

# A BORDER BASED APPROACH FOR HIDING FUZZY WEIGHTED SENSITIVE ITEMSETS

A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*

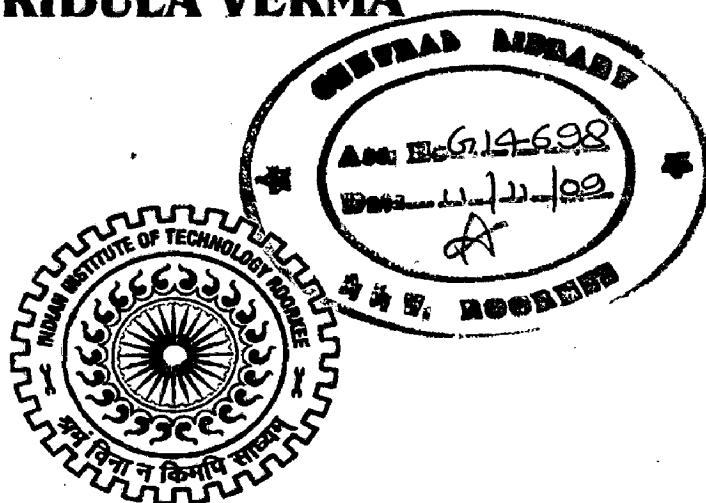
**MASTER OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

By

**MRIDULA VERMA**



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE  
ROORKEE -247 667 (INDIA)**

**JUNE, 2009**

# CANDIDATE'S DECLARATION

---

I hereby declare that the work, which is being presented in the dissertation entitled "A BORDER BASED APPROACH FOR HIDING SENSITIVE FUZZY WEIGHTED ITEMSETS IN QUANTITATIVE DATABASES" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology in Computer Science** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee (India) is an authentic record of my own work carried out during the period from July 2008 to June 2009, under the guidance of **Dr. Durga Toshniwal, Assistant Professor, Department of Electronics and Computer Engineering, IIT Roorkee.**

I have not submitted the matter embodied in this dissertation for the award of any other degree or diploma.

Date:

Place: Roorkee

(MRIDULA VERMA)

---

# CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date:

Place: Roorkee

*Durga Toshniwal.*  
*28/6/09.*  
(Dr. Durga Toshniwal)

Assistant Professor

Department of Electronics and Computer Engineering

IIT Roorkee – 247 667

## ACKNOWLEDGEMENTS

---

I would like to take this opportunity to extend my heartfelt gratitude to my guide and mentor **Dr. Durga Toshniwal**, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for her trust in my work, her able guidance, regular source of encouragement and assistance throughout this dissertation work. I would state that the dissertation work would not have been in the present shape without her inspirational support and I consider myself fortunate to have done my dissertation under her.

I also extend my sincere thanks to **Dr. S. N. Sinha**, Professor and Head of the Department of Electronics and Computer Engineering, for providing facilities for the work.

Finally, I would like to say that I am indebted to my parents for everything that they have done for me. All of this would have been impossible without their constant support.

**MRIDULA VERMA**

## ABSTRACT

---

Association rule mining is an important technique in data mining. Traditional association rule discovery process deals with crisp quantitative data values. However, there are cases when the data values are not well separable into crisp boundaries. This data can be termed as fuzzy data. In such cases, a single value can have membership associated with multiple attributes or groups. Traditional association rule discovery fails to work on such data. Fuzzy association rule mining techniques are used to deal with uncertain or fuzzy data. In many real world applications, all items in the database may not be of equal significance from data mining perspective. So, in such cases weights are assigned to items to reflect their importance. Applying privacy preservation on weighted fuzzy frequent itemsets is an active area of research in data mining.

In context to privacy preservation, fuzzy weighted itemsets can be categorized as sensitive itemsets and non-sensitive itemsets. Sensitive itemsets are those which are critical to the user or application and must remain hidden. Non-sensitive itemsets are those which are less critical and may not remain hidden. Some non-sensitive itemsets have high predicting capability i.e. they may be used to predict sensitive itemsets values. It is important to identify such non-sensitive itemsets and to prevent their misuse. Also, the hiding of sensitive itemsets may affect the sanitized database.

In the thesis, an algorithm has been proposed to extract fuzzy weighted frequent itemsets. The proposed work also identifies the highly predictive fuzzy weighted non-sensitive itemsets and hides them in combination of sensitive itemsets to obtain well maintained sanitized database. To achieve database sanitization, border based approach for hiding fuzzy weighted itemsets has been proposed. The work has been done using quantitative datasets. Case data has been taken from real life applications.

# CONTENTS

---

---

<b>CANDIDATE’S DECLARATION.....</b>	<b>i</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>ii</b>
<b>ABSTRACT.....</b>	<b>iii</b>
<b>TABLE OF CONTENTS.....</b>	<b>iv</b>
<b>LIST OF FIGURES.....</b>	<b>vi</b>
<b>LIST OF TABLES.....</b>	<b>vii</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Introduction.....	1
1.2 Motivation.....	3
1.2 Statement of the Problem.....	4
1.3 Organization of the Dissertation.....	5
<b>CHAPTER 2: BACKGROUND AND LITERATURE REVIEW.....</b>	<b>6</b>
2.1 Privacy Preserving Data Mining.....	6
2.2 Membership Function .....	7
2.3 Fuzzy Association Rules.....	13
2.4 Frequent Pattern Hiding .....	17
2.5 Research Gaps.....	22
<b>CHAPTER 3: PROPOSED WORK.....</b>	<b>24</b>
3.1 Overview.....	24
3.2 Preprocessing.....	25
3.3 Classification and Fuzzification.....	26
3.4 Fuzzy Weighted Itemsets (FWI) Mining.....	26
3.5 Identifying Highly Predictive Non-Sensitive FWIs.....	29
3.6 Border-Based Approach for Hiding Sensitive FWIs.....	32

