	492507
23	469557
24	526473
25	438900
26	492100
27	454575
28	509675
29	485925
30	544825

31	474012
32	531468
33	490941
34	550449
35	524799
36	588411
37	531300
38	595700
39	550275
40	616975
41	588225
42	659525
43	573804
44	643356
46	666333
47	635283
48	712287

Table 5-19 Perturbing data sets T^P in Table 2-13 after mapping to J_{12} = {16, 17, 19, 23, 25, 27, 28, 29, 31, 33, 37, 39}.

5.2 Data Compression, Storage Complexity and Query Time Complexity

A focus of our research is to reduce the storage requirement of the training samples by data compression. In this section, we will discuss two different levels of

compression: (1) column-wise data compression by compressing each data set t of the samples as a number and (2) row-wise compression by further applying other data compression methods to the whole sample table.

5.2.1 Data Compression by J-Sampling

To analyze the storage requirement for the sample table T, we need to study the required space for each data set t in T. Any possible attribute value of an attribute $a \in A$ of t can be a string (which is considered as an array of characters) and the storage requirement for any attribute value $k \in K$ should be large enough to store the possible option with the longest string length; even someone may enumerate the choices of an attribute as a list of number, such as enumerating the options of $Outlook$ with $\{ Dummy1, Sunny, Overcast, Rain \}$ as $\{ 1, 2, 3, 4 \}$, the storage size of t is still dependent on the number of attributes of t (denoted as $|A|$). As a result, the storage complexity of t is:

$O(|A|)$ if K does not contain any string (or other flexible-size) members,

or

$O(|A| * |K_{\max}|)$ otherwise,

where $|K_{\max}|$ is the storage required to store the member with the largest size.

By representing each data set t in T as a J-Product (referred to as J-Sampling hereafter), we are converting it from a tuple of attribute values into a number with fixed size[10]; hence, the storage complexity of t becomes $O(1)$.

Someone may argue that the J-Sequence representation requires extra storage for the mapping table and its storage complexity is $O(|T^U|)$; however, this fact could be ignored when we examine the storage requirement of the entire sample table because (1) we also disregard the storage requirement (which is possibly $O(|T^U|)$) of the enumerating method mentioned above and (2) the size of T^U is much smaller than the size of the samples (given by the research assumptions in Chapter 1.)

Please be aware the field Sample# in all the examples used in this research is added for reading convenience and has no actual usage; therefore, this field will be removed for the rest of the examples after this section.

5.2.2 Data Compressing the J-Samples

From the storage point of view, the sample table will become a list of numbers (in which each number p represents a data set) after J-Sampling the unrealized samples. These numbers p are members of the table T^U after J-Sampling, such that most of them are duplicated when $|T_S|$ is much larger than $|T^U|$ because $|(T'+T^P)|$ is larger than

[10] From my experiment, a float in Python can be used to generate a J-Sequence up to the size of 111. For most data mining classification cases, the number of possible attribute values of an attribute is small (from the examples used in this research, an attribute with the most options has 4 possible attribute values, which means a float can be used to represent 27 attributes) so that one floating number is needed to store for a data set normally. Even though one number may not be enough to store a data set in some rare cases, limited number of floats could cover the usage in general and their total size is still considered as fixed and small. In this dissertation, we assume every data set can be represented by one floating number for all study cases.

$|T_S|$ and all of them are sharing the same space T^U. If we sort the unrealized samples in order, then all the repeated numbers will be aligned together. Table 5-4 shows the sorted samples from Table 5-3. After sorting, data compression can be applied to the numerated samples (referred to as J-Samples) effectively.

