Privacy Preservation for Training Datasets in Database: Application to Decision Tree Learning

by

Pui Kuen Fong
BSc in Computer Science, University of Victoria, 2005

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

MASTER OF SCIENCE

in the Faculty of Engineering / Department of Computer Science

Supervisory Committee

Privacy Preservation for Training Datasets in Database:
Application to Decision Tree Learning

by

Pui Kuen Fong
BSc in Computer Science, University of Victoria, 2005

Supervisory Committee

Jens H. Weber, Department of Computer Science
Supervisor

Alex Thomo, Department of Computer Science
Departmental Member

Kui Wu, Department of Computer Science
Departmental Member

Abstract

Supervisory Committee
Jens H. Weber, Department of Computer Science
Supervisor
Alex Thomo, Department of Computer Science
Departmental Member
Kui Wu, Department of Computer Science
Departmental Member

Privacy preservation is important for machine learning and datamining, but measures designed to protect private information sometimes result in a trade off: reduced utility of the training samples. This thesis introduces a privacy preserving approach that can be applied to decision-tree learning, without concomitant loss of accuracy. It describes an approach to the preservation of privacy of collected data samples in cases when information of the sample database has been partially lost. This approach converts the original sample datasets into a group of unreal datasets, where an original sample cannot be reconstructed without the entire group of unreal datasets. This approach does not perform well for sample datasets with low frequency, or when there is low variance in the distribution of all samples. However, this problem can be solved through a modified implementation of the approach introduced later in this thesis, by using some extra storage.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to extend thanks and appreciation to my supervisor, Dr. Jens H. Weber, who has provided financial and personal support towards my study.

While I was studying full-time, Dr. Weber provided a research assistantship for my work and supported my application for the University of Victoria Fellowship. When I began my career outside of the city, he made arrangements to provide me with academic support remotely. He always offers me excellent guidance and knowledge, with patience and respect.

Finally, I would like to thank my fiancée and my best friend, Jessica Zhao, for her support and encouragement. She has stood by me with dedication, so I can put extra energy on my work and study.

Dedication

This thesis is dedicated to my grandmother, Kwai Lan Choi[1] (1910-2007), who brought me up and loved me all the time.

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

Chapter 1

INTRODUCTION

Datamining is widely used by researchers for science and business purposes. Data collected from individuals (referred to in this thesis as "information providers") are important for decision making or pattern reorganization. The data collection process takes effort and time, and the collected datasets (referred to as "sample datasets" or "samples" in this thesis) are sometimes stored for re-use. However, some unauthorized parties attempt to steal these sample datasets (referred to in this thesis as "privacy attacks") and to exploit the private information of information providers from these stolen datasets. Collected samples may also be lost during the storing process. Therefore, privacy preserving processes are developed to convert datasets containing private information (such as financial, medical and personal information) into altered or sanitized versions, in which the private information is "hidden" from unauthorized retrievers.

On the other hand, privacy-preserving processes which "hide" information may reduce the utility of those sanitized datasets. When their utility decreases to a certain level, the downgraded information prevents accurate analysis — with the result that the primary objective of datamining is compromised.

1.1 Research Background and Objectives

Even when databases of samples with sensitive information are protected securely, partial information of the databases can be lost through procedural mistakes[1][2] or privacy attacks from anywhere within a network[3][4]. This thesis focuses on analyzing

privacy preservation following the loss of some training datasets from the whole sample database used for decision-tree learning. On this basis, we make the following assumptions for the scope of this thesis: first, as is the norm in data collection processes, a large number of sample datasets have been collected to achieve significant datamining results that cover the whole research target. Second, the number of datasets lost constitutes a small portion of the entire sample database. Third, for decision-tree datamining, no attribute is designed for distinctive values, because such values negatively affect decision classification.[2]

The objective of this thesis is to introduce a new privacy preserving approach to the protection of sample datasets that are utilized for decision-tree datamining. Privacy preservation is applied directly to the samples in storage, so that privacy can be safeguarded even if the data storage were to be threatened by unauthorized parties. Although effective against privacy attacks by any unauthorized party, this approach does not affect the accuracy of datamining results. Moreover, this approach can be applied at any time during the data collection process, so that privacy protection can be in effect as early as the first sample is collected.

1.2 Contributions

According to my research on contemporary literatures, many privacy protection approaches preserve private information of sample datasets, but not precision of datamining outcomes. Hence, the utility of the sanitized datasets is downgraded. Some approaches apply transformation functions to sanitize the samples, and employ inverse

[2] Details will be explained in Chapter 6.

functions of those transformation functions to recover the original datasets. The accuracy of these datamining results can be maintained by "decoding" the sanitized datasets; however, security issues of the inverse functions are raised, because they are the keys to recover the original samples.

This thesis has two main contributions. Firstly, it provides an approach that preserves privacy and utility of sample datasets for decision-tree datamining. This approach converts samples into unreal datasets and generates the same datamining results as the originals. Secondly, this approach conducts datamining outcomes from the sanitized datasets directly, such that it is free from security issues of any required "decoding" process.

1.3 Thesis Organization

This thesis consists of seven chapters. Chapter 1 introduces the motivation, contribution and research background of this thesis. It also briefly describes the organization of the research content, so that readers understand the overall scope and presentation of this thesis.

Chapter 2 introduces the definitions and notations that are used throughout this thesis. These definitions and notations are utilized to explain additional concepts in the chapters that follow.

Chapter 3 describes the fundamental theoretical bases of decision-tree learning via the Iterative Dichotomiser 3 (ID3) approach. Diagrams, pseudocode and examples comprehensively elucidate ID3 decision-tree learning.

Chapter 4 describes other scholarly research on privacy preservation on database / datamining, and comments on these works from the viewpoint of the thesis focus. This chapter offers readers an overview of contemporary privacy preservation techniques related to the thesis focus.

Chapter 5 introduces a new perturbation-based privacy preserving approach that meets the research objectives of this thesis. This chapter offers a comprehensive explanation of its implementation, supported by proofs, diagrams, pseudocodes and examples, so that readers understand the whole picture of this approach.

Chapter 6 analyzes the privacy preservation performance of this new approach. Privacy issues are raised and analyzed, and solutions are proposed to improve the implementation method presented in Chapter 5. Additional evaluations, for purposes of the modified approach, are briefly provided in the later section of this chapter.

Chapter 7 provides an overall summary of this thesis, and suggests directions for further research on this topic.

Chapter 2

DEFINITIONS AND NOTATIONS

This chapter explains definitions and notations used in the following chapters. They are presented in the format of lists, enabling readers to easily look up these references while reading this thesis. To understand this chapter, readers will need to understand some fundamental concepts which will not be explained in this thesis: sets[5], tuples[6] and graphs[7].

2.1 Sets and Datasets

Let $A = \{a_1, a_2, \ldots, a_m\}$ be a set of attributes, and $T = \{t_1, t_2, \ldots, t_n\}$ be a data table that associates with A. Each dataset t_i is a tuple of attribute values $< k_1, k_2, \ldots, k_m >$ representing an individual's record such that $\{a_1 = k_i, a_2 = k_2, \ldots, a_m = k_m\}$. We have the following notations:

2.1.1 $t[a]$ denotes the value of attribute a for dataset t.

2.1.2 Let n, m, q, i and j_q be integers where $0 \le q \le n$, $1 \le i \le m$ and $1 \le j_q \le m$.

$A\{\overline{a_i}\}$ denotes $A - \{a_i\}$, which is $\{a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots a_m\}$, and $A\{\overline{a_{j_0}}, \overline{a_{j_1}}, \ldots, \overline{a_{j_n}}, \overline{a_i}\}$ denotes $A\{\overline{a_{j_0}}, \overline{a_{j_1}}, \ldots, \overline{a_{j_n}}\} - \{a_i\}$.

2.1.3 Let $K = < k_1, k_2, \ldots, k_m >$ be a tuple of values associates with a tuple of all attributes in $A = < a_1, a_2, \ldots, a_m >$ then A_K denotes $\{a_1 = k_i, a_2 = k_2, \ldots, a_m = k_m\}$.

2.1.4 Let $t = <t[a_1], t[a_2], \ldots, t[a_m]>$ then $t <\overline{a_i}>$ denotes $<t[a_1], t[a_2], \ldots, t[a_{i-1}], t[a_{i+1}], \ldots, t[a_m]>$ where $1 \leq i \leq m$.

2.1.5 Let T be a set containing some tuples $t = <t[a_1], t[a_2], \ldots, t[a_m]>$ then $T\{\overline{a_i}\}$ denotes a set containing some tuples $t <\overline{a_i}>$ where $1 \leq i \leq m$.

2.1.6 Let $t[a] = k$ then $T_{(a=k)}$ denotes a subset of T that contains t.

2.1.7 Let $t[a] = k$ then $T_{(a \neq k)}$ denotes $T - T_{(a=k)}$.

2.1.8 Let $t[a_i] = k$, $t[a_j] = l$ and $i \neq j$ then $T_{(a_i=k)\wedge(a_j=l)}$ denotes a subset of T that contains t.

2.2 Graphs

Let G be a tree with leaf nodes $L = \{l_1, l_2, \ldots, l_q\}$. A value k is assigned to each leaf node. Let $P = \{p_1, p_2, \ldots, p_q\}$ be the set of all paths in G with end nodes as the root and a leaf l. We have the following notations:

2.2.1 $L(p)$ denotes the value of leaf node l of the path p.

2.2.2 Let $l = k$, then L_k denotes a subset of L that contains l_i where i is an integer and $1 \leq i \leq q$. Each $l_i \in L$ belongs to one and only one $p_i \in P$. P_k denotes a subset of P and its $L(p_i)$ equals k.

Chapter 3

DECISION-TREE LEARNING

A decision tree describes a sequence of tests and their corresponding test outcomes. A test input is represented by a set of attributes with values. The outcome, which is known as the decision, represents the predicted output values of the input. The values of the inputs and outputs can be discrete or continuous. Regression learning approximates continuous-value functions; classification learning approximates discrete-value functions. In this thesis, we are focusing on classification learning, while continuous values can be treated as discrete by applying value ranges instead. The decision-tree structure can be used to represent meaningful information for humans, such as instructions and manuals; therefore, it is a common class of inductive learning methods[8].

3.1 Decision Tree[9]

A decision tree takes A_K as input at the root node, and returns an output value from a leaf node, as the decision of an attribute $d \notin A$. Each internal node (shown in Figure 3-1(a)) of a decision tree holds an attribute $a_i \in A$. Each branch from the node is labelled with a possible value of a_i, and connects to another node. Each leaf node (shown in Figure 3-1(b)) in the tree specifies a possible value of d. An internal node N, including the root node, takes an attribute $a_i \in A$ from the input and tests its value k_i against the values assigned to branches of N. If k_i satisfies the condition of a branch b (since the branches classify the possible values of a_i, k_i will satisfy one and only one branch

condition) the input A_K will be taken by the other node connected to *b*. Another test will be performed if A_K reaches another internal node. Otherwise, a leaf node is reached, and the tree returns the value of the leaf node connected to *b* as the decision of *d*. For greater clarity, an example shown in Figure 3-2 illustrates how a decision tree *G* works with *A* = {Wind, Outlook, Humidity}, *d* = Play and A_K = <Weak, Sunny, High>. Logically, any particular decision-tree hypothesis *G* with input A_K can be written as the following function:

$$G(A_K) = (p_1(A_K) + p_2(A_K) + \ldots + p_n(A_K)),$$

where $\{p_1, p_2, \ldots\ldots, p_n\}$ is the set of all paths in *G,* with end nodes as the root and a leaf *l*. $p_i(A_K)$ returns $L(p_i)$[3] if A_K satisfies the conditions of the tests of p_i; if not, it returns 0. In general, the objective of a decision tree is to predict the decision of *d* based on the condition of input A_K.

[3] Assume non-numeric output values of *d* map to a consistent set of numeric numbers, in which 0 is reserved for representing undefined values. For example, if a possible output value of *d* is any day in a week, then we can assign the possible output set {Undefined, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} = {0, 1, 2, 3, 4, 5, 6, 7}.

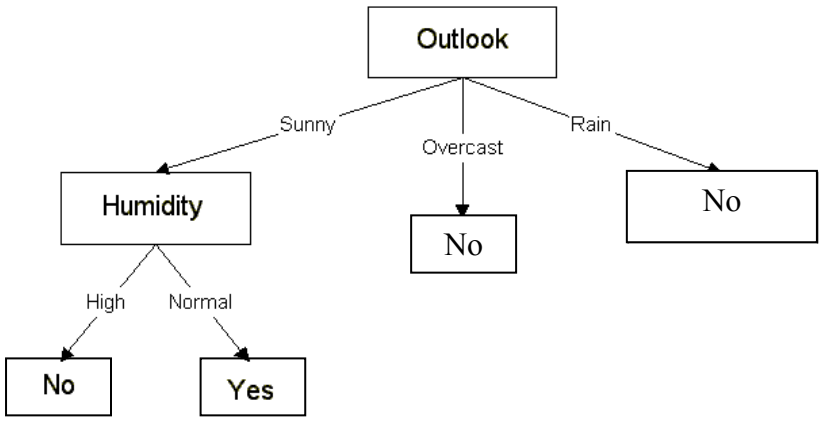

Figure 3-1 A Decision Tree Sample.

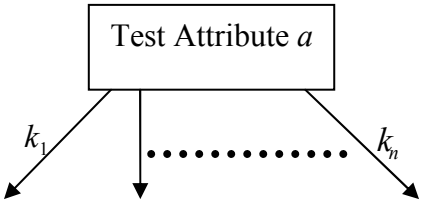

Figure 3-1(a) A Model of Internal Node. *a* is an attributes with possible values $k_{1, \ldots\ldots, } k_{n.}$ In Figure 3-1, the node "Outlook" with its labelled branches is an example of an internal node.

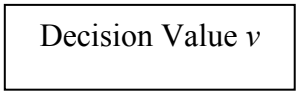

Figure 3-1(b) A Model of Leaf Node. *v* is a possible value of decision attribute *d*. In Figure 3-1, the node "Yes" is an example of a leaf node.

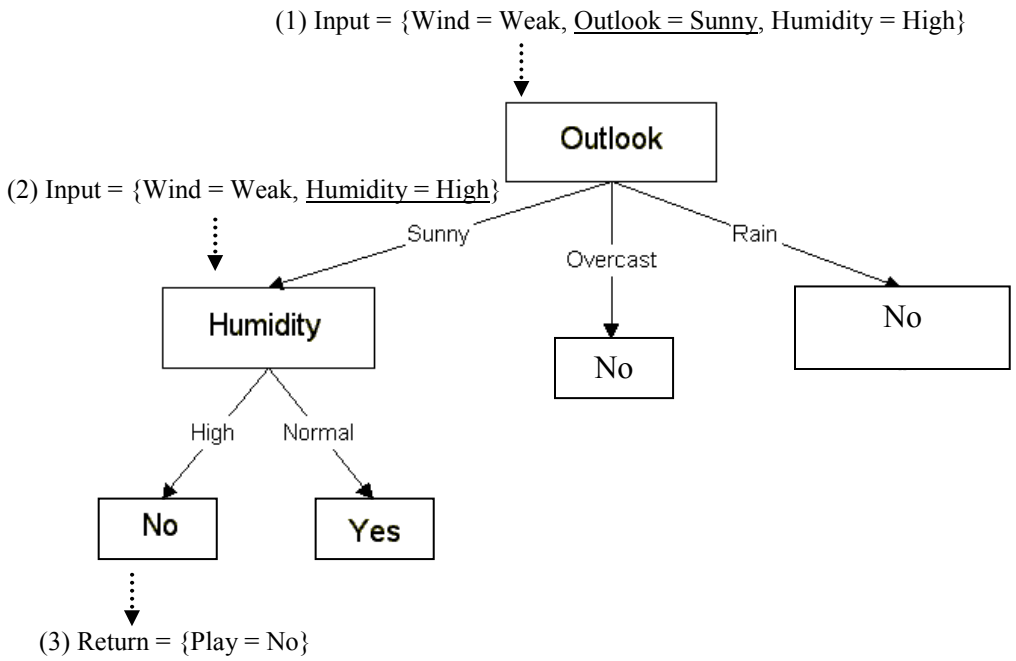

Figure 3-2 The process of generating a decision by decision tree G with input A_K.

3.2 Decision Tree Learning

From the objective of a decision tree, we may question how to determine a decision tree that makes good decisions. In other words, even the term "predict" tells the decision has uncertainties, how can we ensure the decision tree returning correct outputs for most of the cases? Let's take the 14 sample datasets t_i in Table 3-1[10] to test the decision tree G, as each t_i is defined as an input of G with the expected decision. By taking the input samples into G, only 50% of the outputs agree with the expected

decisions. The test result tells the decision tree does not make correct decision half of the time.