<b>CHAPTER 4: SYSTEM DESIGN AND IMPLEMENTATION.....</b>	<b>36</b>
4.1 Database Used.....	36
4.2 Code Platform.....	37
4.3 Modules and Procedures.....	38
<b>CHAPTER 5: RESULTS AND ANALYSIS.....</b>	<b>44</b>
5.1 Results .....	44
5.2 Analysis.....	47
<b>CHAPTER 6: CONCLUSION AND FUTURE WORK.....</b>	<b>49</b>
6.1 Conclusion.....	49
6.2 Future Work.....	50
<b>REFERENCES.....</b>	<b>51</b>
<b>LIST OF PUBLICATIONS.....</b>	<b>54</b>
<b>APPENDIX: SOURCE CODE LISTING.....</b>	<b>55</b>

# LIST OF FIGURES

---

<b>Figure Number</b>	<b>Description</b>	<b>Page Number</b>
1.1	Knowledge Discovery Process	2
2.1	Membership function of a fuzzy set	8
2.2	Single input neuron	11
2.3	A Multi-layered Multiple Input Neural Network	13
2.4	Crisp Partition	14
2.5	Fuzzy Boundaries	15
2.6	Example Itemset Lattice	20
3.1	Overview of the Work	24
3.2	Modified Fuzzy C-Mean Clustering	26
3.3	Frequent Fuzzy Itemset Lattice	33
3.4	Border-based Approach for Hiding Sensitive Fuzzy Weighted Itemsets	34
4.1	Multilayer Feed-forward Neural Network	40
5.1	Membership Function obtained for Class 1	44
5.2	Affect of Support on Number of Frequent Weighted Itemsets	45
5.3	Comparison between FFIM and FFWIM	45
5.4	Effect of Support & no. of Sensitive entries on no. of highly predictive non-sensitive itemsets	46
5.5	Effect of Average Support Difference on Quality Factor	47
5.6	Effect of Number of Transactions on Execution Time	47

# LIST OF TABLES

---

<b>Table Number</b>	<b>Description</b>	<b>Page Number</b>
2.1	Different Activation Functions	12
2.2	Membership function of a fuzzy set	18
2.3	Single input neuron	21
3.1	Fuzzified Quantitative Database	30
3.2	Fuzzified Quantitative Temporary Database	30
4.1	Database Details	37



# CHAPTER 1

## INTRODUCTION

---

Privacy preserving data mining – getting valid data mining results without learning the underlying data values has been receiving attention in the research community. As a young research field, data mining has made broad and significant progress since its early beginning in the 1980s [1]. Today data mining is used in a vast array of areas, and numerous commercial data mining systems are available for various types of databases. This chapter gives an introduction and motivation behind the proposed work, discusses the problem statement of the proposed work and the structure of this thesis.

### 1.1 Introduction

The evolution of information technologies and especially the networks like the Internet enabled companies to easily record data from their customers. Since then, huge amounts of data have been collected and stored in the databases of many enterprises. Due to the fact that a lot of business intelligence is hidden in these large databases, the companies need efficient automated tools to find out patterns and regularities.

Data mining (sometimes called knowledge discovery in data) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases. It consists of five major elements [2], as shown in fig. 1.1.

Many data mining tools have been developed that allow a great variety of analysis techniques, mostly derived from classical statistics. Since its introduction, the technique of association rules mining has received great interest by the data mining community and a lot of research has been done resulting in the development of many different algorithms. Association rules are especially useful for conducting market basket analysis,