Run-length encoding (RLE[19]) matches perfectly with the pattern of the J-Samples in sorted form and it can achieve a high compression rate. In general, RLE encodes a data content into a sequence of tuples that store the data value and the count as a storage unit (denoted as $<value, count>$.) For example, the data content [1, 2, 2, 1, 1, 1, 1, 0, 0, 0] will be compressed as [1, 1, 2, 2, 1, 4, 0, 3]. When the compressed content needs to be decompressed, we decode two pieces of data as a $<value, count>$ unit at a time for reconstructing the original information. From the previous example, we decode <1, 1> as [1], <2, 2> as [2, 2], <1, 4> as [1, 1, 1, 1] and <0, 3> as [0, 0, 0] to form the original content [1, 2, 2, 1, 1, 1, 1, 0, 0, 0]. RLE provides excellent compression rate when the repeated rate of the original content is high and the repeated data are consecutive to the others, which is exactly the case of the sorted J-Samples. If we compress the samples in Table 5-4, then the table will be encoded as [369600, 1, 382800, 1, 392700, 1, 399168, 1, 406725, 2, 409200, 1, 413424, 1, 414400, 1, 424116, 2, 429200, 1, 434775, 2, 438900, 2, 439263, 1, 440300, 1, 441936, 1, 447552, 1, 454575, 1, 458800, 1, 463536, 1, 469557, 1, 474012, 2, 475524, 2, 485925, 1, 487475, 1, 490941, 1, 492100, 2, 492507, 2, 495504, 1, 509675, 2, 524799, 1, 526473, 2, 531300, 2, 531468, 2, 544825, 2, 550275, 1, 550449, 2, 573804, 2, 588225, 2, 588411, 2, 595700, 2, 616975, 2, 635283, 2, 643356, 2, 659525, 1, 666333, 2, 712287, 1].

From the storage unit $< value, count >$, if we treat the storage requirement of *value* is the same as that of *count*, then the above RLE compression example does not reduce the storage space because the average count per storage unit is low. Indeed, RLE functions only when the average count per storage unit is greater than two. The higher the average count, the better the compression rate. As the J-Samples are sorted and bounded within the domain of T^U, we can conclude that the maximum storage requirement equals $|T^U|$*(size of *value* + size of *count*). Since (size of *value* + size of *count*) is a constant order space, storage complexity of the J-Samples after RLE compression is $O(|T^U|)$ and average count per storage unit is larger or equal to:

$$\frac{|(T'+T^P)|}{|T^U|} \geq \frac{|T_S|}{|T^U|}$$

From our research scope, $\frac{|T_S|}{|T^U|}$ is a very large number much greater than 2, which induces high compression rate. In fact, $|T^U|$ is the size of the sample domain and it is a known constant for discrete-value samples, so that the storage complexity is $O(1)$.

J-Product		J-Product (Continue)
369600		492507
382800		495504
392700		509675
399168		509675
406725		524799
406725		526473

409200	526473
413424	531300
414400	531300
424116	531468
424116	531468
429200	544825
434775	544825
434775	550275
438900	550449
438900	550449
439263	573804
440300	573804
441936	588225
447552	588225
454575	588411
458800	588411
463536	595700
469557	595700
474012	616975
474012	616975
475524	635283
475524	635283
485925	643356
487475	643356

490941	659525
492100	666333
492100	666333
492507	712287

Table 5-20 Perturbing data sets T^P in Table 5-3 after sorting (the right column is the continuity of the left).

5.2.3 Statistical Information Retrieval

In addition to storage compression, the run-length encoded J-Samples provide faster statistical query result (such as determining the total counts or average of some data sets) In particular, the counting statistics of each data set in the unrealized training samples is the most crucial piece of information that we need for the data mining process.

When we look into the statistical query process, we can find the J-Sequence numeration method contributes to combine multiple sub-query operations into one simple query. For example, if we want to find the counts of the data set $< Outlook = Rain, Play = Yes >$ in the samples in Table 2-12 (let's name the table as Tp), then we may query with a SQL statement as "SELECT COUNT(*) FROM Tp WHERE Outlook = 'Rain' and Play = 'Yes'". In the internal database query system, the query may be broken down into the following sub-processes:

1) Scanning for the data sets that pass the condition check of Outlook = 'Rain' in Tp.
2) Scanning for the data sets that pass the condition check of Play = 'Yes' in Tp.

3) Determining the intersection set of the results in sub-process 1 and sub-process 2.
4) Finding the count / size of the resulting set from sub-process 3.

Different database systems may develop query plans differently and the final query plan varies according to data statistics, so the above sub-processes is only a possible plan option but not necessarily the actual plan; however, regardless of which final query plan is chosen, the query process will involve some combinations of multiple operations such as scanning, sorting, merging and hashing. The time complexity of each operation is between range $O(N)$ to $O(N^2)$ and that of the combinations of two operations can be ranged from the sum of their time complexities to the product of the two.