If a decision tree is built arbitrary, making the tree return correct outputs could be difficult. Decision tree learning is the process of inducing a decision tree from a training set T, some examples with known values and same attributes, say $A = \{a_1, a_2, \ldots, a_m\}$. If $a_i \in A$ is selected as the attribute of the decision values, then $A\{\overline{a_i}\}$ will be the set of attributes used to train the decision tree. The decision tree G is built by the top-down approach recursively, starting from the root node. The procedure is described as follows:

1) Terminal Case#1: if $T = \{\}$: a leaf node will be added to the tree with a default value.
2) Terminal Case#2: if all datasets in T haves the same decision value k, a leaf node will be added to the tree with value k.
3) Terminal Case#3: if $A\{\overline{a_i}\} = \{\}$, a leaf node will be added to the tree with value k, k is the decision value $t[a_i]$ having the maximum number of counts among all $t \in T$.
4) Recursive Case: if $A\{\overline{a_i}\} \neq \{\}$, $a_j \in A\{\overline{a_i}\}$ will be selected as the attribute of the internal node added to the tree, while each possible value of a_j will correspond to the labelled value of each branch from the node. The branches classify the training set into subsets, as $T_{(a=k)}$ belongs to the branch with value k. This branch is connected to a decision tree G built from the training set $T' = T_{(a=k)}\{\overline{a_i}\}$.

The pseudocode of the above procedure is shown on Figure 3-4. The final decision tree built from the above procedure guarantees to provide an output function for any input with attribute domain $A\{\overline{a_i}\}$. Also, this function will provide the right classifications for the training set. Therefore, if the training set is selected properly, the decision tree will make correct decisions. However, this paper will not discuss about the selection of training set further.

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 3-1 Sample datasets taken from real cases.

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

Figure 3-3 Pseudocode of the decision tree learning algorithm.

3.3 ID3 Algorithm

We can build different decision trees from the same training set by using the procedure described in the previous section, because of the undetermined selection criteria of the test attribute (function CHOOSE-ATTRIBUTE in Figure 3-3) in the recursive case. The effectiveness of a test attribute can be determined by its classification of the training set. A perfect attribute divides the outcomes as an exact classification, which achieves the goal of decision-tree learning. Different criteria are used to select the "best" attributes, e.g. Gini impurity[11]. Among these criteria, information gain is commonly used for measuring distribution of random events. Iterative Dichotomiser 3

(ID3) selects the test attribute based on the information gain provided by the test outcome. Information gain measures the change of uncertainty level after a classification from an attribute. Fundamentally, this measurement is rooted in information theory.

3.3.1 Information Entropy[12]

Information entropy (or "entropy") is a term that was introduced by Claude Shannon's information theory in 1948. In information theory, information content is measured in bits. Entropy measures the minimum number of bits necessary to communicate information. It can also be used to measure the uncertainty associated with a random variable.

If a random variable X has possible outcomes k_i with probabilities $P(k_i)$ while i is an integer and $1 \le i \le n$, then the information content I in bits can be expressed by:

$$H(X) = I(P(k_1), P(k_2), \ldots, P(k_n)) = -\sum_{i=1}^{n} P(k_i) \log_2 P(k_i)$$ [4]

Information content I indicates the uncertainties of event X. I is ranged from 0 to $-\log_2(\frac{1}{n})$, while 0 means the event is absolutely biased and $-\log_2(\frac{1}{n})$ means the event is fair. Let's take an event of a coin toss as an example. If we toss a fair coin, we have $P(head) = P(tail) = \frac{1}{2}$. The information content of the event is:

$$I(\frac{1}{2}, \frac{1}{2}) = -\frac{1}{2} * \log_2(\frac{1}{2}) - \frac{1}{2} * \log_2(\frac{1}{2}) = 1 = -\log_2(\frac{1}{2})$$

However, if the coin were loaded to give 99% heads, the information content of the event will be:

---

[4] $H(X)$ is defined as $-\log_2(1/n)$ if all $P(k_i) = 0/0$.

$$I(\frac{99}{100},\frac{1}{100}) = -\frac{99}{100} * \log_2(\frac{99}{100}) - \frac{1}{100} * \log_2(\frac{1}{100}) = 0.08,$$

which means the event has little uncertainty because it closes to a bias event with $I = 0$. Figure 3-4 shows the relation between the information content of a coin toss and the probability of it coming up heads. From the view of information communication, a coin toss event of a 100% bias coin (it gives 100% heads or 100% tails) gives no information content because we know the outcome before the toss. The closer it is to a fair coin, the harder it is to predict the outcome – and the more information content can be delivered from the event.

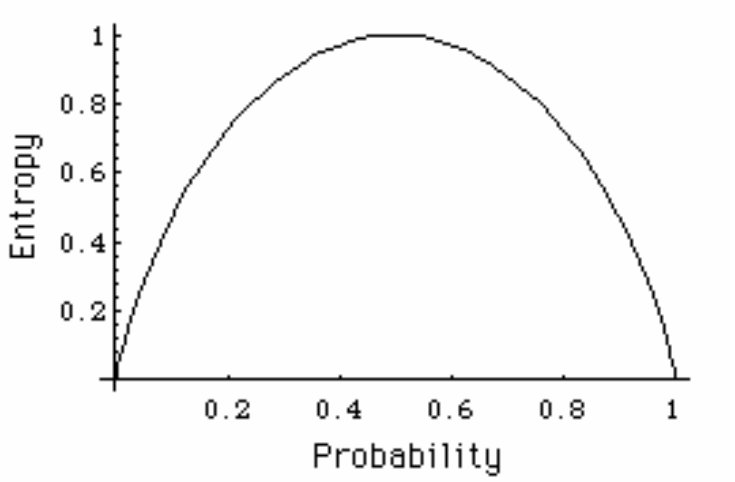

Figure 3-4 Information content of a coin toss as a function of the probability of it coming up heads.

3.3.2 Information Gain

What is the relation between information entropy and the selection criteria of a test attribute? If we treat the values of decision as the outcomes of a classification event of an attribute test, then some information content should be delivered from the event. In other words, we will pick a test attribute to classify the decision values only when it makes the decision more certain.

Information gain measures the gain in information content by a classification event of an attribute test. If T is the training set, a is the test attribute with possible values k_i (i is an integer and $1 \le i \le n$) and d is the decision attribute with possible values v_i (j is an integer and $1 \le j \le m$), then information gain *Gain* is shown as following:

$$Gain(a) = H_d(T) - H_d(T \mid a)$$

where $H_d(T)$ is the information content of d before the test, equals:

$$H_d(T) = -\sum_{j=1}^{m} P(d = v_i)\log_2 P(d = v_i) = -\sum_{j=1}^{m} \frac{|T_{d=v_i}|}{|T|}\log_2(\frac{|T_{d=v_i}|}{|T|})$$

and $H_d(T \mid a)$ is the condition information content of d with given a, equals:

$$H_d(T \mid a) = \sum_{i=1}^{n} P(a = k_i)H_d(T_{a=k_i}) = \sum_{i=1}^{n} \frac{|T_{a=k_i}|}{|T|}H_d(T_{a=k_i})$$

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.

Let's take the training set in Table 3-1 as an example: since the decisions of *Play* are not pure[5], a test attribute will be selected from $\{Outlook, Humidity, Wind\}$. The information entropy of the datasets equals,

$$H_{Play}(T) = -\frac{9}{14} * \log_2(\frac{9}{14}) - \frac{5}{14} * \log_2(\frac{5}{14}) = 0.941$$

If we take attribute *Wind* as the test attribute, the information entropy of the datasets after the classification equals,

$$H_{Play}(T \mid Wind)$$

$$= \frac{8}{14} * H_{Play}(T_{Wind=Weak}) + \frac{6}{14} * H_{Play}(T_{Wind=Strong})$$

$$= \frac{8}{14} * [-\frac{6}{8} * \log_2(\frac{6}{8}) - \frac{2}{8} * \log_2(\frac{2}{8})] + \frac{6}{14} * [-\frac{3}{6} * \log_2(\frac{3}{6}) - \frac{3}{6} * \log_2(\frac{3}{6})]$$

$$= 0.892$$

then information gain from the test equals,

$$Gain(Wind) = 0.941 - 0.892 = 0.049$$

Similarly, we get:

$$Gain(Outlook) = 0.247$$

$$Gain(Humidity) = 0.152$$

Attribute *Outlook* is the best choice as the root, because it has the largest information gain. As *Outlook* is selected as the test attribute, we get a consistent decision value if $Outlook = Overcast$, while decision values of $Outlook = Sunny$ and $Outlook = Rain$ are not unified (see Table 3-2 and Table 3-3.) For $Outlook = Sunny$, we get:

$$Gain(Humidity) = 0.970$$

[5] Pure implies values of the same attribute among all datasets are consistent.

$Gain(Wind) = 0.019$

And for $Outlook = Rain$, we get:

$Gain(Humidity) = 0.019$

$Gain(Wind) = 0.970$

Finally, the decision tree is shown as Figure 3-5.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	Yes
11	Sunny	Normal	Strong	Yes

Table 3-2 Sample datasets with $Outlook = Sunny$.

Sample#	Outlook	Humidity	Wind	Play
4	Rain	High	Weak	Yes
5	Rain	Normal	Weak	Yes
6	Rain	Normal	Strong	No
10	Rain	Normal	Weak	Yes
14	Rain	High	Strong	No

Table 3-3 Sample datasets with $Outlook = Rain$.

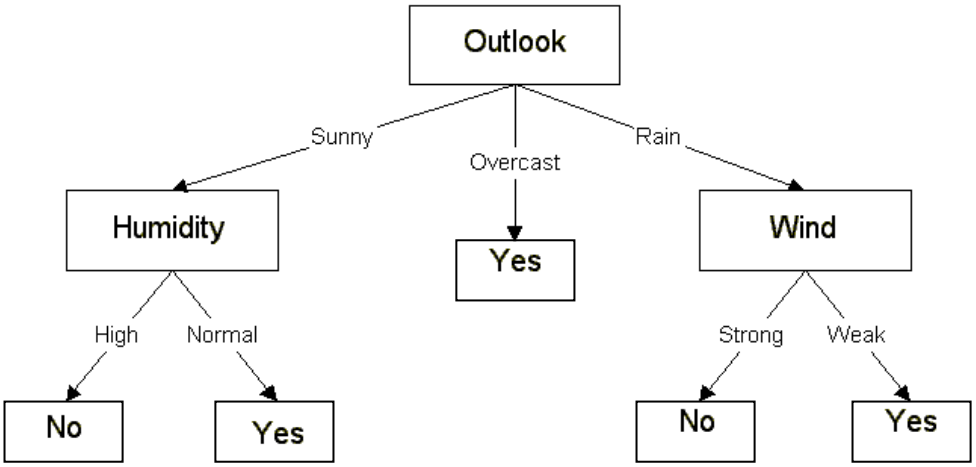

Figure 3-5 The final decision tree built from the training set in Table 3-1.

Chapter 4

DATA PRIVACY IN DATAMINING

In Chapter 3, we discuss decision trees and how they can be trained. To establish a good decision tree, we need a pool of training samples. For most cases, real data are collected from individuals for statistical utilities. Even if explicit identification information, e.g. names, can be removed for classification datamining[6], identities are traceable by matching individuals with a combination of non-identifying information such as date and place of birth, gender, and employer. In addition to storing the samples securely, the private information (particularly that which is medical or financial in nature) of those information providers must be kept in a sanitized version to prevent any kind of privacy leakage. The imperatives of data utility and confidentiality make privacy preservation an important field of research.

In *Privacy Preserving Data Mining: Models and Algorithms*[13], Aggarwal and Yu classify privacy preserving datamining techniques, including data modification, cryptographic, statistical, query auditing and perturbation-based strategies. Cryptographic, statistical and query auditing techniques are related to multi-party datamining protocol, inference control and security assurance, all of which are subjects outside of the focus of this thesis. In this chapter, we explore the privacy preservation techniques used by data modification and perturbation-based approaches, and summarize them in relation to decision-tree datamining.

---

[6] Detailed reasons for this will be explained in Chapter 6.

4.1 Data Modification Approaches

Data modification techniques maintain privacy by modifying attribute values of the sample datasets. Essentially, datasets are modified by eliminating or unifying uncommon elements among all datasets, such that each dataset within the sanitized samples is guaranteed to pass the threshold of similarity with the other datasets. These similar datasets 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 dataset is loosely linked with a certain number of information providers.

4.1.1 k-anonymity[14]

k-anonymity is a common data modification approach that intends to achieve effective data privacy preservation. The term "*k*-anonymity" implies that the quasi-identifier of each sanitized dataset 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 information provider with a significant probability of accuracy. If the quasi-identifier of each dataset is linked to at least k information providers, then they cannot be distinguished from the others. To achieve *k*-anonymity, suppression or aggregation techniques are used to "generalize" attribute values of datasets. After the generalization process, the domains of attributes are shrunk as attribute values are merged into groups. For example, an attribute *Outlook* may be initially defined by possible values $\{Sunny, Overcast, Rain\}$. After generalization, the possible values become $\{Sunny, Dark\}$.

Some generalization rules block uncommon attribute values totally, by replacing those values by a default one (e.g. "*" or "?")[15], such that the rare attribute values merge

into the same group assigned the default value. Some generalization rules freely partition sample datasets in a *d*-dimensional space (where *d* is the number of attributes of the quasi-identifier), to cluster at least *k* datasets in a partition[16]. Because it is NP-hard to find an optimal generalization solution, this thesis considers a heuristic approach. Hierarchy-based generalization is one particular approach to achieving *k*-anonymity. It does this by generalizing datasets via a predefined domain generalization hierarchy, which is a sequence of sets that describe the steps required to generalize a corresponding attribute over the domain of the quasi-identifier.

Let's take Table 3-1 as an example. As our study focuses on the privacy protection of all information of every single dataset, all of the attributes are selected as the quasi-identifier. Assume the domain generalization hierarchy shown in Figure 4-1 is applied for approaching 2-anonymity of all sample datasets. Three generalization steps are needed to achieve 2-anonymity of quasi-identifier $\{Outlook, Humidity, Wind, Play\}$, and the sanitized data table is shown in Table 4-1. The sanitized datasets 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 result in a significant loss of accuracy from a decision tree built from 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 – excepting instances where none of these attributes become a test / decision attribute of the final decision tree at all. However, this condition is impossible to detect until the entire data collection process has been completed.

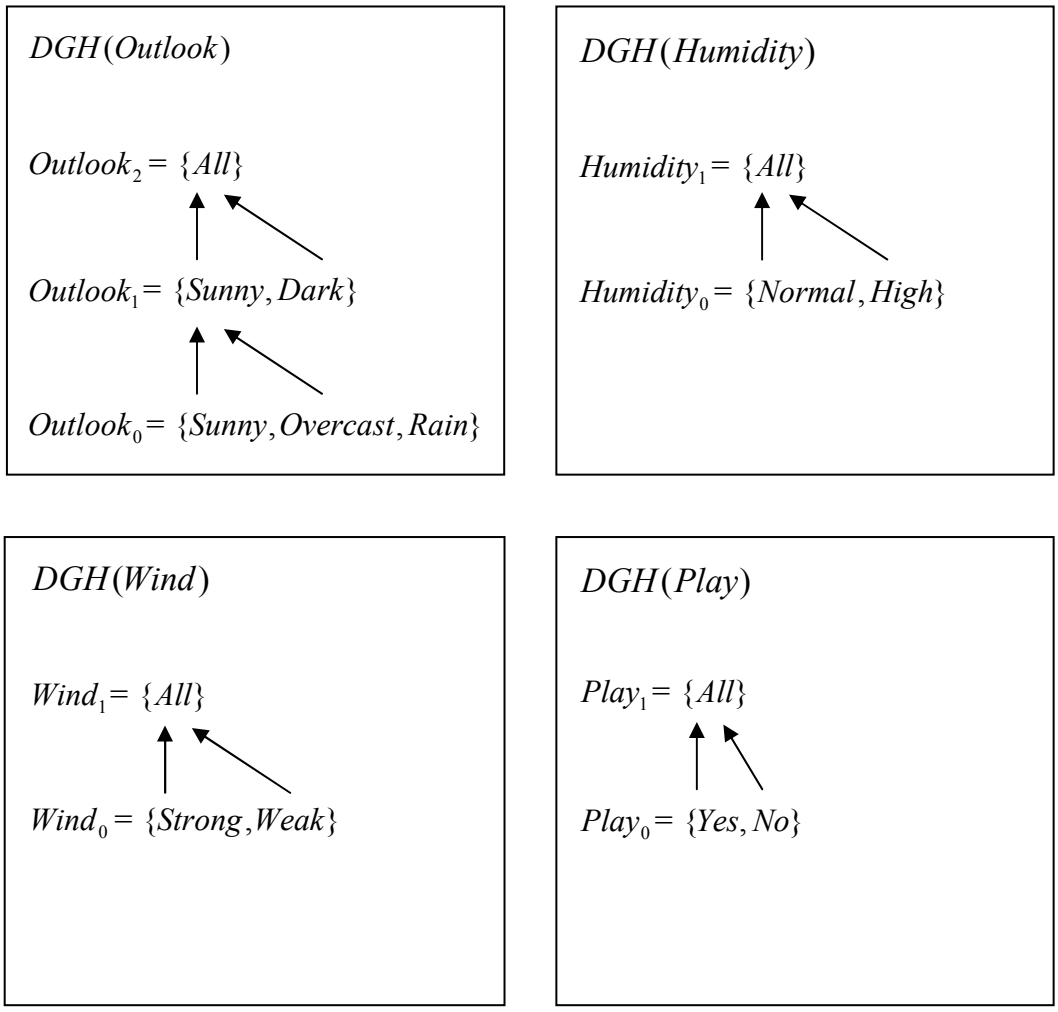

Figure 4-1 Domain generalization hierarchy of quasi-identifier $\{Outlook, Humidity, Wind, Play\}$ with generalization sequences $< Outlook_1, Humidity_1, Wind_1, Outlook_2, Play_1 >$.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	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 4-4 Sanitized data table after 3 generalization steps.