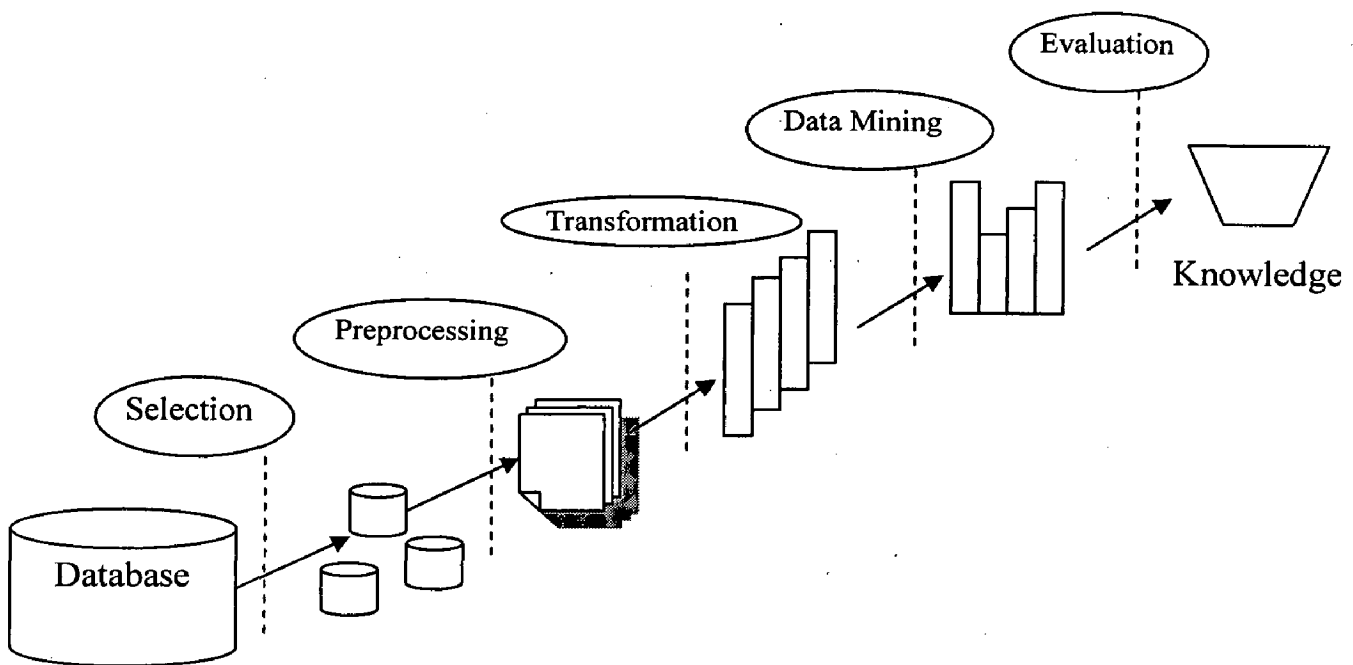


Figure 1.1: Knowledge Discovery Process

where transaction data can be analyzed. Regularities in data of a supermarket for example can be found in this way. An association rule could be “If a customer buys bread and milk, he will mostly buy butter as well”. This information is very useful for business because promotion actions can be designed accordingly.

A problem of classical association rules is that not every kind of data can be used for mining. Rules can only be derived from data containing binary data, where an item either exists in a transaction or it does not exist. When dealing with a quantitative database, no association rules can be discovered. This fact led to the invention of quantitative association rules, where the quantitative attributes are split into intervals and the single elements are either members or nonmembers of those intervals.

Beyond the positive consequences of higher information accuracy, a negative point is a feeling of dwindling privacy for individual person (or company). The objective of data mining is to generalize across population, rather than revealing information about individuals. So, the true problem is not data mining, but the way data mining is done.

Large repositories of data contain sensitive information that must be protected against unauthorized access. Recent advances in data mining and machine learning algorithms have increased the disclosure risks that one may encounter when releasing data to outside parties. Privacy preserving techniques are used to preserve the private information of a user. These techniques, using methods like data hiding, cryptography, permutations, use to affect the basic organization of the database. So, there is a need of techniques to preserve the privacy of user data without affecting the distribution of the database. Mostly these real world databases are of quantitative type. This work proposes an algorithm to hide the sensitive items in the quantitative fuzzy database without affecting the non-sensitive database.

## **1.2 Motivation**

The quantitative approach allows an item either to be member of an interval or not. This leads to an under or overestimation of values that are close to the borders of such “crisp” sets. To overcome this problem, the approach of fuzzy association rules has been developed. It allows the intervals to overlap, making the set fuzzy instead of crisp. Items can then show a partial membership to more than one set, overcoming the above addressed, so-called “sharp boundary problem”. The membership of an item is defined by a membership function and fuzzy set theoretic operations are incorporated to calculate the quality measures of discovered rules. Using this approach, rules can be discovered that might have got lost with the standard quantitative approach.

In previous works [1, 13, 14, 15], whether the database is boolean type or quantitative type there are some sensitive itemsets which should be hidden to make the user privacy preserved. Especially in medical institutions, there are databases including very extensive information about patients. Possible bad purposed usage of those databases threatens personal privacy of patients. Fuzzy data mining concept is usually concerned with medical databases. To estimate the type of the disease in the patient, generally doctors use IF-THEN rules. These IF-THEN rules are generally represented by fuzzy relations. That is why; fuzzy databases are used in medical applications. For example, if a patient is

having fever of  $100^{\circ}\text{C}$  and the size of the radius of red cells in the blood is more than  $5\mu\text{m}$  then it can imply that he has been suffering from Malaria.

There are some recent examples about bad purpose usage of medical information of patient. For example; Kiser, one of the most important medical institutes of United States sent 858 e-mail messages by mistake. Those messages contained IDs of users and their answers for their illnesses and the questions as well. All of those messages were sent to wrong receivers. (Washington Post, 10 August 2000). In another example, Global Healthtrax, an online firm selling health products, sent names, home phones, bank account numbers and credit card data through their website by mistake. (MSNBC, 19 January 2000) [3].

This thesis gives an overview of membership function generation, fuzzy associations mining, and an introduction to hiding sensitive fuzzy weighted itemsets in quantitative databases. A privacy preserving framework has been proposed for fuzzy databases. Additionally, a technique has also been described which is used to generate the membership function for the fuzzy dataset. The following section describes the problem statement of this thesis work.

### **1.3 Statement of the Problem**

The work undertaken in the thesis is given as follows,

- To find frequent fuzzy weighted itemsets in quantitative databases.
- To identify the highly predictable non-sensitive fuzzy weighted itemsets.
- To hide the sensitive fuzzy weighted itemsets and non-sensitive highly predictable fuzzy weighted itemsets while maintaining the quality of database.

The assumptions for the work are:

1. The algorithms are designed for the quantitative databases.
2. During hiding process the support of itemsets are decreased by its associated membership.

## **1.4 Organization of the Thesis**

The report is divided into seven chapters including this chapter that introduces the topic and states the problem. The rest of the thesis report has been organized as follows:

A brief review of literatures studied and the background knowledge for this work has been discussed in Chapter 2. Basic concepts like fuzzy C-Means clustering, neural network, fuzzy itemset mining etc. have been discussed in this chapter. All research gaps found will also be discussed.

The proposed work has been described with the help of block diagram in Chapter 3. Each module of the block diagram has been described in detail.

In Chapter 4, we give the implementation details of this work. It includes a brief description of the database used, code platform and the modules and procedures created in implementation.

All the results and a detailed analysis of these results have been given in Chapter 5. In the end, we conclude the thesis in Chapter 6 with some suggestions for future works.

## CHAPTER 2

# BACKGROUND AND LITERATURE REVIEW

---

Privacy Preserving Data Mining is a research area concerned with the privacy driven from personally identifiable information, when considered for data mining. In particular, in privacy preserving data mining, it is aimed at providing data to public for data mining purpose while not risking personal data and performing high precision of the mining algorithm at the same time [4]. An overview will be given in this chapter. Various privacy preserving data mining systems are available for various types of databases. In real-world scenario generally databases is of quantitative form which can be implemented by using fuzzy concepts. In this chapter, all the literature studied for this work is given.