On the other hand, if we want to find the counts of the data set $< Outlook = Rain, Play = Yes >$ (which implies the value of J-Product = 23*33 = 759) in the samples in Table 5-3, then we would query as "SELECT COUNT(*) FROM Tp WHERE J-Product % 759 = 0". This statement will result in a single scan operation with a counter and its time complexity is $O(N)$.

The RLE method aggregates the information of the sorted samples in the $< value, count >$ format, so that it groups the samples into at most $|T^U|$ units for the scan operation and further optimizes the query speed. Thus, the overall time complexity of the statistical query is $O(|T^U|)$ and it is much smaller than $O(N)$ because $N \geq |T_S| > |T^U|$. Again, $|T^U|$ is a known constant for discrete-value samples such that the time complexity is $O(1)$.

5.3 Column-based Database Architecture

In Section 5.1 and Section 5.2, we introduced an effective way to compress the unrealized training samples and it should resolve the storage concern from Fong and Weber's previous research and improve statistical query efficiency; however, this method may downgrade the privacy protection power of Data Unrealization – if each piece of data loss are highly compressed data and containing a large amount of information.

Our research stands on the following assumption: each privacy attack involves a minor subset of the unrealized training samples because the number of data sets lost from the attack (denoted as T_{lost}) is much smaller than the size of the unrealized samples $(T'+T^{P})$. This claim is shaky when a privacy attack happens on some extremely compressed data table. Consequently, (1) we should not explicitly keep the sample table in the data compressed form; instead, (2) we should compress the data properly in the internal system and isolate the compressed data from any external access. Let's interpret these two highlights from the standpoint of a database management system (DBMS[20, 30]) – the former implies: we should not perform data compression on the information from the samples and store them as a sample table structured in the data compressed form (Table 5-5 is shown as an example) and the later implies: we should select an appropriate DBMS that performs data compression on its internal storage (which is accessible only by the internal system and protected from other accesses) and the selected data compression method should fit our usage. To avoid diverting from our research focus and diving into the very details of DBMS, we assume some DBMS architectures and techniques can be applied to isolate the client view of the data from the internal storage,

so that a database client can only access the decompressed information while the database server stores the compressed data content internally.

A column-oriented DBMS[21, 28, 29] is a good choice that provides the architecture to achieve our purpose discussed above. Different from the traditional row store databases[22, 27], column-based databases store data tables as sections of columns of data such that similar data contents are aggregated in storage. Therefore, they provide the following benefits over other traditional database systems:

1) To achieve best compression, the column with the lowest cardinality will be used as the first sorting keys of the data table normally. As column data is in a uniform type and similar data contents are adjacent to the others, data compression (RLE is a popular one to use, or some other data compression methods with similar effectiveness will be used) will be applied to compress the internal storage.
2) Computation is efficient on aggregation queries over many rows but for a small number of columns.
3) Updates of values on a column are efficient when the changes are applied to all rows.

J-Sequence numeration converts the unrealized samples into a data table containing one single column with many replicated values; furthermore, the data mining process requires the counts of a data set (which has one single column) from all rows of the training samples. If we use a column-oriented DBMS to store the J-Sampled $(T'+T^P)$, then it is sufficient to assert the first two advantages – high compression rate and fast statistical query performance – of this database architecture; hence, the objectives mentioned in

Section 5.2.2 and Section 5.2.3 can be accomplished. Please be aware that advantage three will be in effect for the PROTECTION-KEY-UPDATE function discussed in Section 5.4 for the same reasons.

J-Product	Count
406725	1888889
434775	20000
424116	10000

Table 5-21 A sample table in the form of $< value, count >$.

5.4 Second-level Protection and Protection-Key Updates

J-Sampling provides a second-level privacy protection on the unrealized samples that have been defended by the existing Data Unrealization method. We can view it as a cryptographic method similar to (Anti)monotone Framework but designed for samples with discrete-value attributes. From this point of view, the J-Sequence mapping function acts as the encryption function (and also the decryption function) and the set of numbers in the J-Sequence used to numerate the training samples is the set of encryption keys (and also the decryption keys.) Thus, this method inherits most pros and cons of (Anti)monotone Framework.