4.2 Perturbation-based Approaches[17]

Perturbation-based approaches attempt to achieve privacy protection by distorting information of the original datasets. By applying some data perturbation techniques, datasets are modified such that they are different from the originals. Meanwhile, the perturbed datasets still retain features of the originals, so that records derived from them

can be used to perform datamining, directly or indirectly, via data reconstruction. Two common strategies for data perturbation are noise-adding and random-substitution. Noise-adding adds a noise vector v_i to each sample u_i as a perturbed dataset $(u_i + v_i)$, such that the perturbed dataset is similar to u_i but linkage with the information provider is lost. Because this strategy is usually used for numeric values and has been proven to preserve little data privacy, it is not discussed in this thesis.

4.2.1 Random Substitution[18]

Instead of adding noise, random substitution perturbs samples by randomly replacing values of attributes. If the possible values $\{Sunny, Overcast, Rain\}$ of attribute $Outlook$ take the substitution rule as $\{Sunny \rightarrow Rain, Overcast \rightarrow Sunny, Rain \rightarrow Rain\}$, then datasets $< Sunny, Normal, Weak, Yes >$ and $< Overcast, Normal, Strong, No >$ will be replaced by $< Rain, Normal, Weak, Yes >$ and $< Sunny, Normal, Strong, No >$. Random substitution is attribute-based with a $(n * n)$ invertible matrix M, called a perturbation matrix, where n is the number of possible values of an attribute that is being perturbed. For optimal perturbation, $M = x * G$ where $x = \frac{1}{\gamma + n - 1}$, $G = \begin{bmatrix} \gamma & 1 & 1 & \cdots \\ 1 & \gamma & 1 & \cdots \\ 1 & 1 & \gamma & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$ and γ[7] ≥ 1 ($\gamma = 1$ for maximum privacy and $\gamma = \infty$ for maximum

[7]The original paper of the random substitution approach defines γ by using privacy breaching ρ_1-to-ρ_2. For a detailed discussion of privacy breaching, please refer to [xvii] and [xviii].

accuracy). The random substitution perturbation algorithm is described in Figure 4-2. If we assign $\gamma = 3$ and the random integer r always equals $\frac{|T'|+1}{|T_S|}$ [8] for perturbing attribute *Outlook* (with indices $\{Sunny = 1, Overcast = 2, Rain = 3\}$) of the samples in Table 3-1, the perturbation matrix will be $\begin{bmatrix} 3/5 & 1/5 & 1/5 \\ 1/5 & 3/5 & 1/5 \\ 1/5 & 1/5 & 3/5 \end{bmatrix}$ and the perturbed datasets will be shown as Table 4-2. The time complexity of attribute substitution is $O(|T_S| * n)$.

After random substitution, the information related to a particular attribute in the perturbation datasets is irrelevant to that of the original datasets. For datamining, the perturbation datasets must undergo dataset reconstruction. The reconstructed datasets are an estimation of the originals, based on the reconstruction matrix R, where $R = M^{-1} * Y$ and Y is a row matrix corresponding to the counts of each possible value of the perturbed attribute a in the perturbed datasets T'. Since R should not contain any negative entry, all the negative entries in $M^{-1} * Y$ will become 0 in R. For the datasets in Table 4-2, matrices Y and R corresponding to attribute *Outlook* are [3,6,5] and $\begin{bmatrix} 0.5 \\ 8 \\ 5.5 \end{bmatrix}$ [9]. The reconstruction process requires the perturbed datasets T', the perturbed attribute a, and the reconstruction matrix R as inputs for the algorithm shown in Figure 4-3. To

[8] The size of T' increases incrementally during the random substitution perturbation process.

[9] The original paper of the random substitution approach does not mention how to deal with fractional numbers for the reconstruction method, so I round the numbers as an input for function MATRIX-BASED RECONSTRUCTION.

reconstruct the datasets in Table 4-2, we sort the datasets as shown in Table 4-3 and produce the reconstructed datasets as shown in Table 4-4.

From the reconstructed datasets, we get a decision tree of *Play* , as shown in Figure 4-4. If the decision tree is tested by the original datasets in Table 3-1, about 29% of the datasets produce negative results against the tree. The random substitution is applied only for the attribute *Outlook* , so the remaining attributes still store the real information. If we were to protect the privacy of other attributes by random-substituting their values, the accuracy of the final decision tree would decrease further. Moreover, the complexity of the substitution process would become l times larger where l is the number of privacy protected attributes.

function RANDOM SUBSTITUTION PERTURBATION(T_S , a , M) **returns** T'

inputs: T_S , a set of input sample datasets

a , an attribute with possible values $\{c_1, c_2, \ldots, c_n\}$

M , a $(n * n)$ perturbation matrix with entries $m_{row,cloumn}$

$T' = \{\}$

foreach $t \in T_S$

c = attribute value of a in t

k = index of c in a

$t' = t$

obtain a random number r in range $(0,1]$

find an integer $1 \leq h \leq n$ such that $\sum_{i=1}^{h-1} m_{i,k} < r \leq \sum_{i=1}^{h} m_{i,k}$

c = attribute value of a with index h

set attribute value of a in t' as c

$T' = T' + \{t'\}$

return T'

Figure 4-2 Pseudocode of random substitution perturbation algorithm.

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 4-5 Sanitized data table after random substitution.

```
function MATRIX-BASED RECONSTRUCTION(T', a, M) returns T_R
    inputs: T', a set of perturbed datasets
            a, an attribute with possible values {c_1, c_2, ..., c_n}
            R, a row matrix with n entries r_row,cloumn
    T_R = {}
    sort T' in the order according to c_i in a
    t' = first dataset in T'
    for i = 1 to n do
            c = attribute value of a with index i
            for j = 1 to r_i,1 do
                    set attribute value of a in t' as c
                    T_R = T_R + {t'}
                    if t' is the last dataset in T'
                            return T_R
                    t' = next dataset in T'
    return T_R
```

Figure 4-3 Pseudocode of random substitution perturbation algorithm.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Sunny	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 4-6 Sorted perturbed datasets by $Outlook$ in the order of $[Sunny, Overcast, Rain]$.

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 4-7 Reconstructed datasets according to attribute *Outlook* .

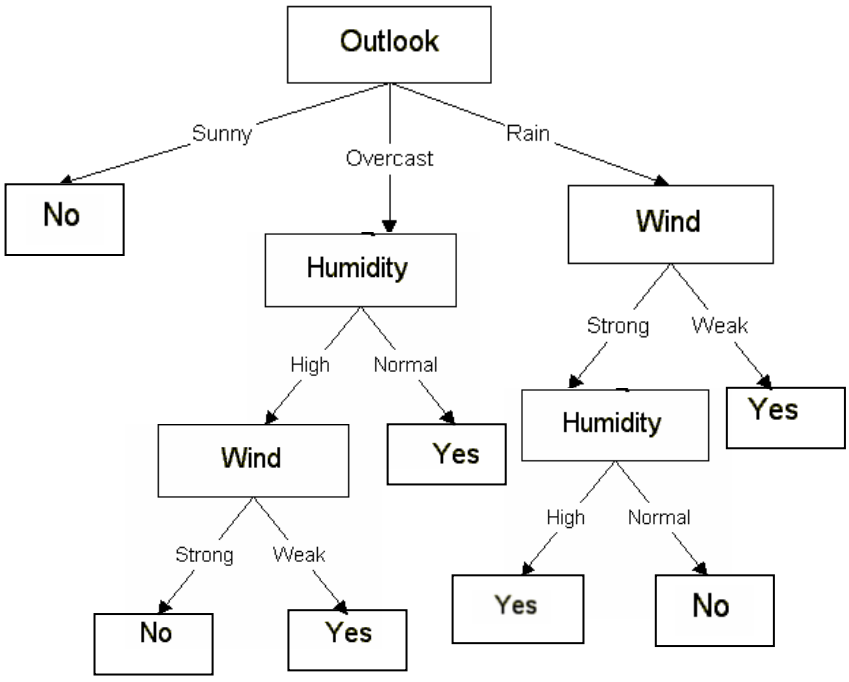

Figure 4-4 Decision tree built from the reconstructed dataset in Table 4-3.

4.2.2 Monotone / Anti-monotone Framework[19]

The perturbation approach via (anti)monotone functions is designed for decision-tree datamining. This framework preserves both the privacy of samples and the accuracy of datamining outcomes. Breakpoints are introduced to break up the sample datasets into subgroups and an (anti)monotone function[10] is assigned to each group. A series of (anti)monotone 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, which is defined as:

[10] Monotone functions can be applied for numeric values only.

Let the original domain [A] be broken up into w [subgroups] $\delta_1(A),\ldots, \delta_w(A)$ *with w transformation functions* $f_1, f_2, \ldots, f_w$. *This set of transformations is said to satisfy the global-monotone invariant iff for all* $1 \leq i < j \leq w$, $\forall v \in \delta_i(A)$, $\forall u \in \delta_j(A)$, *it is necessary that* $f_i(v) < f_j(u)$. *Similarly, the set is said to satisfy the global-anti-monotone invariant if the latter inequality is changed to* $f_i(v) > f_j(u)$.

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 that have different decision values[11]. Based on those subgroups derived from the breakpoints, a family of bijective functions that follow the constraint could be defined arbitrarily[12]. Let's take the samples in Table 4-5, which are sorted by attribute *Age*, to define breakpoints regarding to 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 $55.5 < x$

[11] This thesis simply interprets a breakpoint-choosing method from the original literature. For the full details on breakpoint selection, please refer to the original literature.

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

to δ_1, δ_2, δ_3 and δ_4 respectively, then the samples will be sanitized as the datasets in Table 4-6, 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 datasets sanitized by a transformation function. For example, f_1^{-1}: $Age = y - 5$ 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[13] to recover datasets of subgroups δ_1, δ_2, δ_3 and δ_4 in Table 4-6; second, the composition of decision tree remains the same after transformation, which means the original decision tree can be reconstructed by applying the inverse functions to the transformed decision tree[14] according to the range of breakpoints; and third, the transformation and recover process of each attribute are independent to the others, such that assignment of transformation and inverse functions of each attribute preserves the conservation of the recovered datamining results.

Even though the application of (anti)monotone functions saves both 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 datamining results, or the transformation functions should be kept to determine their inverse functions. In either way, it makes the privacy attackers possible to "crack" a subgroup of original datasets by "stealing" one of the stored functions. Furthermore, (anti)monotone functions are applicable for ranged-valued

[13] f^1 is denoted as the inverse function of f.

[14] Transformed decision tree is the decision tree built from the transformed datasets.

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 enumerates any symbolic-valued attribute into numeric-valued, such as changing *Gender* = <Male, Female> to *Gender* = <0, 1>. From the dimension of a particular discrete-valued attribute, transformed datasets having the same attribute value 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 4-5 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 4-6 Transformed datasets of samples in Table 4-5.

4.3 Conclusion

Data modification approaches are effective at hiding most of the private content in modified samples, but may leave unmodified samples vulnerable. If the modified samples are not recoverable (such as in the *k*-anonymity approach shown above), they are not useful for training a meaningful decision tree. For each sample, data modification preserves either privacy or utility. Hence, the whole group of training samples could be viewed as a pool of unprotected samples, along with some noise datasets.

Compared with data modification approaches, perturbation-based approaches do not make strict tradeoffs between privacy and the preservation of data sample utility. Rather than depending on the attribute values of every sample, perturbation-based approaches rely on a mechanism to preserve the privacy of each sample independently. The mechanism's effectiveness is greatly determined by its design, and as the foregoing discussion of the random substitution approach illustrates, the results – in terms of both privacy and data utility preservation – can be equally random.

Privacy preservation via (anti)monotone functions overcomes the shortage of random substitution approach. The (anti)monotone framework keeps both data sample privacy and utility; however, it raises the security issues on the defence of those inverse functions, which are the keys to reconstruct the originals. Furthermore, this framework

encounters limitations on handling discrete-valued attributes. Therefore, the requirements of effective mechanisms is an area which should be prioritized for further research.

Chapter 5

DATASET COMPLEMENTATION APPROACH

Privacy preservation via dataset complementation is a data perturbed approach that substitutes each original dataset with an entire unreal dataset. Unlike privacy protection strategies discussed in Chapter 4, this new approach preserves the original accuracy of the training datasets without linking the perturbed datasets to the information providers. In other words, dataset complementation can preserve the privacy of individual records and yield accurate datamining results. However, this approach is designed for discrete-value classification only, such that ranged values must be defined for continuous values.

In this chapter, we introduce, with examples, the foundations of dataset complementation and its application in decision-tree learning. The data tables in these examples have an attribute "Sample #", which is used as a primary key reference but not as an option of a decision or test attributes. Readers should keep this in mind while reading this chapter.

5.1 Definitions of Dataset Complement

5.1.1 Universal Set

In set theory, a universal set U is a set which contains all elements[20]. In this paper, a universal set T^U, relating to a data table T, is a set of datasets that contains a single instance of each valid dataset of T. In other words, any combination of a possible value from each attribute in the dataset sequence of T exists in T^U. If t is a dataset in

T associated with a tuple of attributes $< a_1, a_2, \ldots, a_m >$ and a_i has n_i possible values $K_i = \{k_1, k_2, \ldots, k_{n_i}\}$, then $< t[a_1], t[a_2], \ldots, t[a_i], \ldots, t[a_m] > \in T^U$ and $t[a_i] \in K_i$. We define:

T^U is a set containing a single instance of all possible datasets in data table *T*.

Let's take Table 3-1 as an example. The table associates with attributes $< Outlook, Humidity, Wind, Play >$ and possible attribute values are defined as: $Weather = \{Sunny, Overcast, Rain\}$, $Humidity = \{High, Normal\}$, $Wind = \{Strong, Weak\}$ and $Play = \{Yes, No\}$; T^U is shown in Table 5-1 in a data table form. Since the datasets in a data table are not necessarily unique, we allow for multiple instances of an element existing in the same set (known as a multiset, or a bag[21]). If T_D is a subset of T and q is a positive integer, then we define:

A *q*-multiple-of T_D, denoted as qT_D, is a set of datasets containing *q* instances of each dataset in T_D.

Therefore, $2T^U = \{ < Sunny, High, Strong, Yes >, < Sunny, High, Strong, No >, < Sunny, High, Weak, Yes >, < Sunny, High, Weak, No >, < Sunny, Normal, Strong, Yes >, < Sunny, Normal, Strong, No >, < Sunny, Normal, Weak, Yes >, < Sunny, Normal, Weak, No >, < Overcast, High, Strong, Yes >, < Overcast, High, Strong, No >, < Overcast, High, Weak, Yes >, < Overcast, High, Weak, No >, < Overcast, Normal, Strong, Yes >, < Overcast, Normal, Strong, No >, < Overcast, Normal, Weak, Yes >, < Overcast, Normal, Weak, No >, < Rain, High, Strong, Yes >, < Rain, High, Strong, No >, < Rain, High, Weak, Yes >, < Rain, High, Weak, No >,$

$< Rain, Normal, Strong, Yes >$, $< Rain, Normal, Strong, No >$, $< Rain, Normal, Weak, Yes >$, $< Rain, Normal, Weak, No >$, $< Sunny, High, Strong, Yes >$, $< Sunny, High, Strong, No >$, $< Sunny, High, Weak, Yes >$, $< Sunny, High, Weak, No >$, $< Sunny, Normal, Strong, Yes >$, $< Sunny, Normal, Strong, No >$, $< Sunny, Normal, Weak, Yes >$, $< Sunny, Normal, Weak, No >$, $< Overcast, High, Strong, Yes >$, $< Overcast, High, Strong, No >$, $< Overcast, High, Weak, Yes >$, $< Overcast, High, Weak, No >$, $< Overcast, Normal, Strong, Yes >$, $< Overcast, Normal, Strong, No >$, $< Overcast, Normal, Weak, Yes >$, $< Overcast, Normal, Weak, No >$, $< Rain, High, Strong, Yes >$, $< Rain, High, Strong, No >$, $< Rain, High, Weak, Yes >$, $< Rain, High, Weak, No >$, $< Rain, Normal, Strong, Yes >$, $< Rain, Normal, Strong, No >$, $< Rain, Normal, Weak, Yes >$, $< Rain, Normal, Weak, No > \}$.

For a *q*-multiple-of a universal set, all possible values of the same attribute *a* have the same number of counts. This feature also applies to the combination of any two attributes[15]. If a *q*-multiple-of a universal set is taken as the training set, regardless of which decision attribute is chosen, there is no good choice of test attribute because the information gain of any test is 0[16]. The closer a training set gets to *q*-multiple-of a universal set, the smaller becomes the quantity of information content of an attribute that can be retrieved from another.

Lemma 5-1:

---

[15] See Lemma 5-1.

[16] See Lemma 5-2.