### 2.1 Privacy Preserving Data Mining

The use of Internet has enabled in the last years an unprecedented level of automated data collection. Parallel to this, data mining has emerged as an important discipline providing powerful tools for data analysis. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified [5]. Beyond the positive consequences of higher information accuracy, a negative point is a feeling of dwindling privacy for individual person (or company).

Privacy preserving data mining is first defined by R. Agrawal and R. Srikant [6]. They addressed the problem of development of accurate model without access to precise information in individual data records. In their work, R. Agrawal and R. Srikant introduced a quantitative measure to evaluate the amount of privacy offered by a method and introduced their reconstruction procedure to reconstructing the original data distribution given a perturbed distribution.

A most useful and generalized classification of different techniques was done by V. S. Verykios et al. [1]. They classified different privacy preserving data mining techniques into five categories. These categories are namely data distribution, data modification, data

mining algorithm, data or rule hiding and the privacy preserving. They also gave a review of Heuristic-based techniques, Cryptography-based techniques and Reconstruction-based techniques of privacy preserving and also provide an evaluation of privacy preserving algorithms.

Data mining can be done by various techniques i.e. the useful knowledge from the database can be extracted by various techniques. According to the dimensions of the data mining techniques, dimensions of techniques for the privacy preservation also expands. Privacy preservation can be applied to clustering, association rule mining, classification and other data mining techniques.

## **2.2 Membership Function**

Fuzzy concept uses membership functions to provide membership to items into a fuzzy set. A membership function puts a lot of impact on the result of the fuzzy computation. So, selection of an accurate membership function is an important task for applying fuzzy concept. The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion.

Member degrees of fuzzy sets include similarity, preference, and uncertainty [7]. Membership functions on  $X$  represent fuzzy subsets of  $X$ . The membership function which represents a fuzzy set is usually denoted by  $\mu_A$ . For an element  $x$  of  $X$ , the value  $\mu_A(x)$  is called the membership degree of  $x$  in the fuzzy set. The membership degree  $\mu_A(x)$  quantifies the grade of membership of the element  $x$  to the fuzzy set. The value 0 means that  $x$  is not a member of the fuzzy set; the value 1 means that  $x$  is fully a member of the fuzzy set. The values between 0 and 1 characterize fuzzy members, which belong to the fuzzy set only partially.

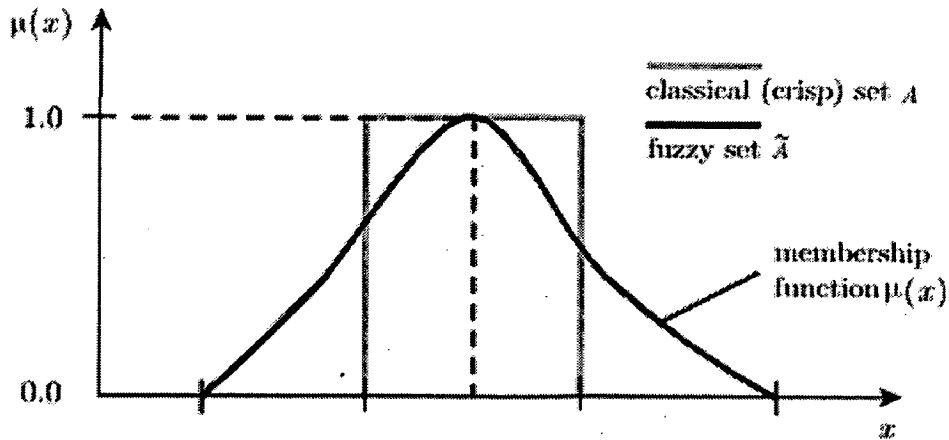


Figure 2.1: Membership function of a fuzzy set

Fig. 2.1 demonstrates a membership function with respect to a crisp boundary. The approach adopted for acquiring the shape of any particular membership function is often dependent on the application. For most fuzzy logic control problems the assumption is that the membership functions are linear - usually triangular in shape [8]. However, for many other applications triangular membership functions are not appropriate.

In [9] all the basic techniques of membership function generation is described. The techniques include heuristic based, feed-forward neural networks, clustering and mixture decomposition etc. The author iterates that there is no single best method and the choice of method depends on the particular problem.

Lucero and Patricia in [10] give a method for membership function generation if the training data is present. If the membership of each data point to each class is defined, then they show the technique to find out the membership function. They also stated that a module which automatically creates membership functions for a system's input parameters with neuro-fuzzy systems will be much more efficient.

The author proposed a fuzzy learning method for automatically deriving membership functions from a set of given training examples. The proposed approach can significantly reduce the time and sort needed to develop a fuzzy expert system. As an example, they explain the technique to find out a triangular membership function.



The technique we have used in this work is finding membership function using clustering to get the class label and using these class labels, the membership values of data items for these classes has been obtained using neural network.

### ***Clustering***

Clustering involves the task of dividing data points into homogeneous classes or clusters so that items in the same class are as similar as possible and items in different classes are as dissimilar as possible. Clustering can also be thought of as a form of data compression, where a large number of samples are converted into a small number of representative prototypes or clusters.

In non-fuzzy or hard clustering, data is divided into crisp clusters, where each data point belongs to exactly one cluster. In fuzzy clustering, the data points can belong to more than one cluster, and associated with each of the points are membership grades which indicate the degree to which the data points belong to the different clusters. This sub-chapter demonstrates the fuzzy c-means clustering algorithm. The clustering used in this work is a bit modified version of FCM.

Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition. It is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^N \sum_{j=1}^C u_{ij}^m \|x_i - c_j\|^2, \quad 1 \leq m < \infty \quad (1)$$

where  $m$  is any real number greater than 1,  $u_{ij}$  is the degree of membership of  $x_i$  in the cluster  $j$ ,  $x_i$  is the  $i^{\text{th}}$  of  $d$ -dimensional measured data,  $c_j$  is the  $d$ -dimension center of the cluster, and  $\|*\|$  is any norm expressing the similarity between any measured data and the center. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above.

This iteration will stop when  $\max_{ij} \left\{ \left| u_{ij}^{(k+1)} - u_{ij}^{(k)} \right| \right\} < \varepsilon$ , where  $\varepsilon$  is a termination criterion between 0 and 1, whereas  $k$  is the iteration steps. This procedure converges to a local minimum or a saddle point of  $J_m$ . The algorithm is composed of the following steps:

Step 1: Initialize  $U = [u_{ij}]$  matrix,  $U^{(0)}$

Step 2: At  $k$ -step: calculate the centers vectors  $C^{(k)} = [c_j]$  with  $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^N u_{ij}^m \cdot x_i}{\sum_{i=1}^N u_{ij}^m} \quad (2)$$

Step 3: Update  $U^{(k)}, U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^c \left( \frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \quad (3)$$

Step 4: If  $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$  then STOP; otherwise return to step 2.

### **Artificial Neural Network:**

The basic concept of artificial neural network comes from the Biological neural network which works in human brain. A biological neuron receives electrochemical signals from many sources (other neurons) and when the excitation in the neuron is high enough, it starts fire and passes the signal to the next neuron.

An artificial neuron is defined as follows [11]:

- It receives a number of inputs (either from original data, or from the output of other neurons). Each input comes via a connection that has a strength (or *weight*); these weights correspond to synaptic efficacy in a biological neuron. Each neuron also has a single threshold value. The weighted sum of the inputs is formed, and the threshold subtracted, to compose the *activation* of the neuron.
- The activation signal is passed through an activation function (also known as a transfer function) to produce the output of the neuron.

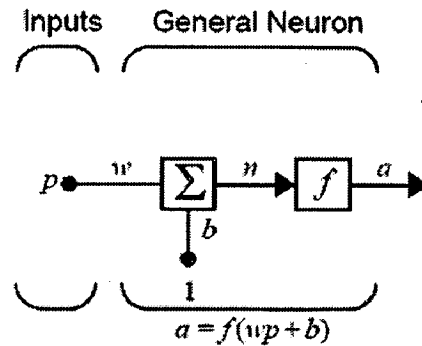


Figure 2.2: Single input neuron

Coming to the very basic concept, a single neuron model is shown in fig. 2.2. The scalar input  $p$  is multiplied by a scalar weight  $w$  to form  $wp$ , which is sent to the summing function. The other input called bias ( $b$ ) is passed to summing function. The summing function gives output  $n$  which goes to the transfer function (or activation function)  $f$ , which produces the neuron output  $a$ .

In this case, the neuron output is calculated as,

$$a = f(wp + b) \tag{4}$$

The actual output depends on the particular transfer function that is chosen. Table 2.1 shows the various types of transfer function which are generally used in artificial neural network.

Typically a neural network has multiple input  $p_1, p_2, \dots, p_n$ . In these type of networks the weights are shown by a matrix of size  $l * n$ . Thus the weights will be  $w_{l, 1}, w_{l, 2} \dots w_{l, n}$ . Here the output will be,  $a = f(Wp + b)$ , where  $W$  is a one dimensional matrix of weights. Coming to more complex type of networks, the following fig is showing a three layer multiple input neural network.

As shown in the fig. 2.3, there are  $R$  inputs,  $L_1$  neurons in the first layer,  $L_2$  neurons in the second layer, etc. We can have different number of neurons on different layers. The outputs of one and two will be the inputs of layer two and three. Thus, layer 2 can be viewed as one-layer network with  $R = L_1$  inputs,  $L = L_2$  neurons and an  $L_1 * L_2$  weight matrix  $W_2$ . The input to the layer 2 is  $a^1$  and output is  $a^2$ . A layer whose output is the n/w





Name	Input/Output Relation	Icon
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$	
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$	
Linear	$a = n$	
Log-Sigmoid	$a = 1/(1+e^{-n})$	

Table 2.1: Different Activation Functions

output is called an output layer. All the other layers are called hidden layers. In the above network there are two hidden layers.

After this brief introduction to neural network the main concept to discuss is the learning algorithm. After the design of the neural network, we have to train it so that it can classify our input data objects correctly. In this step, we provide a set of training data to the network and using a proper learning algorithm, network use to learn. There are various algorithms provided for learning. Here the best known example is being described i.e. Back-Propagation algorithm.

In *back propagation*, the gradient vector of the error surface is calculated. This vector points along the line of steepest descent from the current point, so we know that if we move along it a "short" distance, we will decrease the error. A sequence of such moves (slowing as we near the bottom) will eventually find a minimum of some sort. The difficult part is to decide how large the steps should be.