J-Sampling preserves both the privacy and utility of the samples; furthermore, it can be used to shield privacy information of the data mining outcomes – because we

don't need to decrypt the samples during the data mining process and the outcomes can be kept in the decrypted form without affecting the performance at any point[11]. If we redo the exercise in Section 3.2.2 with J-Sequence map used in Section 5-1, then the data mining output will be resulted as Table 5-6, which is the same as the result in Table 3-4 after decryption. The data mining processor only needs the mapping keys to mine the results; therefore, the mapping function can be securely kept away from unauthorized parties.

(Anti)monotone Framework does not support samples with discrete-value attributes, but this restriction has been overcome by J-Sampling. In addition, the sample domain is hidden because a data set is no longer a tuple of some meaningful attributes with a finite number of attribute values, but a positive number without a maximum boundary. However, security concern of the decryption functions and keys still holds. Fortunately, even thought the decryption functions and keys may be at risk in some privacy attacking cases, the decrypted information of samples T in T_{lost} is still fortified by Data Unrealization, which has been proven to be effective when the sample size is large in previous research.

A valid privacy attack needs both the J-Sequence map and most data sets of the J-Samples; otherwise, privacy concern does not happen because either T_{lost} cannot be decrypted or T_{lost} is not related to the original samples. For most attacking cases, administrators get alerted when T_{lost} remains a minor subset of T (Chapter 1 indicates that $\frac{|T_{lost}|}{|T|}$ is always less than 15%.) If security updates can take place on the J-Samples

[11] In fact, Section 5.2 proves that the computing performance will be faster and the storage requirement will be smaller.

and the mapping keys when Data Unrealization persists, then the private information in both T and T_{lost} should be considered to be safe because information in T_{lost} (including information of the mapping functions and keys) becomes irrelevant to information in T, which means T_{lost} is totally useless and meaningless for the privacy attacker. To perform the security updates, we can call function PROTECTION-KEY-UPDATE in Figure 5-1 with T, J_N and another J-Sequence J_M where J_N is a strict subset of J_M and J_N is the set of all mapping keys. According to the research in Section 5.3, the SQL update query "**UPDATE TABLE** *T* **SET** J-Product = (J-Product * *m* / *n*) **WHERE** (J-Product % *n* = 0)" in the security update function has high performance.

J-Product	Supports
17	5
23	5
25	7
27	7
31	6
29	8
33	9
37	5
891	6
957	6

Table 5-22 The resulting frequent itemset table T_O by applying the APRIOR' function with encrypted samples.

```
function PROTECTION-KEY-UPDATE (T, J_N, J_M) returns a boolean
    input: T, a sample table numerated by J_N
           J_N, a J-Sequence with size N
           J_M, a J-Sequence with size M
    if J_N is not a strict subset of J_M
        return false
    m_set ← remove J_N from J_M
    n_set ← J_N
    m_set ← randomize the elements in m_set
    n_set ← randomize the elements in n_set
    while n_set is not empty
        n ← the first number of n_set
        m ← the first number of m_set
        T ← UPDATE TABLE T SET J-Product = (J-Product * m / n)
        WHERE (J-Product % n = 0)
        n_set ← remove the first number of n_set
        m_set ← remove the first number of m_set
        m_set ← append n to the end of m_set
    return true
```

Figure 5-9 Pseudocode of the PROTECTION-KEY-UPDATE algorithm.

Chapter 6

CONCLUSION AND FUTURE WORKS

PPDM is significantly important for data mining activities involving sensitive private information. On one hand, data miners attempt to maximize the utility of training samples; on the other hand, they need to minimize the potential of data privacy loss throughout the data mining process. Researchers put a lot of effort in approaching both objectives; however, they are contesting against each other for many cases.

In previous research, Fong and Weber developed a novel PPDM method – Data Unrealization – that preserves both data privacy and utility of discrete-value training samples. During the privacy preserving process, the set of training data sets is dynamically modified as an unrealized version that has some relationship with the whole set of original data sets but is irrelevant to any individual one of them. This method guarantees the same data mining outcomes as the originals for decision tree data mining.