T^U is the universal set of a data table T, which associates with a tuple of attributes $< a_1, a_2, \ldots, a_m >$. If $A = \{a_1, a_2, \ldots, a_m\}$, $\{a_i, a_{i+1}, \ldots, a_{j-1}, a_j\} \subseteq A$ where a_i has n_i possible values $K_i = \{p_1, p_2, \ldots, p_{n_i}\}$, a_j has n_j possible values $K_j = \{r_1, r_2, \ldots, r_{n_j}\}$ and $i \leq j$ such that $k_i \in K_i$, $k_j \in K_j$ and $< k_1, k_2, \ldots, k_i, \ldots, k_j, \ldots, k_m > \in T^U$, then for any non-negative integer q:

$|qT^U|$

$= q * |T^U|$

$=$ q * (number of possible values of a_1 * number of possible values of a_2 * … * number of possible values of a_m)

$= q * n_1 * n_2 * \ldots * n_m$

$|qT^U{}_{(a_i=k_i)}|$

$= q * |T^U{}_{(a_i=k_i)}|$

$=$ q * (number of possible values of a_1 * number of possible values of a_2 * … * number of possible values of a_{i-1} * number of possible values of a_{i+1} * … * number of possible values of a_m)

$= \dfrac{q * n_1 * n_2 * \ldots * n_m}{n_i}$

Similarly,

$|qT^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}|$

$= q * |T^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}|$

$$= \frac{q * n_1 * n_2 * \ldots * n_m}{n_i * \ldots * n_j}$$

Lemma 5-2:

Following Lemma 5-1, if a_i is the decision attribute and a_j is the test attribute, then:

$$H_{a_i}(qT^U)$$

$$= -\sum_{i=1}^{n_i} \frac{|qT^U_{(a_i=k_i)}|}{|qT^U|} \log_2 \frac{|qT^U_{(a_i=k_i)}|}{|qT^U|}$$

$$= -n_i \sum_{i=1}^{n_i} \frac{q^*|T^U_{(a_i=k_i)}|}{q^*|T^U|} \log_2 \frac{q^*|T^U_{(a_i=k_i)}|}{q^*|T^U|}$$

$$= -\log_2 \frac{1}{n_i}$$

$$H_{a_i}(qT^U_{(a_j=k_j)})$$

$$= -\sum_{i=1}^{n_i} \frac{|qT^U_{(a_i=k_i)\wedge(a_j=k_j)}|}{|qT^U_{(a_j=k_j)}|} \log_2 \frac{|qT^U_{(a_i=k_i)\wedge(a_j=k_j)}|}{|qT^U_{(a_j=k_j)}|}$$

$$= -n_i * \frac{q^*|T^U_{(a_i=k_i)\wedge(a_j=k_j)}|}{q^*|T^U_{(a_j=k_j)}|} \log_2 \frac{q^*|T^U_{(a_i=k_i)\wedge(a_j=k_j)}|}{q^*|T^U_{(a_j=k_j)}|}$$

$$= -\log_2 \frac{1}{n_i},$$

Therefore, all events in qT^U are fair.

$$H_{a_i}(qT^U|a_j)$$

$$= \sum_{j=1}^{n_j} \frac{|qT^U_{(a_j=k_j)}|}{|qT^U|} H_{a_i}(qT^U_{(a_j=k_j)})$$

$$= n_j * \frac{q^* | T^U{}_{(a_j = k_j)} |}{q^* | T^U |} H_{a_i}(qT^U{}_{(a_j = k_j)})$$

$$= -\log_2 \frac{1}{n_i}$$

Therefore, $H_{a_i}(qT^U \mid a_j)$ is fair.

Gain(a_j) = 0

Therefore, no information content of any attribute can be gained from another attribute.

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 5-8 A universal set T^U of data table T.

5.1.2 Dataset Complement

A relative complement of X in Y, denoted as $Y \setminus X$, refers to all elements in set Y belonging to Y, excepting those in set X. $Y \setminus X$ can be determined by subtracting X from Y. If Y is a universal set U, then $U \setminus X$ is called an absolute complement and denoted as X^C. In this paper, we apply the above concepts to the definition of a complement of a set of datasets, relating to a data table T.

A relative complement of a set of datasets T_{D_2} in a set of datasets T_{D_1} is denoted as $T_{D_1} \setminus T_{D_2}$ and equals to $T_{D_1} - T_{D_2}$.

An absolute complement of a set of datasets T_D is denoted as $T_D{}^C$ and equal to $T^U \setminus T_D$.

A *q*-absolute-complement of a set of datasets T_D is denoted as $qT_D{}^C$ and equal to $qT^U \setminus T_D$.

Since $qT_D{}^C = qT^U - T_D \Rightarrow qT^U = qT_D{}^C + T_D$, $qT_D{}^C \subseteq qT^U$ and $T_D \subseteq qT^U$. Let's reconsider Table 5-1 as an example. If we have a set of datasets T_{D_1} = $\{<Rain, High, Weak, Yes>, <Sunny, High, Strong, No>\}$ and T_{D_2} = $\{<Overcast, High, Weak, Yes>, <Overcast, High, Weak, No>, <Overcast, Normal, Weak, Yes>\}$, then $T_{D_1}{}^C \setminus T_{D_2} = T^U - T_{D_1} - T_{D_2} = \{<Sunny, High, Strong, Yes>, <Sunny, High, Weak, Yes>, <Sunny, High, Weak, No>, <Sunny, Normal, Strong, Yes>, <Sunny, Normal, Strong, No>, <Sunny, Normal, Weak, Yes>, <Sunny, Normal, Weak, No>, <Overcast, High, Strong, Yes>, <Overcast, High, Strong, No>, <Overcast, Normal, Strong, Yes>, <Overcast, Normal, Strong, No>, <Overcast, Normal, Weak, No>, <Rain, High, Strong, Yes>, <Rain, High, Strong, No>, <Rain, High, Weak, No>, <Rain, Normal, Strong, Yes>, <Rain, Normal, Strong, No>, <Rain, Normal, Weak, Yes>, <Rain, Normal, Weak, No>\}$. The result of $T_{D_1}{}^C \setminus T_{D_2}$ is shown in Table 5-2.

If T_{D_2} is a subset of T_{D_1}, then all the elements existing in T_{D_2} also exist in T_{D_1}. Thus, the information content gained by classifying *q*-absolute-complement of a set of datasets, says T_D, could be determined by using the size of qT^U and the information of T_D[17].

[17] See Lemma 5-3 and 5-4.

Lemma 5-3:

Following the conditions described in Lemma 5-1, if T_{D_1} and T_{D_2} are two sets of datasets that are subsets of T and $T_{D_2} \subseteq T_{D_1}$, then $t \in T_{D_2} \Rightarrow t \in T_{D_1}$.

$$|T_{D_1} \setminus T_{D_2}|$$

$$= |T_{D_1} - T_{D_2}| = |T_{D_1}| - |T_{D_2}|$$

$$|[T_{D_1} \setminus T_{D_2}]_{(a_i=k_i)}|$$

$$= |T_{D_1(a_i=k_i)} - T_{D_2(a_i=k_i)}| = |T_{D_1(a_i=k_i)}| - |T_{D_2(a_i=k_i)}|$$

$$|[T_{D_1} \setminus T_{D_2}]_{(a_i=k_i)\wedge(a_j=k_j)}|$$

$$= |T_{D_1(a_i=k_i)\wedge(a_j=k_j)} - T_{D_2(a_i=k_i)\wedge(a_j=k_j)}| = |T_{D_1(a_i=k_i)\wedge(a_j=k_j)}| - |T_{D_2(a_i=k_i)\wedge(a_j=k_j)}|$$

Lemma 5-4:

Following the conditions described in Lemma 5-2 and 5-3,

$$H_{a_i}(qT_D{}^C)$$

$$= -\sum_{i=1}^{n_i} \frac{|[qT^U \setminus T_D]_{(a_i=k_i)}|}{|[qT^U \setminus T_D]|} \log_2 \frac{|[qT^U \setminus T_D]_{(a_i=k_i)}|}{|[qT^U \setminus T_D]|}$$

$$= -\sum_{i=1}^{n_i} \frac{\frac{q*n_1*n_2*\ldots*n_m}{n_i} - |T_{D(a_i=k_i)}|}{(q*n_1*n_2*\ldots*n_m) - |T_D|} \log_2 \frac{\frac{q*n_1*n_2*\ldots*n_m}{n_i} - |T_{D(a_i=k_i)}|}{(q*n_1*n_2*\ldots*n_m) - |T_D|}$$

$$H_{a_i}(qT_D{}^C{}_{(a_j=k_j)})$$

$$= -\sum_{i=1}^{n_i} \frac{|[qT^U \setminus T_D]_{(a_i=k_i)\wedge(a_j=k_j)}|}{|[qT^U \setminus T_D]_{(a_j=k_j)}|} \log_2 \frac{|[qT^U \setminus T_D]_{(a_i=k_i)\wedge(a_j=k_j)}|}{|[qT^U \setminus T_D]_{(a_j=k_j)}|}$$

$$= -\sum_{i=1}^{n_i} \frac{x}{y} \log_2 \frac{x}{y},$$

where $x = \frac{q * n_1 * n_2 * \ldots * n_m}{n_i * n_j} - |T_{D(a_i=k_i)\wedge(a_j=k_j)}|$ and

$$y = \frac{q * n_1 * n_2 * \ldots * n_m}{n_j} - |T_{D(a_j=k_j)}|$$

$$H_{a_i}(qT_D{}^C \mid a_j)$$

$$= \sum_{j=1}^{n_j} \frac{|qT_D{}^C{}_{(a_j=k_j)}|}{|qT_D{}^C|} H_{a_i}(qT_D{}^C{}_{(a_j=k_j)})$$

$$= \sum_{j=1}^{n_j} \frac{|qT^U{}_{(a_j=k_j)}| - |T_{D(a_j=k_j)}|}{|qT^U| - |T_D|} H_{a_i}(qT_D{}^C{}_{(a_j=k_j)})$$

$$= \sum_{j=1}^{n_j} \frac{\frac{q * n_1 * n_2 * \ldots * n_m}{n_j} - |T_{D(a_j=k_j)}|}{(q * n_1 * n_2 * \ldots * n_m) - |T_D|} H_{a_i}(qT_D{}^C{}_{(a_j=k_j)})$$

$$= -\sum_{j=1}^{n_j} \sum_{i=1}^{n_i} \frac{x}{z} \log_2 \frac{x}{y},$$

where $x = \frac{q * n_1 * n_2 * \ldots * n_m}{n_i * n_j} - |T_{D(a_i=k_i)\wedge(a_j=k_j)}|$,

$$y = \frac{q * n_1 * n_2 * \ldots * n_m}{n_j} - |T_{D(a_j=k_j)}| \text{ and}$$

$$z = (q * n_1 * n_2 * \ldots * n_m) - |T_D|$$

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Strong	Yes
2	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
13	Overcast	Normal	Strong	Yes
14	Overcast	Normal	Strong	No
16	Overcast	Normal	Weak	No
17	Rain	High	Strong	Yes
18	Rain	High	Strong	No
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 5-9 A relative complement $T_{D_1}{}^{C} \setminus T_{D_2}$ where $T_{D_1} = \{ < Rain, High, Weak, Yes >, < Sunny, High, Strong, No > \}$ and $T_{D_2} = < Overcast, High, Weak, Yes >, < Overcast, High, Weak, No >, < Overcast, Normal, Weak, Yes > \}$.

5.2 Data Complementation Approach

Because each dataset of a training set represents the real information of an information provider, protection against real information leakage from a dataset to an unauthorized party is of primary concern. If datasets of the training set are perturbed, encrypted or broken down, it is difficult to retrieve the real information directly even if an unauthorized party obtains access to those datasets in any way. The privacy preservation approach of dataset complementation converts input datasets t_i into unreal ones by subtracting t_i from some predefined training sets with unreal information. The resulting unreal datasets can be used to calculate the information contents required by the ID3 algorithm. In this way, the datasets retrieved from the final training set, which contains some predefined datasets excepting the ones in t_i, are unreal individually, but meaningful when they are used together.

5.2.1 Unrealized Training Set

Traditionally, a training set T is constructed by inserting sample datasets into a data table. However, a data complementation approach requires an extra data table T^P. T^P is a perturbing set that generates unreal datasets for converting the sample datasets as an unrealized training set T'. Whenever we get a sample dataset t, we remove t from the perturbing table T^P if T^P contains t and $T^P \setminus \{t\}$ is not empty; otherwise, we insert $\{t\}^C = T^U \setminus \{t\}$ into T^P. After that, one dataset t'_i is transferred from T^P to T'. For convention, the first dataset in T^P will be transferred to T' in this thesis. If $T_S = \{t_1, t_2, \ldots, t_n\}$ is the set of sample datasets taken in the whole sample collection process

and t_i is a dataset we get from T_S each time, then the procedure for generating the unrealized training set T' from the set of sample datasets T_S can be described as follows:

1) Terminal Case: if $T_S = \{\}$, return training set T' and perturbing set T^P.
2) Recursive Case #1: if $t_i \in T^P$ and $T^P \setminus \{t_i\} \neq \{\}$, then $T^P = T^p - \{t_i\} - \{t'_i\}$, $T_S = T_S - \{t_i\}$ and $T' = T' + \{t'_i\}$. Build T' from T_S with T^P.
3) Recursive Case #2: if $t_i \notin T'$ or $T^P \setminus \{t_i\} = \{\}$, then $T^P = T^p + \{t_i\}^C - \{t'_i\}$, $T_S = T_S - \{t_i\}$ and $T' = T' + \{t'_i\}$. Build T' from T_S with T^P.

The pseudocode for unrealizing the training set is shown on Figure 5-1. To unrealize the samples, we initialize both T' and T^P as empty sets, i.e. UNREALIZED TRAINING-SET(T_S, T^U, {}, {}) is called. Consistent with the procedure described above, T^U is added as a parameter of the function because reusing pre-computed T^U is more efficient than recalculating T^U when $\{t\}^C$ is needed in Recursive Case #2. The recursive function UNREALIZED TRAINING-SET takes one dataset 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 3-1 as T_S, and the universal set T^U in Table 5-1, as the inputs of UNREALIZED TRAINING-SET(T_S, T^U, {}, {}). If we assume the sequence of t_i taken in T_S is Sample #1, Sample #2, …, Sample #14, then Tables 5-3 and 5-4; Tables 5-5 and 5-6; and Tables 5-7 and 5-8 show the datasets in T' and T^P after

the function takes Sample #1, Sample #7 and Sample #8. The function returns T' and T^P shown in Tables 5-9 and 5-10.

function UNREALIZED TRAINING-SET(T_S, T^U, T', T^P) **returns** $<T', T^P>$
  inputs: T_S, a set of input sample datasets
    T^U, a universal set
    T', a set of output training datasets
    T^P, a set of unreal datasets
  if T_S is empty **then**
    return $<T', T^P>$
  $t_i \leftarrow$ a dataset in T_S
  if t_i is an element of T^P **and** $T^p \setminus \{t_i\}$ is not empty **then**
    $T^P \leftarrow T^p - \{t_i\}$
    $t'_i \leftarrow$ a dataset in T^P
  else
    $T^P \leftarrow T^p + T^U - \{t_i\}$
    $t'_i \leftarrow$ a dataset in T^P
  return UNREALIZED TRAINING-SET($T_S - \{t_i\}$, T^U, $T'+\{t'_i\}$, $T^p - \{t'_i\}$)

Figure 5-1 Pseudocode of unrealized training set algorithm.

Sample#	Outlook	Humidity	Wind	Play
1	Sunny	High	Strong	Yes

Table 5-10 Datasets in T' after 1st recursion of the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, {}).

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

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 5-11 Datasets in T^P after 1st recursion of the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, {}).

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

Table 5-12 Datasets in T' after 7th recursion of the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, 0).

Sample#	Outlook	Humidity	Wind	Play
10	Overcast	High	Strong	No
12	Overcast	High	Weak	No
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
20	Rain	High	Weak	No
21	Rain	Normal	Strong	Yes
24	Rain	Normal	Weak	No

Table 5-13 Datasets in T^P after 7th recursion of the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, 0).

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

Table 5-14 Datasets in T' after 8th recursion of the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, {}).

Sample#	Outlook	Humidity	Wind	Play
12	Overcast	High	Weak	No
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
20	Rain	High	Weak	No
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
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
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
25	Overcast	Normal	Weak	Yes
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
33	Rain	Normal	Weak	Yes
34	Rain	Normal	Weak	No

Table 5-15 Datasets in T^P after 8th recursion of the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, {}).

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 5-16 Training datasets T' returned by the function call UNREALIZED 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 5-17 Perturbing datasets T^P returned by the function call UNREALIZED TRAINING-SET(T_S, T^U, {}, {}).

5.2.2 Reconstruct Information Entropy and Information Gain

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 datasets can be performed.

From the UNREALIZED TRAINING-SET algorithm shown in Figure 5-1, it is trivial that the size of T_S is the same as the size of T'. Furthermore, all datasets in $(T'+T^P)$ are based on the datasets in some T^U, excepting the ones in T_S, i.e. T_S is the q-absolute-complement of $(T'+T^P)$ for some positive integer q. The size of the q-multiple-of T^U, equals $2*|T'|+|T^P|$[18], can be found from the sizes of T' and T^P at any instance. Therefore, entropies of the original datasets T_S, with any decision attribute and any test attribute, can be determined by the unreal training set T' and perturbing set T^P[19], so that the best test attribute can be chosen without T_S in a decision-tree learning process.