The typical back-propagation network has an input layer, an output layer, and at least one hidden layer. Each layer is fully connected to the succeeding layer. The training process normally uses some variant of the Delta Rule, which starts with the calculated

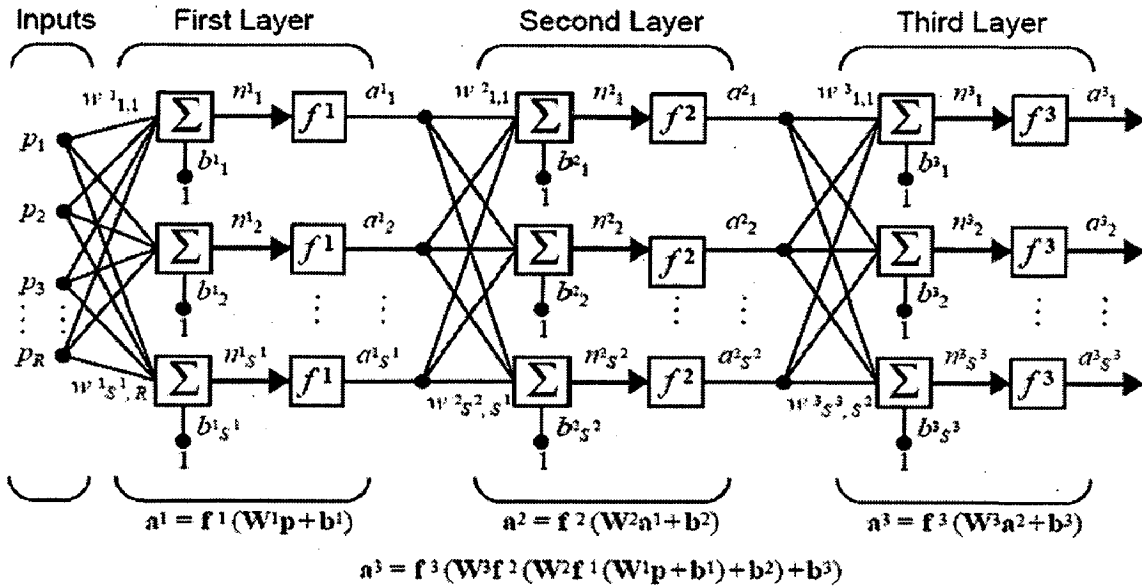


Figure 2.3: A Multi-layered Multiple Input Neural Network

difference between the actual outputs and the desired outputs. Using this error, connection weights are increased in proportion to the error times a scaling factor for global accuracy. Doing this for an individual node means that the inputs, the output, and the desired output all have to be present at the same processing element. The complex part of this learning mechanism is for the system to determine which input contributed the most to an incorrect output and how does that element get changed to correct the error. To solve this problem, training inputs are applied to the input layer of the network, and desired outputs are compared at the output layer. During the learning process, a forward sweep is made through the network, and the output of each element is computed layer by layer. The difference between the output of the final layer and the desired output is back-propagated to the previous layer(s), usually modified by the derivative of the transfer function, and the connection weights are normally adjusted using the Delta Rule. This process proceeds for the previous layer(s) until the input layer is reached.

### 2.3 Fuzzy Association Rules

Among various data mining techniques, association rule mining is the most popular one. In this technique, we use to find out interesting associations and correlations among itemsets in the database. Various works have been done in the field of association rule mining. Most studies have shown how binary valued transactions can be handled.

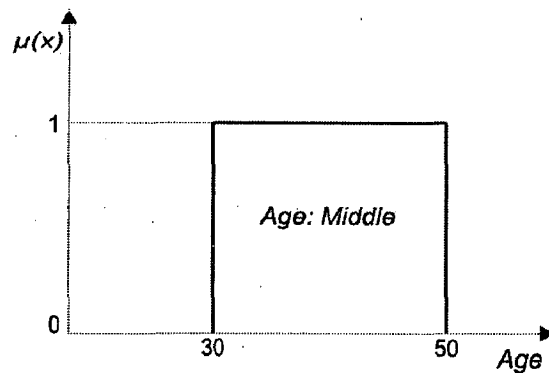


Figure 2.4: Crisp Partition

However, transaction in the real-world applications usually consists of quantitative values, so designing sophisticated data-mining algorithms able to deal with various types of data presents a challenge to workers in this research field.

Based on classical association rule mining, a new approach has been developed expanding it by using fuzzy sets. The new fuzzy association rule mining approach emerged out of the necessity to mine quantitative data frequently present in databases efficiently. When dividing an attribute in the data into sets covering certain ranges of values, we are confronted with the sharp boundary problem.

Elements near the boundaries of a crisp set will either be ignored or overemphasized. For example, one can consider a set representing persons of middle age, ranging from 30 to 50 years old (see Fig. 2.4). In this example, a person aged 29 years would be a 0% representative and a 31 year old would be 100%. In reality, the difference between those ages is not that great. Implementing fuzziness can overcome this problem.

The same problem can occur if one is dealing with categorical data. Sometimes, it is not ultimately possible to assign an item to a category. As an example, one can say that a tomato is a vegetable but also, in a way, a fruit. Crisp sets would only allow assigning the item to one single category; fuzzy sets allow different grades of membership to more than one set.

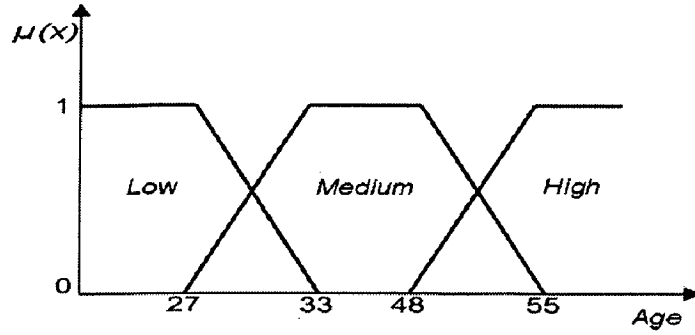


Figure 2.5: Fuzzy Boundaries

For example, in the previous case we might want to partition the variable *Age* into three fuzzy sets. The fuzzy sets and their membership functions will have to be defined by a domain expert. For easy demonstration, the borders of the sets are defined and split the overlapping part equally between the so generated fuzzy sets. The following borders for the fuzzy sets of the variable age:  $Age.Low = \{0-33\}$ ,  $Age.Medium = \{27-55\}$ ,  $Age.High = \{48-\infty\}$  can be used. The generated fuzzy sets are shown in fig. 2.5. For all areas having no overlap of the sets, the support will simply be 1 for the actual itemset. If there is an overlap, the membership can be computed by using the borders of the overlapping fuzzy sets. The added support will here always sum up to 1.