This dissertation extends their research scope and achieves new accomplishments: (1) it expands the application of Data Unrealization on other data mining algorithms, (2) it introduces data compression methods that reduce storage requirements for unrealized training samples and increase data mining performance and (3) it adds a second-level privacy protection that works perfectly with Data Unrealization.

From an application perspective, this dissertation proves that statistical information (i. e. counts, probability and information entropy) can be retrieved precisely from unrealized training samples, so that Data Unrealization is applicable for all counting-based, probability-based and entropy-based data mining models with 100%

accurate data mining results. These research findings are supported by mathematical proofs and examples. Algorithms of itemset mining and the Naïve Bayes model are given to show the mechanisms of changing the algorithm input from the original samples to the unrealized ones.

For our research purpose, a new number sequence, which is named J-Sequence (and denoted as J_N for a J-Sequence with N members), is introduced to provide fast and simple operations on the following tasks: (1) J-Lookup – which determines the existence of a member from any J-Product P where a J-Product is the product of some members in J_N, (2) J-Size – which determines the number of members contained in P, (3) J-Comparsion – which determines the order of multiple J-Products in terms of their size, (4) J-Union – which determines the set of all members contained in multiple J-Products and (5) J-Intersection – which determines the common members of multiple J-Products. The properties used to support these operations are theoretically proven – which determines the union sets of multiple J-Products. An algorithm is also provided to generate a J-Sequence with size N.

J-Sequence with size N can be utilized to compress a discrete-value data set with the total number of N possible attribute values (or less,) regardless of the number of column of the data set. Each attribute value is mapped by a unique member in the J-Sequence and the set of all J-Products covers the entire sample domain T^U, which has $|T^U|$ combinations of possible attribute values. After compression, the compressed data set will contain only one single column and it is the J-Product representing all J-Sequence members corresponding to the attribute values of the original data set. In this way, the entire set of training samples with sample size M will become a list of M numbers;

consequently, the storage complexity of the compressed samples is $O(M)$ and the time complexity of operation that counts the frequency of any data set t from the compressed samples is $O(M)$ because M times of J-Lookup are needed.

RLE converts a list of data values into a list of unit structured as the form of < $value, count$ >. This compression method can be applied to the compressed samples (denoted as J-Samples) for optimizing data compression. Sorting the J-Samples brings all samples with the same J-Product value aligned together. Appling RLE on the sorted J-Samples generates a list of $|T^U|$ (or less) structured < $value, count$ > units where $value$ is the J-Product value and *count* is its frequency in the J-Samples; hence, the overall storage complexity is $O(|T^U|)$ and the query time complexity is $O(|T^U|)$ regardless of the sample size. If we consider $|T^U|$ as a fixed constant because the size of sample domain is known and limited for discrete-value samples, then the storage complexity and the query time complexity become $O(1)$ and is $O(1)$ correspondingly.

As the resulting sample table is highly compressed, the same amount of data contains a lot more information than the original form. This may result in higher risk of privacy loss during a privacy attack event. To approach high compression rate and sustain the degree of privacy protection, our research recommends storing the uncompressed J-Samples into a column-oriented database system instead of storing the fully compressed samples as a sample table. In this way, the compression will be performed in the internal storage layer and isolated from any external access other than internal system calls; meanwhile, any access from the client side (which is the source of privacy attacks through the internet) will still view the samples in an uncompressed J-Samples form. As a result, performance and storage optimization can be achieved without sacrificing privacy

preservation. In fact, J-Sampling acts as an extra cryptographic approach (in which a J-Sequence acts as the protection keys) that provides a second-level privacy shelter with the flexibility of protection-key updates.

This dissertation is mainly a theoretical paper that validates all the research findings by mathematical theorems and some classroom examples. In future research, we recommend implementing the theory practically on different DBMS with large real-life samples and then re-evaluating the entire research work. Sample noise is another crucial factor that is absent currently and needs to be filled by future studies. In addition, the research is grounded from a claim – the sample size is much larger than the size of sample domain; therefore, research is suggested to examine the conditions of this statement such as the threshold of the sample-size/domain-size ratio.