By modifying the DECISION-TREE LEARNING algorithm in Figure 3-1, a decision tree can be built from the size of the q-multiple-of T^U, the unrealized training set T' and the perturbing set T^P, which is shown in Figure 5-2. $H_{a_i}(q[T'+T^P]^C) = H_{a_i}(T_S) = 0$ implies datasets in T_S having a single classification. Other than the input datasets and CHOOSE-ATTRIBUTE function, the default decision value of the input sets is changed. Since $|T_{S(a_i=k_i)}| = |qT^U{}_{(a_i=k_i)}| - |[T'+T^P]_{(a_i=k_i)}| \Rightarrow |T_{S(a_i=k_i)}|$

[18] See Lemma 5-5.

[19] See Lemma 5-6.

$+|[T'+T^P]_{(a_i=k_i)}|$ = constant, the default decision value can be retrieved from the minority decision value of $(T'+T^P)$, instead of the majority decision value of T_S.

If we consider the datasets in Table 3-1 as T_S, then Table 5-6a and 5-6b are T' and T^P. Initially, we have $|qT^U| = 2*|T'|+|T^P| = 48$:

$$H_{Play}(T_S)$$

$$= -\frac{24-|T'_{(Play=Yes)}|+|T^P{}_{(Play=Yes)}|}{14}\log_2\frac{24-|T'_{(Play=Yes)}|+|T^P{}_{(Play=Yes)}|}{14}$$

$$-\frac{24-|T'_{(Play=No)}|+|T^P{}_{(Play=No)}|}{14}\log_2\frac{24-|T'_{(Play=No)}|+|T^P{}_{(Play=No)}|}{14}$$

$$= -\frac{9}{14}\log_2\frac{9}{14}-\frac{5}{14}\log_2\frac{5}{14}$$

$$= 0.410+0.531$$

$$= 0.941$$

$$H_{Play}(T_S|Wind)$$

$$= -\frac{12-|T'_{(Play=Yes)\wedge(Wind=Strong)}|+|T^P{}_{(Play=Yes)\wedge(Wind=Strong)}|}{14}*$$

$$\log_2\frac{12-|T'_{(Play=Yes)\wedge(Wind=Strong)}|+|T^P{}_{(Play=Yes)\wedge(Wind=Strong)}|}{24-|T'_{(Wind=Strong)}|-|T^P{}_{(Wind=Strong)}|}$$

$$-\frac{12-|T'_{(Play=No)\wedge(Wind=Strong)}|+|T^P{}_{(Play=No)\wedge(Wind=Strong)}|}{14}*$$

$$\log_2\frac{12-|T'_{(Play=No)\wedge(Wind=Strong)}|+|T^P{}_{(Play=No)\wedge(Wind=Strong)}|}{24-|T'_{(Wind=Strong)}|-|T^P{}_{(Wind=Strong)}|}$$

$$-\frac{12-|T'_{(Play=Yes)\wedge(Wind=Weak)}|+|T^P{}_{(Play=Yes)\wedge(Wind=Weak)}|}{14}*$$

$$\log_2 \frac{12-|T'_{(Play=Yes)\wedge(Wind=Weak)}|+|T^P_{(Play=Yes)\wedge(Wind=Weak)}|}{24-|T'_{(Wind=Weak)}|-|T^P_{(Wind=Weak)}|}$$

$$-\frac{12-|T'_{(Play=No)\wedge(Wind=Weak)}|+|T^P_{(Play=No)\wedge(Wind=Weak)}|}{14}*$$

$$\log_2 \frac{12-|T'_{(Play=No)\wedge(Wind=Weak)}|+|T^P_{(Play=No)\wedge(Wind=Weak)}|}{24-|T'_{(Wind=Weak)}|-|T^P_{(Wind=Weak)}|}$$

$$= -\frac{3}{14}\log_2\frac{3}{6}-\frac{3}{14}\log_2\frac{3}{6}-\frac{6}{14}\log_2\frac{6}{8}-\frac{2}{14}\log_2\frac{2}{8}$$

$$= 0.214 + 0.214 + 0.178 + 0.286$$

$$= 0.892$$

$$Gain(Wind) = 0.941 - 0.892 = 0.049$$

Similarly, we get:

$$Gain(Outlook) = 0.247$$

$$Gain(Humidity) = 0.152$$

Therefore, attribute "Outlook" is selected as the test attribute and we get Tables 5-11 and 5-12, Tables 5-13 and 5-14, and Tables 5-15 and 5-16 as the tables associated with possible values of "Outlook". As:

$$|qT^U_{(Outlook=Sunny)}| = |qT^U_{(Outlook=Overcast)}| = |qT^U_{(Outlook=Rain)}| = \frac{|qT^U|}{3} = 16,$$

we get:

$$H_{Play}(T_{S(Outlook=Overcast)}) = 0$$

with the minority decision value as "Yes". By applying another DECISION-TREE LEARNING' procedure for Tables 5-11 and 5-12 and Table 5-15 and 5-16, we get:

$$Gain(Humidity) = 0.970$$

$Gain(Wind) = 0.019$

for datasets in Tables 5-11 and 5-12 and:

$Gain(Humidity) = 0.019$

$Gain(Wind) = 0.970$

for datasets in Table 5-15 and 5-16. Therefore, we find the final decision tree (shown in Figure 5-3) is exactly the same as the one in Figure 3-5.

Lemma 5-5:

Given that $T_S = q[T'+T^P]^C$, $|T'| = |T_S|$ for some positive integer q and the conditions of Lemma 5-1 and Lemma 5-3, if a_i is the decision attribute with n_i possible attribute values and a_j is the test attribute with n_j possible attribute values, then:

$$T_S = q[T'+T^P]^C$$

$$\Rightarrow \quad T_S = qT^U - (T'+T^P)$$

$$\Rightarrow \quad |T_S| = |qT^U - (T'+T^P)|$$

$$\Rightarrow \quad |T_S| = |qT^U| - |(T'+T^P)|, \text{ since } (T'+T^P) \subseteq qT^U$$

$$\Rightarrow \quad |T_S| = |qT^U| - |T'| - |T^P|$$

$$\Rightarrow \quad |T'| = |qT^U| - |T'| - |T^P|, \text{ since } |T'| = |T_S|$$

$$\Rightarrow \quad |qT^U| = 2*|T'| + |T^P|$$

$$\Rightarrow \quad |qT^U| = 2*|T'| + |T^P|$$

$$\Rightarrow \quad |qT^U| = 2*|T'| + |T^P|$$

$$|qT^U{}_{(a_i=k_i)}|$$

$$= \frac{2*|T'|+|T^P|}{n_i}$$

$$|qT^U{}_{(a_i=k_i)\wedge\ldots\wedge(a_j=k_j)}|$$

$$= \frac{2*|T'|+|T^P|}{n_i * \ldots * n_j}$$

Lemma 5-6:

Given that $T_S = q[T'+T^P]^C$ for some positive integer q and the conditions of Lemma 5-4, if a_i is the decision attribute with n_i possible attribute values and a_j is the test attribute with n_j possible attribute values, then:

$$H_{a_i}(T_S)$$

$$= H_{a_i}(q[T'+T^P]^C)$$

$$= -\sum_{i=1}^{n_i} \frac{\frac{|qT^U|}{n_i} - |[T'+T^P]_{(a_i=k_i)}|}{|qT^U| - |[T'+T^P]|} \log_2 \frac{\frac{|qT^U|}{n_i} - |[T'+T^P]_{(a_i=k_i)}|}{|qT^U| - |[T'+T^P]|}$$

$$= -\sum_{i=1}^{n_i} \frac{x}{y} \log_2 \frac{x}{y},$$

where $x = \frac{|qT^U|}{n_i} - |T'_{(a_i=k_i)}| + |T^P{}_{(a_i=k_i)}|$ and

$$y = |qT^U| - |T'| - |T^P|$$

$$H_{a_i}(T_{S(a_j=k_j)})$$

$$= H_{a_i}(q[T'+T^P]^C{}_{(a_j=k_j)})$$

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

$$= -\sum_{i=1}^{n_i} \frac{x}{y} \log_2 \frac{x}{y},$$

where $x = \frac{|qT^U|}{n_i * n_j} - |T'_{(a_i=k_i)\wedge(a_j=k_j)}| - |T^P{}_{(a_i=k_i)\wedge(a_j=k_j)}|$ and

$$y = \frac{|qT^U|}{n_j} - |T'_{(a_j=k_j)}| - |T^P{}_{(a_j=k_j)}|$$

$H_{a_i}(T_S \mid a_j)$

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

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

$$= -\sum_{j=1}^{n_j} \sum_{i=1}^{n_i} \frac{x}{z} \log_2 \frac{x}{y},$$

where $x = \frac{|qT^U|}{n_i * n_j} - |T'_{(a_i=k_i)\wedge(a_j=k_j)}| - |T^P{}_{(a_i=k_i)\wedge(a_j=k_j)}|$,