Kuok et al. describe fuzzy association rules as follows [12]: “Mining fuzzy association rule is the discovery of association rules using fuzzy set concepts such that the quantitative attribute can be handled”. As in classical association rules,  $I = \{i_1, i_2, \dots, i_m\}$  represents all the attributes appearing in the transaction database  $T = \{t_1, t_2, \dots, t_n\}$ .  $I$  contains all the possible items of a database, different combinations of those items are called itemsets. Each attribute  $i_k$  will associate with several fuzzy sets. In order to represent the fuzzy sets associated with  $i_k$ , the following notion is used,

$$F_{ik} = \{f_{ik}^1, f_{ik}^2, \dots, f_{ik}^m\} \quad (5)$$

where  $f_{ik}^j$  is the  $j$ th fuzzy set in  $F_{ik}$ . As an example, the attribute salary could look as follows:  $F_{Age} = \{high, medium, low\}$ . Fuzzy sets and their corresponding membership functions have to be defined by domain experts. Each of the fuzzy sets can be viewed as a  $[0, 1]$  valued attribute, called fuzzy attribute.

A fuzzy association rule has the following form:

$$\text{If } X \text{ is } A \text{ then } Y \text{ is } B \quad (6)$$

In this case,  $X = \{x_1, x_2, \dots, x_p\}$  and  $Y = \{y_1, y_2, \dots, y_q\}$  are itemsets which are subsets of  $I$ . It is important to notice that those two sets must be disjoint and thus do not have any attributes in common. Fuzzy values,

$$A = \{f_x^1, f_x^2, \dots, f_x^p\} \text{ and } B = \{f_y^1, f_y^2, \dots, f_y^q\} \quad (7)$$

contain the fuzzy sets that are associated with  $X$  and  $Y$ . Known from classical association rules,  $X \text{ is } A$  is the antecedent,  $Y \text{ is } B$  is the consequent. If a sufficient amount of records approves this rule, it will call as satisfied.

In order to enable the evaluation of a fuzzy association rule, we use the standard approach for calculating support and confidence, replacing the set-theoretic operations by the corresponding fuzzy set-theoretic operations [13]:

$$\text{supp}(A \rightarrow B) = \sum_{x \in D} (T(A(x), B(x))) \quad (8)$$

$$\text{conf}(A \rightarrow B) = \sum_{x \in D} (T(A(x), B(y))) / \sum_{(x, y) \in D} A(x) \quad (9)$$

Additionally, if  $A$  supports  $B$ ,  $B$  will automatically also support  $A$ . This is due to the fact that the support is computed by simply summing up the memberships of the different items in the database. Thus:

$$\text{supp}_{[x, y]}(A \rightarrow B) = \text{supp}_{[x, y]}(B \rightarrow A) \quad (10)$$

Work done in [14] describes a technique of mining the quantitative data in large relational tables. It defines the traditional association rule as a ‘‘Boolean Association Rule’’, and introduces a new term as ‘‘Quantitative Association Rule’’. The basic approach which is described in this work is mapping of quantitative association rule problem into boolean association rule problem. It first finds out the partitions in the quantitative data and then maps the data values into these partitions. Thus all the values in the database get partitioned through crisp boundaries.



The main problem with this approach of partitioning is the information loss due to crisp boundaries, is the sharp boundary problem defined by Kuok in [12]. They describe that a fuzzy concept is better than the partitioning method, since fuzzy sets provides a smooth transition between members and non-members of a set. Kuok uses two factors as significance and certainty to find out the large itemsets and rule interestingness respectively.

Hong et al. proposed a fuzzy mining algorithm to mine fuzzy rules from quantitative transaction data [15]. Basically, the fuzzy mining algorithms first used membership functions to transform each quantitative value into a fuzzy set in linguistic terms. The algorithm then calculated the scalar cardinality of each linguistic term on all the transaction data. The mining process based on fuzzy counts was then performed to find fuzzy association rules.

One more efficient algorithm is given in [16], by Hong, Kuo and Wang. It uses an AprioriTid mining algorithm with comparatively reduced computational time. They took reference from the work done by R. Aggrawal and R Srikant, which describes two fast algorithm of association rule mining. The AprioriTID is a fast algorithm since it uses only one scanning of the database from memory. So the memory I/O time get decreases.

## **2.4 Frequent Itemset Hiding**

Data hiding is a popular technique to preserve the privacy of the user. In this technique sensitive data is hided from the attackers. There is a specific class of methods in the knowledge hiding area, known as frequent itemset and association rule hiding. Other classes of methods, under the same area, include classification rule hiding, clustering model hiding, sequence hiding and so on and so forth. "Association rule hiding" has been mentioned for the first time in 1999 in a workshop paper by Atallah et al. [13].

According to [14], association rule hiding algorithms can be divided into three distinct classes, i.e. heuristic approaches, that involves efficient, fast algorithms that selectively sanitize a set of transactions from the database to hide the sensitive knowledge, border-

based approaches, that considers the task of sensitive rule hiding through modification of the original borders in the lattice of the frequent and the infrequent patterns in the dataset and exact approaches that contains non-heuristic algorithms which conceive the hiding process as a constraint satisfaction problem that they solve by using integer or linear programming.

The concept of border is used in [15] by Sun and Yu. They used this concept to hide the sensitive itemsets so that the non sensitive itemsets can be minimally affected by the hiding process. By iterative revising the borders and calculating the affects on non-sensitive items this algorithm succeed in maintaining the database free from any side effect by sensitive frequent itemset hiding process.