Bibliography

[1] Vassilios S. Verykios , Elisa Bertino , Igor Nai Fovino , Loredana Parasiliti Provenza , Yucel Saygin , Yannis Theodoridis, *State-of-the-art in privacy preserving data mining*, ACM SIGMOD Record, v.33 n.1, March 2004 [doi>10.1145/974121.974131]

[2] CSO Security and Risk, Data Protection (2012, February 15) *The 15 worst data security breaches of the 21st Century*, Retrieved March 8, 2013, from http://www.csoonline.com/article/700263/the-15-worst-data-security-breaches-of-the-21st-century

[3] BBC News Technology (2012, June 6) *LinkedIn passwords leaked by hackers*, Retrieved March 8, 2013, from http://www.bbc.co.uk/news/technology-18338956

[4] Dara Kerr (2012, April 23) *Google ups cash reward for being hacked*, Retrieved March 8, 2013, from http://news.cnet.com/8301-1023_3-57419595-93/google-ups-cash-reward-for-being-hacked/

[5] Pui Kuen Fong , Jens H. Weber-Jahnke, *Privacy Preserving Decision Tree Learning Using Unrealized Data Sets*, IEEE Transactions on Knowledge and Data Engineering, v.24 n.2, p.353-364, February 2012 [doi>10.1109/TKDE.2010.226]

[6] Charu C. Aggarwal , Philip S. Yu, *Privacy-Preserving Data Mining: Models and Algorithms*, Springer Publishing Company, Incorporated, 2008

[7] Latanya Sweeney, *k-anonymity: a model for protecting privacy*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, v.10 n.5, p.557-570, October 2002 [doi>10.1142/S0218488502001648]

[8] Adam Meyerson , Ryan Williams, *On the complexity of optimal K-anonymity*, Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, June 14-16, 2004, Paris, France [doi>10.1145/1055558.1055591]

[9] i-Won Byun , Ashish Kamra , Elisa Bertino , Ninghui Li, Efficient k-anonymization using clustering techniques, Proceedings of the 12th international conference on Database systems for advanced applications, April 09-12, 2007, Bangkok, Thailand

[10] Hillol Kargupta , Souptik Datta , Qi Wang , Krishnamoorthy Sivakumar, *On the Privacy Preserving Properties of Random Data Perturbation Techniques*, Proceedings of the Third IEEE International Conference on Data Mining, p.99, November 19-22, 2003

[11] Jim Dowd , Shouhuai Xu , Weining Zhang, *Privacy-preserving decision tree mining based on random substitutions*, Proceedings of the 2006 international conference on Emerging Trends in Information and Communication Security, June 06-09, 2006, Freiburg, Germany [doi>10.1007/11766155_11]

[12] Shaofeng Bu; Lakshmanan, L.V.S., Ng, R.T., Ramesh, G, *Preservation Of Patterns and Input-Output Privacy*, IEEE 23rd International Conference on Data Engineering, ICDE 2007, p. 696 – 705, 2007

[13] Wikipedia, the free encyclopedia. *ID3 algorithm. (2013, February 24)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/ID3_algorithm

[14] Wikipedia, the free encyclopedia. *C4.5 algorithm. (2013, March 4)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/ C4.5_algorithm

[15] Wikipedia, the free encyclopedia. *Set theory. (2013, March 3)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/ Set_theory

[16] Wikipedia, the free encyclopedia. *Association rule learning. (2013, February 26)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/ Association_rule_learning

[17] Wikipedia, the free encyclopedia. *Apriori alogrithm. (2013, February 26)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/Apriori_algorithm

[18] Wikipedia, the free encyclopedia. *Naïve Bayes classifier. (2013, February 26)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/Naïve_Bayes_classifier

[19] David Salomon. 2004. *Data Structure in Graphs*. Data Compression (3rd. ed.). Springer, New York, NY, 20–25.