$$y = \frac{|qT^U|}{n_j} - |T'_{(a_j=k_j)}| - |T^P{}_{(a_j=k_j)}| \text{ and}$$

$$z = |qT^U| - |T'| - |T^P|$$

```
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 datasets
            T^P, set of perturbing datasets
            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 ← {datasets in T' with best = v_i}
            T^P_i ← {datasets 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 5-2 Pseudocode of the modified decision tree learning algorithm using T' and T^P.

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

Table 5-18 Unrealized training dataset $T'_{(Outlook=Sunny)}$.

Sample#	Outlook	Humidity	Wind	Play
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

Table 5-19 Perturbing datasets $T^{P}_{(Outlook=Sunny)}$

Sample#	Outlook	Humidity	Wind	Play
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

Table 5-20 Unrealized training data $T'_{(Outlook=Overcast)}$

Sample#	Outlook	Humidity	Wind	Play
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

Table 5-21 Perturbing datasets $T^{P}_{(Outlook=Overcast)}$.

Sample#	Outlook	Humidity	Wind	Play
13	Rain	High	Strong	Yes
14	Rain	High	Weak	No

Table 5-22 Unrealized training data $T'_{(Outlook=Rain)}$.

Sample#	Outlook	Humidity	Wind	Play
21	Rain	Normal	Strong	Yes
24	Rain	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 5-23 Perturbing datasets $T^{P}_{(Outlook=Rain)}$.

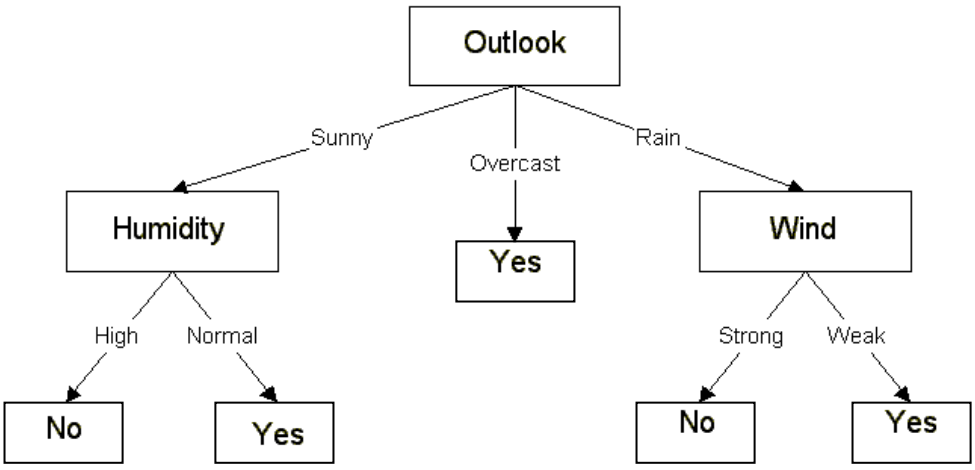

Figure 5-3 The final decision tree built from datasets in Table 5-15 and 5-16.

5.3 Dataset Reconstruction

The previous section introduced a modified decision-tree learning algorithm by using the unrealized training set T' and perturbing set T^P directly. Alternatively, we can reconstruct the original sample datasets T_S from T' and T^P[20], and then apply the regular procedure for getting the decision tree from T_S. The reconstruction process is dependent upon the full information of T' and T^P; reconstruction of parts of T_S based on parts of T' and T^P is not possible.

Lemma 5-7:

By following the conditions in Lemma 5-1 and Lemma 5-5, we conclude:

$$2*|T'|+|T^P| = |qT^U| = q\ *|T^U|$$

$$\Rightarrow \quad q = \frac{2*|T'|+|T^P|}{|T^U|}$$

$$\because \quad T_S = qT^U - T' - T^P$$

$$\therefore \quad T_S = \frac{2*|T'|+|T^P|}{|T^U|} * T^U - T' - T^P$$

---

[20] See Lemma 5-7.

Chapter 6

EVALUATION

In Chapter 5, we introduced a modified decision-tree learning process via dataset complementation. This approach converts the original sample datasets into unrealized training datasets with some perturbing datasets, such that each dataset used for decision making is not directly linked to the information provider. However, is this approach really preserving privacy in every circumstance? In this chapter, we will discuss potential privacy problems inherent in this approach, and provide a solution for them. In later sections of this chapter, we will cover additional problems that may arise.

6.1 Privacy Issues

6.1.1 Background

As described in Chapter 1, this thesis is based upon certain assumptions. Firstly, the sample size s is large in comparison to the number of datasets leaked to an unauthorized party. Secondly, each attribute does not have distinctive values because these negatively affect decision classification. For example, if social insurance number or full address are used as attributes and we have s sample datasets, then we will have a decision tree as an internal node having s branches – with one dataset belonging to each branch.[21] [22] This does not yield any useful decision at all. Meanwhile, these distinctive

[21] The count of each distinctive value of an attribute is equal or close to 1. Therefore, the attribute has 0 (or close to 0) information entropy, which it is why it is the selected test attribute from the CHOOSE-ATTRIBUTE process.

[22] Some attributes, such as Postal Code, have a large group of attribute values and each value is shared by very limited individuals. These attributes should be eliminated because they make an internal node have a large number of branches with a very small number of dataset belonging to each branch.

values are all identifying information that can heavily impact privacy. By excluding those identifying attributes, we need most information of a dataset to identify the information provider.

In "Toward Privacy in Public Database[22]", privacy preservation is defined as follows:

[P]rotection from being brought to the attention of others[.]

Because this thesis focuses on protection of individuals' information against leak from sample datasets, we will assume that all information in a sample dataset contains some private content. Privacy in a dataset is defined to have been lost when all of the following conditions apply: 1) real information of some person *A* is collected as a sample dataset; 2) the sample dataset, which may be sanitized[23], is stored in a sanitized database; 3) person *B* obtains some dataset(s) from the database; 4) person *B* retrieves, from those datasets, some real information about *A* without *A*'s authorization.

Assuming the amount of privacy information in an original sample dataset is 1, we can simplify the privacy loss described above in the following formula:

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

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 datasets obtained by person B, $|T_D|$ is the total number datasets in the sanitized database and $P(T_S, T_S)$ is the total amount of privacy information of the whole database T_D. Privacy information of a sample can be "brought to attention of others" either when an original dataset is recovered from a sanitized

[23] For convention, if a sample dataset is stored directly without any privacy preservation, then the sanitized dataset implies the original sample itself in the remaining content of this thesis.

dataset, or when a sanitized dataset matches with a sample by chance. Importantly, the privacy information of a sanitized dataset, a reconstructed dataset and a sample datasets may vary. $P(T_S, T_D)$ can be measured by the matching rate between each sample dataset in T_S and each sanitized dataset in T_D (with the privacy information in a sanitized dataset P_D), along with the reconstruction of the original datasets with privacy information P_R:

$$P(T_S, T_D) = (1-\alpha)\sum_{t' \in T_D}\sum_{t \in T_S} M(t,t')P_D(t,t') + \alpha \sum_{t' \in T_D}\sum_{t \in T_S} P_R(t,t'),$$

where $M(t,t') = 1$ if t' is the same as the sanitized version of t, or 0 otherwise, and α is the probability that *B* could reconstruct the original datasets.

For example, if a set of sample datasets are not privacy-preserved, then T_S equals T_D. $P_D(t,t')$ and $P_R(t,t')$ are equal to 1. Consequently, $P(T_S, T_S)$ ranges from $x^*|T_S|$ (if all datasets in T_S are evenly distributed and x is the average counts of all datasets in T_S) to $|T_S|^2$ (if all datasets in T_S are the same) and $P_{loss}(T_S, T_S)$ ranges from $r*x$ to $r^*|T_S|$.

6.1.2 Privacy in Data Set Complementation

For dataset complementation, T_D is the datasets in the unrealized training set T' and perturbing set T^P. A dataset in $(T'+T^P)$ does not have any direct relationship with any dataset in the sample datasets in T_S. Datasets in T_S should be recovered as a whole by using all of the datasets in T' and T^P. If we have all datasets (or almost all datasets) in $(T'+T^P)$, we define:

$$\sum_{t' \in (T' + T^P)} \sum_{t \in T_S} P_R(t,t') = |T_S|$$

$$\Rightarrow \quad P_R(t,t') = \frac{|T_S|}{|[T' + T^P]| * |T_S|}, \text{ since all } P_R(t,t') \text{ are equal weighted.}$$

$$\Rightarrow \quad P_R(t,t') = \frac{1}{|T_S| + |T^P|}, \text{ since } |T'| = |T_S|$$

In the scope of this thesis, we assume the number of datasets lost, r, is comparatively small regarding to the total number of datasets in T_S, so α is defined as 0 because reconstruction is impossible. Thus, the reconstructed privacy information $\sum_{t' \in T_D} \sum_{t \in T_S} P_R(t,t')$ can be omitted.

Since a sanitized dataset is still in the domain of T_S, $P_D(t,t')$ is defined as 1. To determine the matching perturbed privacy information, we must first to determine the relation between T_S and $(T' + T^P)$. Again, $T_S + (T' + T^P) = qT^U$. 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 datasets in T^U as Figure 6-1(a). Since T_S and $(T' + T^P)$ are subsets of qT^U, if T_S can be represented as the datasets contained in rectangles shown in Figure 6-1(b), then datasets in $(T' + T^P)$ are the ones not contained by the rectangles. By reorganizing the datasets in T^U according to the number of counts of t_i in T_S, we get the relationship of qT^U, T_S and $(T' + T^P)$ shown in Figure 6-2 where x (the minimum among the counts of t_i in T_S) indicates the region (called "Region x" in this thesis) with datasets purely in T_S, y (the maximum among the counts of t_i in T_S

$-x$) indicates the region (called "Region y" in this thesis) with both the datasets in T_S and $(T'+T^P)$, and z ($q - y - x$) indicates the region (called "Region z" in this thesis) with datasets purely in $(T'+T^P)$. If $y = 0$, it implies that the distribution of the datasets in T_S is even[24], then we have:

$$\sum_{t'\in(T'+T^P)}\sum_{t\in T_S} M(t,t')$$

$$= x * z * n$$

$$= \frac{|T_S|}{n} * (\frac{2*|T'|+|T^P|}{n} - \frac{|T_S|}{n}) * n, \text{ by Lemma 5-7}$$

$$= \frac{(2*|T'|+|T^P|-|T_S|)*|T_S|}{n}$$

$$= \frac{(|T_S|+|T^P|)*|T_S|}{|T^U|}, \text{ since } |T'| = |T_S|$$

$$P_{loss}(T_S, T'+T^P)$$

$$= \frac{r}{|T'|+|T^P|} * \frac{(|T_S|+|T^P|)*|T_S|}{|T^U|}$$

$$= r * \frac{|T_S|}{|T^U|}$$

If the dataset distribution of T_S is not even, in Region y, at least a t_i in T_S has number of counts as y and another one has number of counts as 0. Let's consider the extreme cases of y shown in Figure 6-3(b), called the "flattest-distribution" case in this thesis and Figure 6-3(c), called the "narrowest-distribution" case in this thesis. In Figure

[24] See Figure 6-3(a).

6-3(b), all t_i in T_S has the number of counts x or $(x+1)$. If $\frac{m}{n}$ of the t_i in T_S has counts $(x+1)$ and the rest have counts x, we have:

$$\sum_{t'\in(T'+T^P)}\sum_{t\in T_S} M(t,t')$$

$$= (x+1)*z*m + x*(1+z)*(n-m)$$

$$= \frac{|T_S|+(n-m)}{n} * \frac{|T'|+|T^P|-(n-m)}{n} * m + \frac{|T_S|-m}{n} * \frac{|T'|+|T^P|+m}{n} * (n-m)$$

$$= \frac{|T_S|^2 + |T_S|*|T^P| - m*(n-m)}{n}, \text{ since } |T'| = |T_S|$$

$$= \frac{|T_S|*(|T_S|+|T^P|) - m*(|T^U|-m)}{|T^U|}$$

$$P_{loss}(T_S, T'+T^P)$$

$$= \frac{r}{|T'|+|T^P|} * \frac{|T_S|*(|T_S|+|T^P|) - m*(|T^U|-m)}{|T^U|}$$

$$= r * \frac{|T_S|*(|T_S|+|T^P|) - m*(|T^U|-m)}{|T^U|*(|T_S|+|T^P|)}$$

$$= r*\left(\frac{|T_S|}{|T^U|} - \frac{m*(|T^U|-m)}{|T^U|*(|T_S|+|T^P|)}\right)$$

For this case, $1 \le m \le (|T^U|-1)$ and $|T^P| \ge 0$, such that:

$$\because \quad \frac{\Delta P_{loss}(T_S, T'+T^P)}{\Delta m} = r * \frac{2*m-|T^U|}{|T^U|*(|T_S|+|T^P|)})$$

$$\therefore \quad P_{loss}(T_S, T'+T^P) \text{ is decreasing in the range } [1, \frac{|T^U|}{2}] \text{ and increasing in}$$

the range $[\frac{|T^U|}{2}, |T^U|-1]$.

Therefore, $P_{loss}(T_S, T'+T^P)$ is ranged from $r*(\frac{|T_S|}{|T^U|} - \frac{|T^U|}{4*(|T_S|+|T^P|)})$ to $r*(\frac{|T_S|}{|T^U|} - \frac{|T^U|-1}{|T^U|*(|T_S|+|T^P|)})$ [25]. If $|T_S| >> |T^U|$, $\frac{|T^U|}{4*(|T_S|+|T^P|)}$ and $\frac{|T^U|-1}{|T^U|*(|T_S|+|T^P|)}$ are relatively small comparing with $\frac{|T_S|}{|T^U|}$, then $P_{loss}(T_S, T'+T^P)$ can be estimated as $r*\frac{|T_S|}{|T^U|}$ in this case.

On the other hand, Figure 6-3(b) shows one t_i in T_S has the number of counts $(x+y)$ and the rest have counts of x. If $\frac{n-1}{n}$ of the t_i in T_S have counts x, we have:

$$\sum_{t'\in(T'+T^P)} \sum_{t\in T_S} M(t,t')$$

$$= (x+y)*z + x*(y+z)*(n-1)$$

$$= \frac{|T_S|-y}{n} * z + y*z + \frac{|T_S|-y}{n} * \frac{|T'|+|T^P|-z}{n-1} * (n-1)$$

$$= \frac{(|T_S|-y)*(|T_S|+|T^P|)}{n} + y*z \text{, since } |T'| = |T_S|$$

$$= \frac{(|T_S|-y)*(|T_S|+|T^P|) + y*(|T_S|+|T^P|-n*y+y)}{n},$$

since $z = \frac{|T_S|+|T^P|-(n-1)*y}{n}$

$$= \frac{|T_S|*(|T_S|+|T^P|) - y^2*(|T^U|-1)}{|T^U|}$$

$$P_{loss}(T_S, T'+T^P)$$

[25] Assume $|T^U| \geq 2$ for all analysis in this thesis.

$$= \frac{r}{|T'|+|T^P|} * \frac{|T_S|*(|T_S|+|T^P|)-y^2*(|T^U|-1)}{|T^U|}$$

$$= r * \frac{|T_S|*(|T_S|+|T^P|)-y^2*(|T^U|-1)}{|T^U|*(|T'|+|T^P|)}$$

$$= r*(\frac{|T_S|}{|T^U|}-\frac{y^2*(|T^U|-1)}{|T^U|*(|T_S|+|T^P|)})$$

For this case, $1 \le y \le |T_S|$ and $|T^P| \ge 0$, such that:

$$\because \quad \frac{\Delta P_{loss}(T_S, T'+T^P)}{\Delta y} = -r * \frac{2*y*(|T^U|-1)}{|T^U|*(|T_S|+|T^P|)}$$

$$\therefore \quad P_{loss}(T_S, T'+T^P) \text{ is strictly decreasing in the range } [1, |T^U|].$$

Therefore, $P_{loss}(T_S, T'+T^P)$ is ranged from $r*(\frac{|T_S|}{|T^U|}-\frac{|T_S|^2*(|T^U|-1)}{|T^U|*(|T_S|+|T^P|)})$ to $r*(\frac{|T_S|}{|T^U|}-\frac{|T^U|-1}{|T^U|*(|T_S|+|T^P|)})$.

For the other cases, they can be considered as some transition stages from Figure 6-3(c) to Figure 6-3(b) (or Figure 6-3(a) for the even-distribution case) by transferring some counts d from a higher frequent dataset t_h to a lower frequent dataset t_l at a time such that the count of t_h is still higher than that of t_l[26]. Assuming C_h and C_l are the counts of t_h and t_l before any transfer, and P_{before} and P_{after} are the privacy loss before and after the change, we have:

$$P_{after}$$

$$= P_{before} - C_h*(q-C_h) - C_l*(q-C_l)$$

[26] See Figure 6-3(d) as an example.

$$+ (C_h - d) * (q - (C_h - d)) + (C_l + d) * (q - (C_l + d))$$

$$= P_{before} - d * q + 2 * d * C_h - d^2 + d * q - 2 * d * C_l - d^2$$

$$= P_{before} + 2 * d * (C_h - C_l) - 2 * d^2$$

$$\because \quad 1 \le d \le \frac{(C_h - C_l)}{2} \text{ and } (C_h - C_l) \ge 2$$

$$0 \le \frac{\Delta P_{after}}{\Delta d} = 2 * (C_h - C_l) - 4 * d \le 2 * (C_h - C_l) - 4$$

$\therefore$ P_{after} is increasing in the range [1, $\frac{(C_h - C_l)}{2}$] where $P_{after} = P_{before}$ when $d = 0$, so that $P_{after} > P_{before}$.

For all the cases with the same $|T^U|$, $|T_S|$ and $|T^P|$, each transition step moving from a narrower-case T_{before} with P_{before} towards a flatter-case T_{after} with P_{after} increases the privacy loss. If we consider the variance[27] of dataset counts of a data table (denoted as $Var(T)$ in this thesis) of those cases with the same average count $\hat{C}$, and define the variances of T_{before} and T_{after} as $Var(T_{before})$ and $Var(T_{after})$, we have:

$$Var(T_{after})$$

$$= Var(T_{before}) - (C_h - \hat{C})^2 - (C_l - \hat{C})^2 + ((C_h - d) - \hat{C})^2 + ((C_l + d) - \hat{C})^2$$

$$= Var(T_{before}) - 2 * d * (C_h - C_l) + 2 * d^2,$$

which is similar to relation between P_{before} and P_{after}; therefore, with the same $|T^U|$, $|T_S|$ and $|T^P|$, larger variance of the dataset counts implies less privacy loss.

---

[27] $Var(X) = \sum_{X_i \in X} (X_i - \hat{X})^2$

For the data complementation approach, we conclude that $P_{loss}(T_S, T'+T^P)$ is ranged from $r*(\frac{|T_S|}{|T^U|} - \frac{|T_S|^2 *(|T^U|-1)}{|T^U|*(|T_S|+|T^P|)})$ to $r*\frac{|T_S|}{|T^U|}$, where r is the number of records lost. Other than $|T_S|$, $|T^U|$ and $|T^P|$, $Var(T_S)$ is another factor determining the privacy loss from the lost dataset. Since $|T_S|$, $|T^U|$, $Var(T_S)$ are totally dependent upon the pool of sample datasets and the domains of a dataset entry, $|T^P|$ is the most flexible factor that can be controlled by a better definition of the UNREALIZED TRAINING-SET function as shown in Figure 5-1. To diminish the privacy loss, $|T^P|$ has to be minimized. If each dataset in T' is taken from the most frequent dataset of T^P (by modifying the statement "$t'_i \leftarrow$ a dataset in T^P" to "$t'_i \leftarrow$ the most frequent dataset in T^P" in the UNREALIZED TRAINING-SET function,) T^P always maintains the greatest variety of datasets, such that the lowest number of universal set must be added during the unrealized process. As a result, $|T^P|$ is minimized because $|T^P| = |qT^U| - 2*|T_S|$ as q is minimized, such that the minimum value of $|T^P|$ is $\left\lceil \frac{2*|T_S|}{|T^U|} \right\rceil * |T^U| - 2*|T_S|$ if $\frac{2*|T_S|}{|T^U|} \geq c$, or $(c*|T^U| - 2*|T_S|)$ otherwise, where c is the count of the most frequent datasets in T_S[28].

[28] See Lemma 6.1.

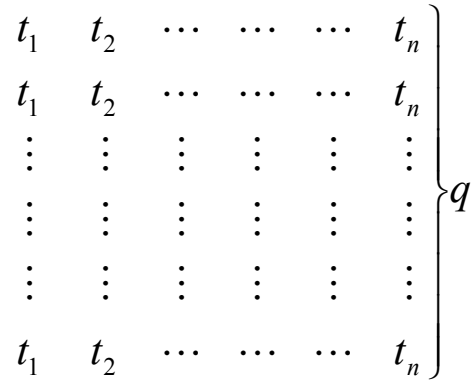

Figure 6-1(a) Distributing datasets in qT^U by dataset value.

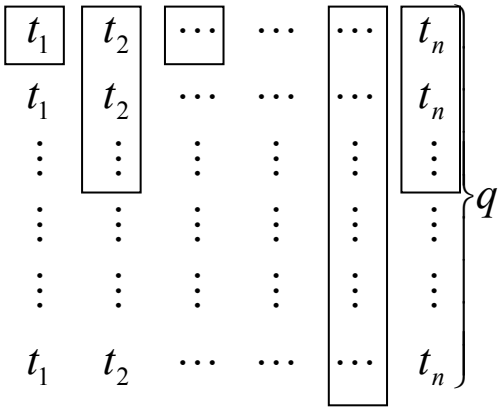

Figure 6-1(b) Datasets of T_S are contained in the rectangles.

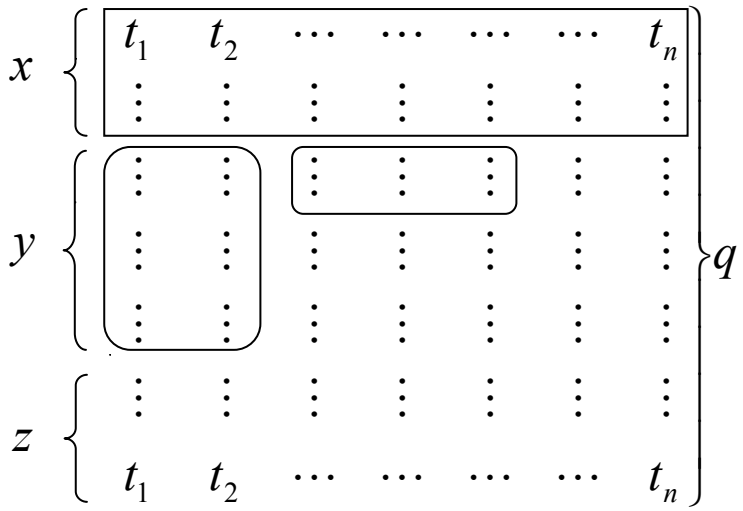

Figure 6-2 Rearranged datasets in T_S according to their number of counts.

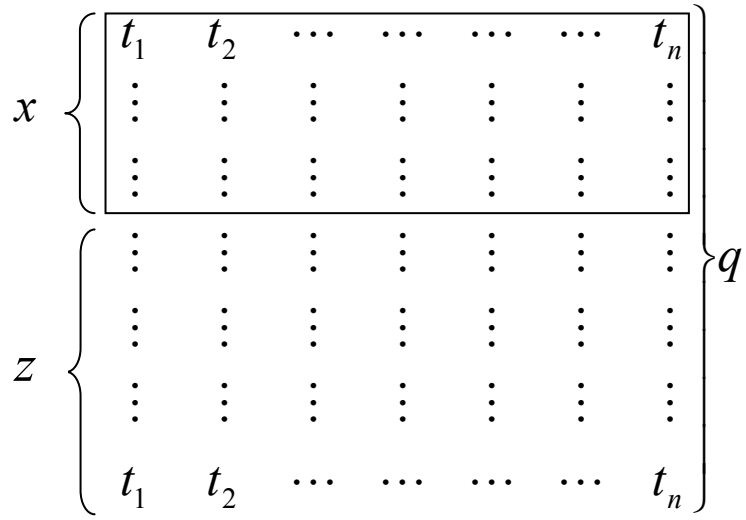

Figure 6-3(a) The even-distribution case of T_S (y = 0).

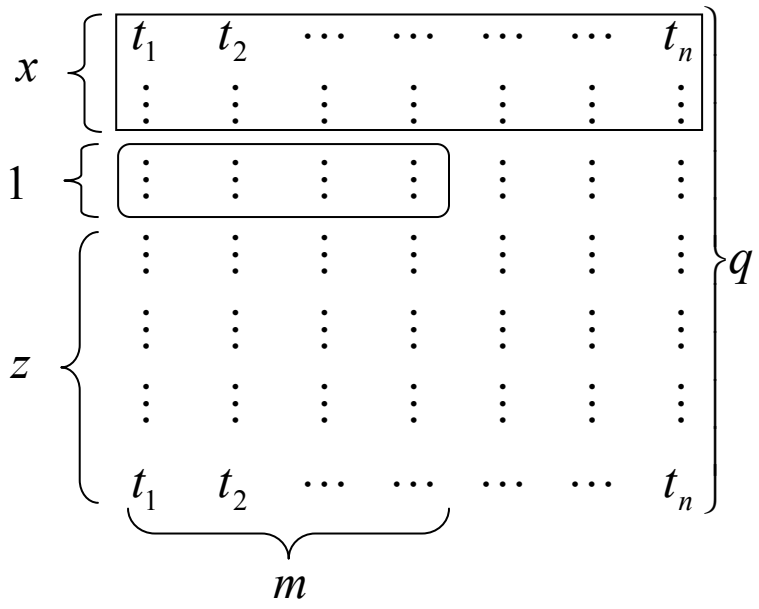

Figure 6-3(b) The flattest-distribution case of T_S ($y = 1$).

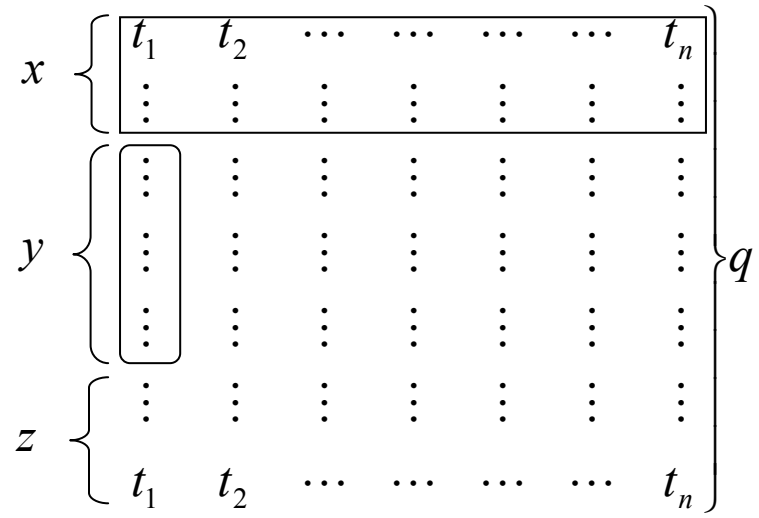

Figure 6-3(c) The narrowest-distribution case of T_S ($y = |T_S| - x * n$).

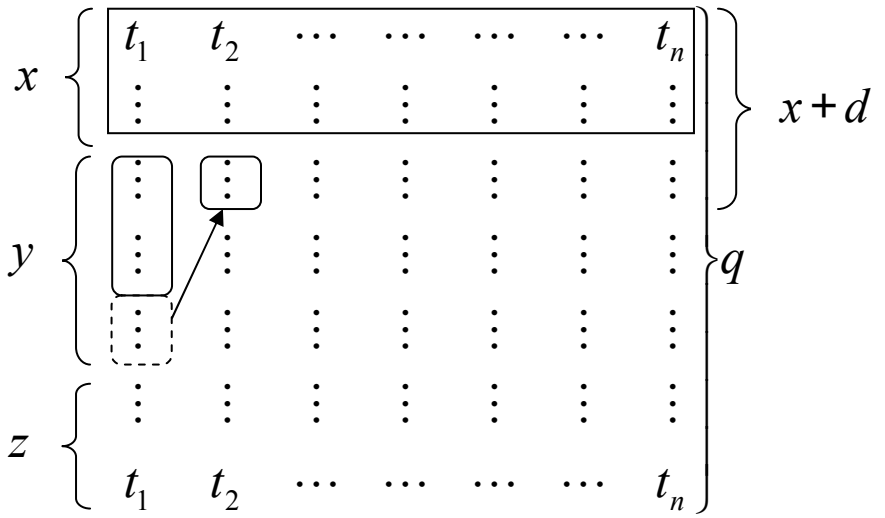

Figure 6-3(d) Transferring counts from a higher frequent dataset to a lower one.

Lemma 6-1:

From Figure 6-2, we find:

$$q = x + y + z$$

$$\Rightarrow \quad \frac{2*|T_S| + |T^P|}{|T^U|} = x + y + z \text{, by Lemma 5-5}$$

$$\Rightarrow \quad |T^P| = z*|T^U| + (x + y)*|T^U| - 2*|T_S|$$

As positive integer $z \geq 0$ and $(x + y)$ is the counts of the most frequent datasets in T_S, to minimize positive integer $|T^P|$, $z = \left\lceil \frac{2*|T_S|}{|T^U|} \right\rceil - (x + y)$ if $\frac{2*|T_S|}{|T^U|} \geq (x + y)$, or 0 otherwise. Therefore, $|T^P| =$

$$\left\lceil \frac{2*|T_S|}{|T^U|} \right\rceil * |T^U| - 2*|T_S| \quad \text{if} \quad \frac{2*|T_S|}{|T^U|} \geq (x+y) \quad , \quad \text{or} \quad ((x+y)*|T^U| - 2*|T_S|)$$

otherwise.

6.1.2 Privacy Loss on Low Variance Cases

In section 6.1.1, we found that privacy preservation of the sample datasets relies on $|T_S|$, $|T^U|$, $|T^P|$ and $Var(T_S)$. While $|T^U|$ is a constant because it is based on the domain of a sample dataset, $|T_S|$ is dependent upon the statistical purposes and $|T^P|$ is proven as controllable, $Var(T_S)$ is an uncontrollable variable that drives the privacy preserving result, whereas a higher variance preserves more privacy. The worst cases are the ones shown in Figures 6-3(a) and Figure 6-3(b), in which P_{loss} approaches $r * \frac{|T_S|}{|T^U|}$ (which is equal to $r * x$ where is the average counts of all datasets in T_S.) Therefore, samples with low variance do not protect privacy well, as its P_{loss} is similar to that of unprotected samples with even-distribution (i.e. the best case of unprotected samples).

6.1.3 Privacy Issue on Low Frequency Datasets

Another privacy issue is related to some uncommon datasets in T_S with an extremely low number of counts as compared with the other datasets. Figure 6-4 shows an extreme example of this case, where T_S contains m datasets with counts $(x+y)$ and the rest with counts x. If we compare the probabilities of getting a high frequency dataset (such as t_1) in the original datasets and in the sanitized datasets, we have:

$$P(t_1 \mid T_S)$$

$$= \frac{x+y}{|T_S|}$$

$$= \frac{|T_S| + (n-m)*y}{|T^U|*|T_S|}$$

$$= \frac{1}{|T^U|} + \frac{(n-m)*y}{|T^U|*|T_S|}$$

$$P(t_1 \mid [T' + T^P])$$

$$= \frac{z}{|T'| + |T^P|}$$

$$= \frac{|T_S| + |T^P| - (n-m)*y}{|T^U|*(|T_S| + |T^P|)}, \text{ since } |T_S| = |T'|$$

$$= \frac{1}{|T^U|} - \frac{(n-m)*y}{|T^U|*(|T_S| + |T^P|)}$$

$$P(t_1 \mid [T' + T^P]) - P(t_1 \mid T_S)$$

$$= -\frac{(n-m)*y}{|T^U|*(|T_S| + |T^P|)} - \frac{(n-m)*y}{|T^U|*|T_S|}$$

$$= -\frac{(2*|T_S| + |T^P|)*(n-m)*y}{|T^U|*|T_S|*(|T_S| + |T^P|)}$$

Similarly, for a low frequency dataset (such as t_n), we have:

$$P(t_n \mid [T' + T^P]) - P(t_n \mid T_S)$$

$$= \frac{(2*|T_S| + |T^P|)*m*y}{|T^U|*|T_S|*(|T_S| + |T^P|)}$$

Conclusively, the probability of getting a high frequency dataset in the sanitized datasets decreases while that of a low frequency dataset increases. If m and y are larger, the

probability changes will obviously be large. In the same manner, privacy of an extremely low frequency dataset originally is potentially safe but much easier to be leaked out in the sanitized datasets. In other words, if the information provided by person *A* is uncommon when compared with that of others, person *B* has little chance to retrieve his / her information by randomly obtaining datasets from the original samples; however, among all datasets in the sanitized datasets, the one from *A* has the highest possibility to be retrieved by *B*.

The privacy issue of low frequency datasets can be improved, if there are some datasets contained by T^U but not T_S. Figure 6-5 shows an extreme example of this case, which is similar to the case shown in Figure 6-4 but with l datasets in T^U not contained by T_S. For the least frequent dataset t_{n-l} in T_S, we have:

$$P(t_{n-l} \mid T_S)$$

$$= \frac{x}{|T_S|}$$

$$= \frac{|T_S| - m * y + l * x}{|T^U| * |T_S|}$$

$$= \frac{1}{|T^U|} + \frac{l * x - m * y}{|T^U| * |T_S|}$$

$$P(t_{n-l} \mid [T' + T^P])$$

$$= \frac{y + z}{|T'| + |T^P|}$$

$$= \frac{|T_S| + |T^P| - l * x + m * y}{|T^U| * (|T_S| + |T^P|)}, \text{ since } |T_S| = |T'|$$

$$= \frac{1}{|T^U|} - \frac{l*x - m*y}{|T^U|*(|T_S| + |T^P|)}$$

$$P(t_{n-l} | [T' + T^P]) - P(t_{n-l} | T_S)$$

$$= -\frac{l*x - m*y}{|T^U|*(|T_S| + |T^P|)} - \frac{l*x - m*y}{|T^U|*|T_S|}$$

$$= \frac{(2*|T_S| + |T^P|)*(m*y - l*x)}{|T^U|*|T_S|*(|T_S| + |T^P|)}$$

$$\because \quad x = \frac{|T_S| - m*y + l*x}{|T^U|}$$

$$\therefore \quad x = \frac{|T_S| - m*y}{|T^U| - l}$$

If $m*y - l*x \le 0,$

then $m*y - l*\frac{|T_S| - m*y}{|T^U| - l} \le 0$

$$\Rightarrow \quad \frac{|T^U|}{|T_S|}*m*y \le l$$

With some datasets with 0 count in T_S, the probability of getting the least frequency dataset in the sanitized datasets decreases from $\frac{1}{|T^U|} + \frac{m*y}{|T^U|*(|T_S| + |T^P|)}$ to $\frac{1}{|T^U|} - \frac{l*x - m*y}{|T^U|*(|T_S| + |T^P|)}$. Furthermore, if $\frac{|T^U|}{|T_S|}*m*y \le l$, the probability of getting a low frequency dataset in the sanitized datasets will be lower than the probability of getting the dataset from the original samples.

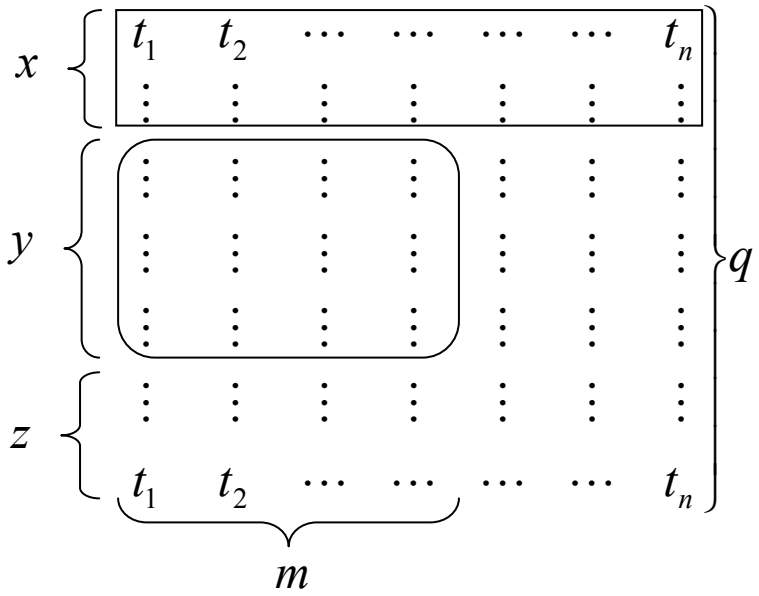

Figure 6-4 A typical example that T_S has some extremely low frequency datasets.

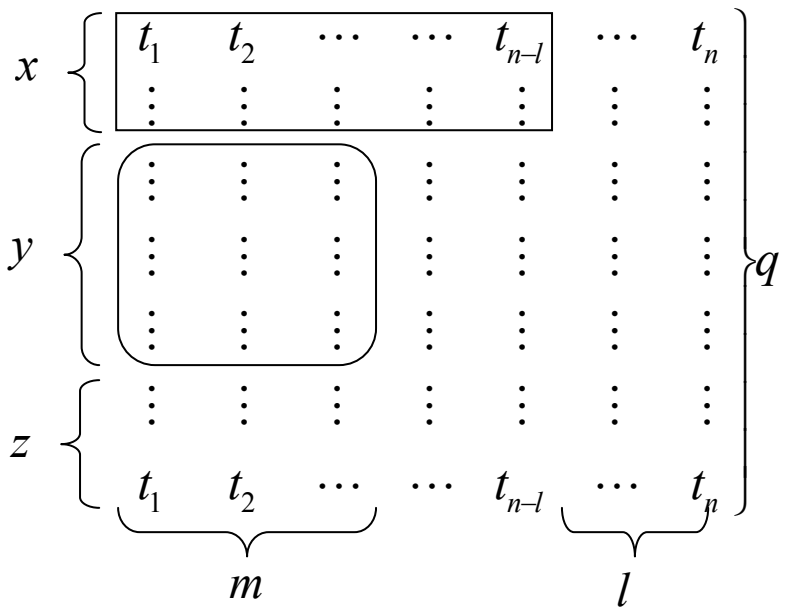

Figure 6-5 A typical example that T_S has some datasets with extremely low counts and some datasets with 0 counts.

6.2 Create Dummy Attribute / Attribute Values

In section 6.1, we discussed two potential privacy issues of the dataset complementation approach. Firstly, privacy does not preserve well if the variance of the dataset counts (including the datasets with count 0) in T_S is low. Secondly, dataset complementation increases the possibility of privacy loss for those datasets with extremely low counts (albeit with values larger than 0) as compared to the other datasets. In this section, we introduce an approach to solve both issues.

In section 6.1.2, we found that if T_S does not contain some number l of datasets in T^U, the problem can be improved. Furthermore, if $l \geq \frac{|T^U|}{|T_S|} * m * y$, the problem would be solved. If $l = |T^U|$, we have:

$$l = \frac{y^* | T^U |^2}{y^* | T^U |}$$

$$\Rightarrow \quad l \geq \frac{y^* | T^U |^2}{| T_S | + y^* | T^U |}$$

$$\Rightarrow \quad l \geq \frac{| T^U |}{| T_S |} * (| T^U | - l) * y$$

$$\Rightarrow \quad l \geq \frac{| T^U |}{| T_S |} * m * y \text{, since } m < (| T^U | - l)$$

Therefore, if we can expand the domain of a sample dataset to (at least) double the size of the universal set T_υ^U such that $| T_\upsilon^U | \geq 2^* | T^U |$, then $| T^U |$ of the datasets in T_υ^U will have 0 counts in T_S. There are two ways to expand the domain of a sample: firstly, we can pick up an attribute and double its possible values, e.g. expanding the possible values of attribute *Wind* from $\{Strong, Weak\}$ to $\{Strong, Weak, Crazy, Deathly\}$ where *Crazy* and *Deadly* are dummy attribute values that are not selectable by anyone during the data collection process; secondly, we can add an dummy attribute with two possible values to the sample domain while one value is always selected and the other is never, e.g. adding an attribute $TrueValue = \{True, False\}$ in the data table in Table 3-1 where all values of *TrueValue*[29] equal *True*. In keeping with convention, all the datasets with attribute values that would be never selected are called dummy datasets (and belong to the dummy set) in this thesis.

[29] The attribute will be never chosen as the test attribute because it always has information gain 0.