[20] Wikipedia, the free encyclopedia. *Database management system. (2013, March 6)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/DBMS

[21] Wikipedia, the free encyclopedia. *Column-oriented DBMS. (2013, February 26)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/Column-oriented_DBMS

[22] Wikipedia, the free encyclopedia. *Relational database management system. (2013, February 26)*. Retrieved March 8, 2013 from http://en.wikipedia.org/wiki/Relational_DBMS

[23] Wikipedia, the free encyclopedia. 2012. *Bitmap. (November 2012).* Retrieved November 28, 2012 from http://en.wikipedia.org/wiki/Bitmap

[24] Wikipedia, the free encyclopedia. 2012. *Linked list. (November 2012).* Retrieved November 28, 2012 from http://en.wikipedia.org/wiki/Linked_list

[25] Steven S. Skiena. 2008. *Data Structure in Graphs*. The Algorithm Design Manual (2nd. ed.). Springer, New York, NY, 151–155.

[26] Wikipedia, the free encyclopedia. 2012. *Neighbourhood (graph theory). (October 2012).* Retrieved November 24, 2012 from http://en.wikipedia.org/wiki/Neighbourhood_(graph_theory)

[27] Daniel J. Abadi , Samuel R. Madden , Nabil Hachem, *Column-stores vs. row-stores: how different are they really?*, Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 09-12, 2008, Vancouver, Canada [doi>10.1145/1376616.1376712]

[28] M. J. Turner , R. Hammond , P. Cotton, *A DBMS for large statistical databases*, Proceedings of the fifth international conference on Very Large Data Bases, p.319-327, October 03-05, 1979, Rio de Janeiro, Brazil

[29] Mike Stonebraker , Daniel J. Abadi , Adam Batkin , Xuedong Chen , Mitch Cherniack , Miguel Ferreira , Edmond Lau , Amerson Lin , Sam Madden , Elizabeth O'Neil , Pat O'Neil , Alex Rasin , Nga Tran , Stan Zdonik, *C-store: a column-oriented DBMS*, Proceedings of the 31st international conference on Very large data bases, August 30-September 02, 2005, Trondheim, Norway

[30] Raghu Ramakrishnan and Johannes Gehrke. 2003. Database Management Systems (3rd. ed.), McGraw-Hill, New York, NY

[31] Wikipedia, the free encyclopedia. *Multiset. (2013, February 23).* Retrieved March 19, 2013 from http://en.wikipedia.org/wiki/Multiset

[32] James William, *Unrealization Approaches for Privacy Preserving Data Mining*, master's thesis, Department of Computer Science, University of Victoria, 2010. [http://dspace.library.uvic.ca:8080/bitstream/handle/1828/3156/mainthesisUVIC.pdf?sequence=1]

[33] N. Li, T. Li, and S. Venkatasubramanian, *t-Closeness: Privacy Beyond k-Anonymity and l-Diversity*, ICDE 2007, p. 106–115, 2007.

[34] Marek Klonowski , Michał Przykucki , Tomasz Strumiński , Małgorzata Sulkowska, *Practical universal random sampling*, Proceedings of the 5th international conference on Advances in information and computer security, November 22-24, 2010, Kobe, Japan

[35] Kamalika Chaudhuri, Nina Mishra, *When random sampling preserves privacy*, Proceedings of the 26th annual international conference on Advances in Cryptology, p.198-213, August 20-24, 2006, Santa Barbara, California [doi>10.1007/11818175_12]

[36] Qingkai Ma, Ping Deng, *Secure Multi-party Protocols for Privacy Preserving Data Mining*, Proceedings of the Third International Conference on Wireless Algorithms, Systems, and Applications, p.526-537, 2008 [doi>1007/978-3-540-88582-5_49]

[37] Wei Jiang , Chris Clifton , Murat Kantarcıoğlu, *Transforming semi-honest protocols to ensure accountability*, Data & Knowledge Engineering, v.65 n.1, p.57-74, April, 2008 [doi>10.1016/j.datak.2007.06.014]

[38] Wikipedia, the free encyclopedia. *Homomorphic encryption. (2013, March 6)*. Retrieved March 20, 2013 from http://en.wikipedia.org/wiki/Homomorphic_encryption

[39] Wikipedia, the free encyclopedia. *Public-key cryptography. (2013, March 20)*. Retrieved March 20, 2013 from http://en.wikipedia.org/wiki/Public-key_cryptography