Since $|T'| = |T_S|$ and c are always larger or equal to $\frac{|T_S|}{|T^U|}$, $|T^P|$ can be controlled[30] to be as small as $(2*c*|T^U|-2*|T_S|)$ where c is the count of the most frequent datasets in T_S. After the expansion, the sanitized datasets, which contain $c*|T^U|$ more dummy datasets, are always larger than $|T_S|$. If we change the UNREALIZED TRAINING-SET to make $|T'|$ barely contain some dummy datasets,[31] $|T^P|$ will be equal to its smallest value $(2*c*|T^U|-2*|T_S|)$. As a result, T' and T^P will not contain any of the most frequent datasets in T_S.

By using UNREALIZED TRAINING-SET, the even-distribution cases shown in Figure 6-3(a) will have n datasets with the count y and the rest of n datasets with the count 0 in T_S[32]. In this case, $|T^P| = (2*q*|T^U|-2*|T_S|) = 0$ and $P_{loss}(T_S, T'+T^P)$ is equal to 0.

For the narrowest case[33], we have:

$$\sum_{t'\in(T'+T^P)}\sum_{t\in T_S} M(t,t')$$

$$= x*y*(n-1)$$

$$= \frac{|T_S|-y}{n}*y*(n-1)$$

$$= \frac{(|T_S|-y)*y*(|T^U|-1)}{|T^U|}$$

---

[30] See Lemma 6-1.

[31] See Figure 6-6.

[32] See Figure 6-7(a).

[33] See Figure 6-7(b).

$$P_{loss}(T_S, T'+T^P)$$

$$= \frac{r}{|T_S|+|T^P|} * \frac{(|T_S|-y)*y*(|T^U|-1)}{|T^U|}$$

$$= r * \frac{(|T_S|-y)*y*(|T^U|-1)}{|T^U|*(2*q^*|T^U|-|T_S|)}, \text{ since } |T^P| = (2*q^*|T^U|-2^*|T_S|)$$

$$\because \quad \frac{\Delta P_{loss}(T_S, T'+T^P)}{\Delta y} = r * \frac{(|T^U|-1)*[2*y^2*(1-|T^U|)-2*y^*|T_S|+|T_S|^2]}{|T^U|*(2*y*(|T^U|-1)+|T_S|)^2}$$

$\therefore$ the absolute extrema of $P_{loss}(T_S, T'+T^P)$ is located at the point when $y =$

$$0, \ \frac{|T_S|*(\sqrt{|T^U|}-1)}{2*(|T^U|-1)}, \ \frac{-|T_S|*(\sqrt{|T^U|}+1)}{2*(|T^U|-1)} \text{ (rejected) or } |T_S|.$$

Therefore, $P_{loss}(T_S, T'+T^P) \le r * \frac{|T_S|*(|T^U|-\sqrt{2|T^U|-1})}{2*|T^U|*(|T^U|-1)}$ where $y = \frac{|T_S|*(\sqrt{|T^U|}-1)}{2*(|T^U|-1)}$.

(Notes: $P_{loss}(T_S, T'+T^P) \le r * \frac{|T_S|*(R|T^U|-2\sqrt{(R|T^U|-1)(R-1)}+R-2)}{|T^U|*R^2(|T^U|-1)}$ where $R \ge 2$.)

Based on the case of Figure 6-7(b) with $y = x$, if a dataset has d more counts[34], the change of $P(T_S, T'+T^P)$ will be $(y+d)*(y-d)-y^2 = -d^2$, which means the privacy loss will be less after the change. In addition, the same result will occur if a dataset has d less counts[35]. For the other cases, we can consider that they have some datasets with more or less counts as compared to the case of Figure 6-7(b) with $y = x$.

Hence, in overall, $P_{loss}(T_S, T'+T^P) \le r * \frac{|T_S|*(|T^U|-\sqrt{2|T^U|-1})}{2*|T^U|*(|T^U|-1)}$, such that the privacy issue discussed in section 6.1.1 is solved.

[34] See Figure 6-7(c).

[35] See Figure 6-7(d).

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

inputs: T_S, a set of input sample datasets

T^U, a universal set

T', a set of output training datasets

T^P, a set of unreal datasets

if T_S is empty **then**

return $<T', T^P>$

$t_i \leftarrow$ a dataset in T_S

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

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

$t'_i \leftarrow$ a dataset in T^P that belongs to the dummy set

else

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

$t'_i \leftarrow$ a dataset in T^P that belongs to the dummy set

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

Figure 6-6 Pseudocode of modified unrealized training set algorithm.

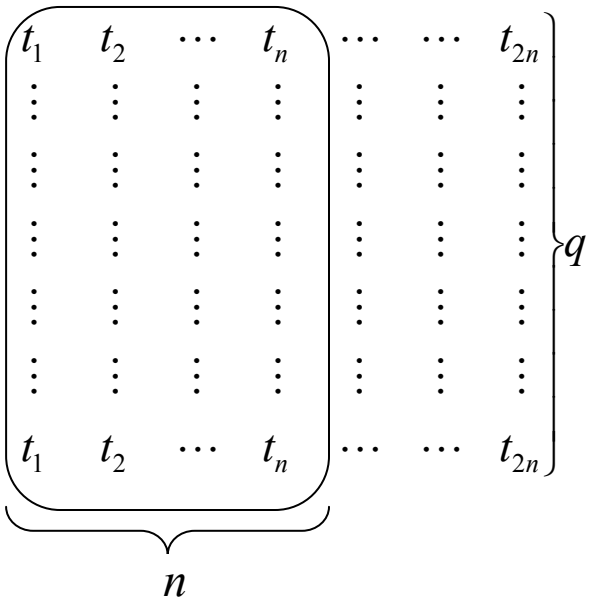

Figure 6-7(a) A modified version of Figure 6-3(a) with T_S has *n* zero-count datasets.

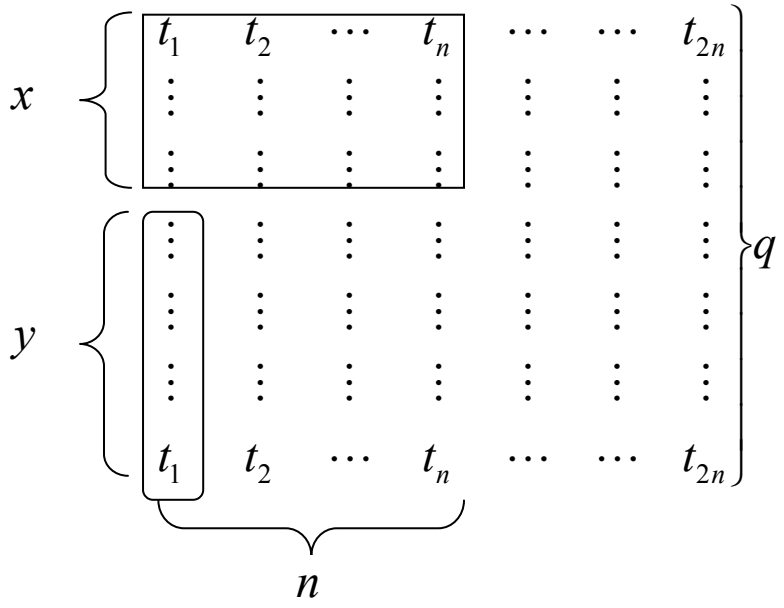

Figure 6-7(b) A modified version of Figure 6-3(c) with T_S has *n* zero-count datasets.

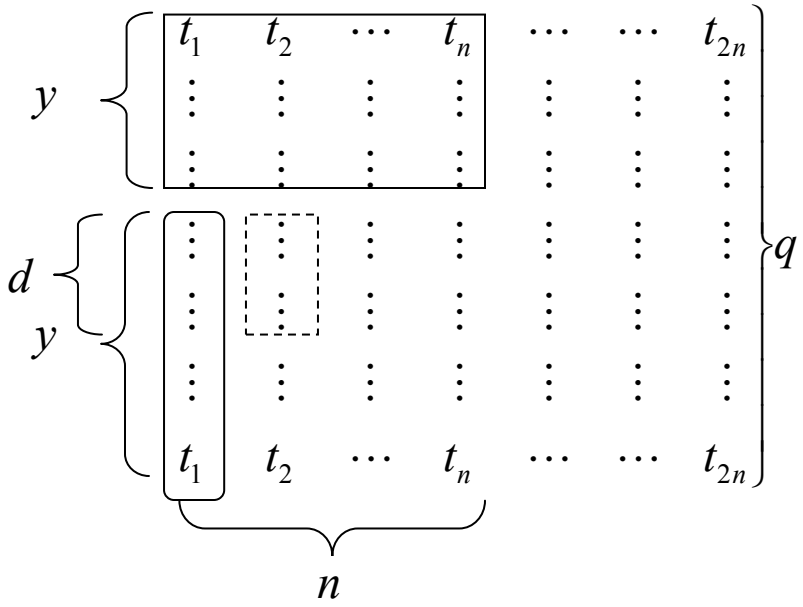

Figure 6-7(c) A modified version of Figure 6-8(b) with a dataset has counts $(y + d)$.

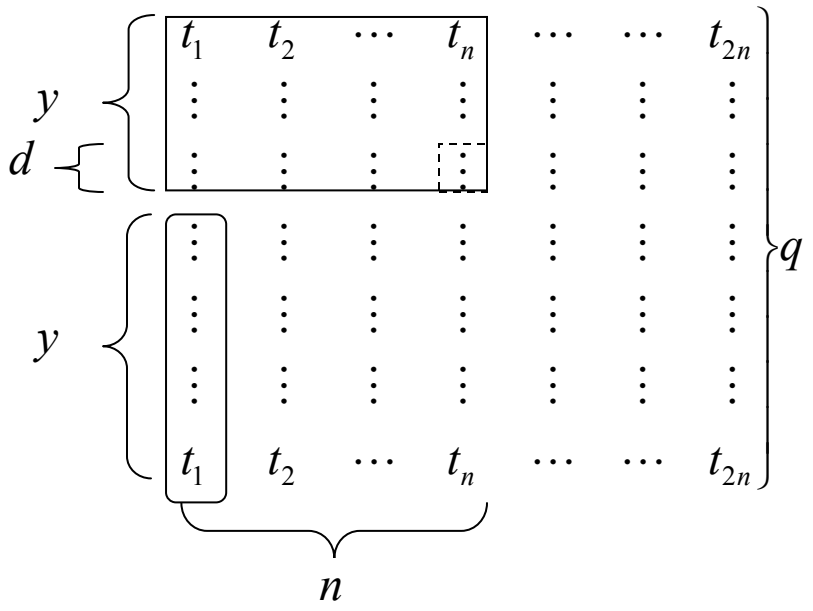

Figure 6-7(d) A modified version of Figure 6-8(b) with a dataset has counts $(y - d)$.

6.3 Storage Requirement

In Chapter 6.2, we developed an improved dataset complementation method for privacy preservation of the training samples. However, it requires storage $|T'| + |T^P| = |T_S| + (2 * c^* |T^U| - 2^* |T_S|) = (2 * c^* |T^U| - |T_S|)$, where c is the count of the most frequent dataset in T_S. If all datasets in T_S are the same, the storage requirement is $(2^* |T_S|^* |T^U| - |T_S|) = (2^* |T^U| - 1)^* |T_S|$, which is $(2^* |T^U| - 1)$ times[36] of the storage required for the original datasets. Usually, c is an unknown if we preserve the privacy of the samples from the beginning of the data collection process, such that $(2^* |T^U| - 1)$ times of the storage is required for the worst case.

[36] This is the worst case.

6.4 Complexity

For the function UNREALIZED TRAINING-SET in Figure 5-2, the time complexity is $O(|T_S|)$. However, we need to prepare a universal set as an input of this function. To build a universal set we must expand all possible values of all attributes, with the result that the complexity is $O(\hat{V}_A^{|A|})$ where $|A|$ is the number of attributes and $\hat{V}_A$ is the average of the number of possible values of each attribute. Therefore, the total complexity is $O(|T_S| + \hat{V}_A^{|A|})$.

If we take the improved dataset complementation method described in Chapter 6.2, the complexity of the function UNREALIZED TRAINING-SET is still $O(|T_S|)$ if the dummy datasets in T^P are stored separately for efficiency. Meanwhile, the number of attributes is doubled such that the complexity of building a universal set becomes $O(2 * \hat{V}_A^{|A|})$. As a result, the total complexity is $O(|T_S| + 2 * \hat{V}_A^{|A|})$. Based on the scope of this thesis, the number of samples taken should be much larger than size of a universal set, such that $|T_S|$ is the main factor that dominates the time complexity as $O(|T_S|)$.

Chapter 7

CONCLUSION AND FUTURE WORKS

Privacy preservation in datamining activities is of significant importance for many applications. However, the privacy preserving process sometimes reduces the utility of training datasets, which causes inaccurate datamining results. Privacy preservation approaches focus on different areas of a datamining process, and datamining methods also vary. This thesis focuses on privacy protection of the training samples applied for decision tree datamining.

This thesis presents a new privacy preserving approach via dataset complementation, which removes each sample from a set of perturbing datasets[37]. During the privacy preserving process, this set of perturbed datasets is dynamically modified. As the sanitized version of the original samples, these perturbed datasets are stored to enable a modified decision tree datamining method. This method guarantees to provide the same datamining outcomes as the originals, which is proved mathematically and by a test using one set of sample datasets in this thesis.

From the viewpoint of privacy preservation, the original datasets can only be reconstructed in their entirety if someone has all perturbed datasets, which is not supposed to be the case for an unauthorized party. If someone gets r of the sanitized datasets, the matching rate of the originals is from $r*(\frac{|T_S|}{|T^U|}-\frac{|T_S|^2*(|T^U|-1)}{|T^U|*(|T^{P'}|)})$ to

[37] The perturbing datasets include an unrealized training set and a perturbing set in the main content of this thesis.

$r * \frac{|T_S|}{|T^U|}$ where $|T_S|$, $|T^{P'}|$ and $|T^U|$ are the sizes of the sample, sanitized and universal datasets. In cases with a small matching rate, the matching rate of those sample datasets with low frequency is high. On the other hand, for those cases without such a problem, their matching rate approaches $r * \frac{|T_S|}{|T^U|}$, which is the best case if all samples are not protected.

This thesis also sets out an improved version of the dataset complementation approach. It ensures that the matching rate is bounded between 0 and $r * \frac{|T_S| * (|T^U| - \sqrt{2|T^U| - 1})}{2 * |T^U| * (|T^U| - 1)}$ by doubling the domain of a dataset. Therefore, this improved approach results in a matching rate that is always less than one-third of the best case of the unprotected samples. In all cases, the complexity of the sanitization process is $O(|T_S|)$. However, the worst case requires $(2 * |T^U| - 1)$ times the amount of storage needed for unprotected samples.

The privacy preserving approach discussed in this thesis would not function if all training datasets were leaked, because the dataset reconstruction algorithm is generic. Therefore, further research is required to eliminate this limitation. Future research should also explore means to reduce the storage requirement associated with the derived dataset complementation approach. This thesis relies on theoretical proofs with limited practical tests, so testing with real samples should be the next step to gain solid ground on real-life application.

Bibliography

[1] BBC News (2007, November 27) *Brown apologises for records loss.* Retrieved September 12, 2008, from http://news.bbc.co.uk/2/hi/uk_news/politics/7104945.stm

[2] Lomas, N. (2008, August 25) *Data on 84,000 U.K. prisoners is lost.* Retrieved September 12, 2008, from http://news.cnet.com/8301-1009_3-10024550-83.html

[3] Kaplan, D. (2007, May 9) *Hackers steal 22,000 Social Security numbers from University of Missouri database.* Retrieved September 12, 2008, from http://www.scmagazineus.com/Hackers-steal-22000-Social-Security-numbers-from-University-of-Missouri-database/article/34964/

[4] Goodin, D. (2007, September 15) *Hackers infiltrate TD Ameritrade client database.* Retrieved September 12, 2008, from http://www.channelregister.co.uk/2007/09/15/ameritrade_database_burgled/

[5] *Set (mathematics), (2008, September 20).* Retrieved September 21, 2008, from Wikipedia, http://en.wikipedia.org/wiki/Set_(mathematics)

[6] *Tuple, (2008, September 20).* Retrieved September 21, 2008, from Wikipedia, http://en.wikipedia.org/wiki/Tuple

[7] *Tree (graph theory), (2008, August 14).* Retrieved September 21, 2008, from Wikipedia, http://en.wikipedia.org/wiki/Tree_graph

[8] Quinlan, J.R. (1986). Induction of Decision Trees. *Machine Learning,* 1 (1), 81-106.

[9] Russell, S. & Peter, N. (2002). *Artificial Intelligence. A Modern Approach*, 2. India: Prentice-Hall.

[10] Mitchell, T. (1997). *Machine Learning*, 1. New York: McGraw-Hill.

[11] *Decision Tree Learning. (2008, August 08).* Retrieved September 15, 2008, from Wikipedia, http://en.wikipedia.org/wiki/Decision_tree_learning

[12] *Entropy (Information Entropy). (2008, September 18).* Retrieved September 15, 2008, from Wikipedia, http://en.wikipedia.org/wiki/Information_entropy

[13] Aggarwal, C. & Yu, P. (2008). *Privacy-Preserving Data Mining:, Models and Algorithms*. New York: Springer.

[14] Sweeney, L. (2002, May). k-Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10, 557-570.

[15] Meyerson, A. & Williams, R. (2004). On the Complexity of Optimal K-Anonymity. *Symposium on Principles of Database Systems. Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 223-228.

[16] Byun, J., Kamra, A., Bertino, E. & Li, N. (2007). Efficient k-Anonymization using Clustering Techniques. *Lecture Notes in Computer Science*, 4443, 188-200.

[17] Kargupta, H., Datta, S., Wang, Qi. & Sivakumar, K. (2003). On the Privacy Preserving Properties of Random Data Perturbation Techniques. *Proceedings of the Third IEEE International Conference on Data Mining.* 99-106.

[18] Dowd, J., Xu, S. & Zhang, W. (2006). Privacy-Preserving Decision Tree Mining Based on Random Substitions. *Proceedings of the 2006 International Conference on Emerging Trends in Information and Communication Security (ETRICS'06), LNCS 3995*, 145-159.

[19] Bu S., Lakshmanan L., Ng R. & Ramesh G. (2007, April). Preservation Of Patterns and Input-Output Privacy. *Proceedings of the twenty-third IEEE International Conference on Data Engineering.* 696-705.

[20] Church, A. (1974). Set Theory with a Universal Set. *Proceedings of the Tarski Symposium. Proceedings of Symposia in Pure Mathematics*, 25, 297-308.

[21] *Multiset, (2008, September 19)*. Retrieved September 21, 2008, from Wikipedia, http://en.wikipedia.org/wiki/Multiset.

[22] Chawla, S., Dwork, C., McSherry, F., Smith, A. and Wee, H. (2005). Toward Privacy in Public Databases, In *2nd IACR Theory of Cryptography Conference* -TCC 2005, 363–385.