In [16], two new algorithms which rely on the maxmin criterion for the hiding of sensitive itemsets in an association rule hiding framework. Both algorithms apply the idea of the maxmin criterion in order to minimize the impact of the hiding process to the revised positive border which is produced by removing the sensitive itemsets and their super itemsets from the lattice of frequent itemsets. This approach relies on the maxmin criterion which is a method in decision theory for maximizing the minimum gain.

Finally, there is one more work on frequent itemset hiding based on borders. This technique uses integer programming approach [17] of operation research to solve the problem of effect of hiding process on the database. In the first step this technique introduces the concept of distance between two databases (taken as original and the sanitized one) and a measure to quantify it. After that using integer programming approach it tries to minimize the distance between the two databases after each step of itemset hiding.

### ***Border Theory***

This concept will be used to apply a basic border between the sensitive and non-sensitive frequent itemsets. The key idea is that the border of non-sensitive frequent itemsets is used to track the impact on the result database during the hiding process, and maintain the

quality of the result database by selecting the modification with minimal impact at each step.

**Itemset lattices:** An itemset lattice contains all of the possible itemsets for a transaction database. Each itemset in the lattice points to all of its supersets. When represented graphically, an itemset lattice can help to understand the concepts behind the borders. The concept of *border* is initially introduced in [21] and it is well applied in the research of maintaining the frequent itemsets. For the completeness of the report, a brief review of the concept of border is given here.

Consider a set of itemsets  $U$ , the upper border of  $U$  denoted as  $Bd^+(U)$ , will be a subset of  $U$  with the following properties:

- 1)  $Bd^+(U)$  is an antichain collection of sets.
- 2)  $\forall X \in U$ , there exist at least one itemset  $Y \in Bd^+(U)$  holding  $X \subseteq Y$ .

In mathematics, in the area of order theory, an **antichain** is a subset of a partially ordered set such that any two elements in the subset are incomparable. Let  $S$  be a partially ordered set. We say two elements  $a$  and  $b$  of a partially ordered set are **comparable** if  $a \leq b$  or  $b \leq a$ . If two elements are not comparable, we say they are **incomparable**; that is,  $x$  and  $y$  are incomparable if neither  $x \leq y$  nor  $y \leq x$ .

A **chain** in  $S$  is a subset  $C$  of  $S$  in which each pair of elements is comparable; that is,  $C$  is totally ordered. An **antichain** in  $S$  is a subset  $A$  of  $S$  in which each pair of different elements is incomparable; that is, there is no order relation between any two different elements in  $A$ .

Similarly, the negative border of  $U$  is denoted by  $Bd^-(U)$ , has the properties as,

- 1)  $Bd^-(U)$  is an antichain collection of sets.
- 2)  $\forall X \in U$ , there exists at least one itemset  $Y \in Bd^-(U)$  holding  $Y \subseteq X$ .

An itemset in the upper border or lower border is called a border element. For fig. 2.8 the positive and the negative borders will be given as,

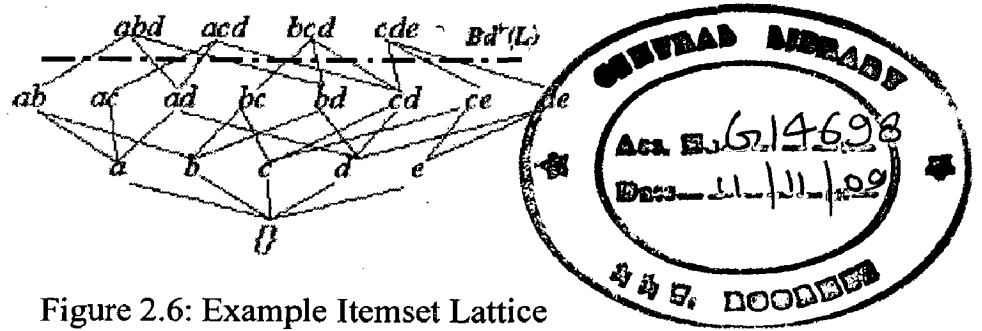


Figure 2.6: Example Itemset Lattice

$$Bd^+(L) = \{abd, acd, bcd, cde\}$$

$$Bd(L) = \{a, b, c, d, e\}$$

### ***Border-based Approach of Hiding Frequent Itemset***

Consider  $D$  is our original database and  $D'$  is the sanitized version of the database i.e. the database in which all the sensitive itemsets are not  $\sigma$ -frequent, where  $\sigma$  is the minimum support of an itemset. Also, let  $L$  is the set of all  $\sigma$ -frequent itemsets,  $\Delta L$  is the set of all frequent itemsets which have to be hidden,  $L_r$  is the set of all non-sensitive frequent itemsets and  $L'$  is the set of  $\sigma$ -frequent itemsets in  $D'$ .

The following are the considerations:

1. Any  $\sigma$ -frequent itemset does not belong to  $\Delta L$ .
2. The factor  $|L_r - L'|$  must be minimized.

According to the Apriority property, concentrating on the border  $Bd^+(L_r)$  during the hiding process is effective in avoiding the over-hiding non-sensitive frequent itemset. Let us consider  $\Lambda(X)$  be the set of transactions that contain frequent itemset  $X$ . A set  $C$  of **hiding candidates** of itemset  $X$  is defined as,

$$C = \{(T, x) \mid T \in \Lambda(X) \wedge x \in X\}. \tag{11}$$

Once a hiding candidate  $(T_0, x_0)$  is deleted, i.e.,  $x_0$  is deleted from transaction  $T_0$ , the new set of  $C'$  hiding candidate is  $C - \{(T, x) \mid T = T_0\}$ . Each border element  $B$  in  $Bd^+$  is assigned a weight, showing its vulnerability of being affected by item deletion. The

weight of  $B$  is dynamically computed based on its current support during the hiding process. Whenever  $Supp(X)$  of a sensitive frequent itemset  $X$  is reduced, for each hiding candidate  $c$ , its impact on the border as the sum of weights of the border elements that will be affected by deleting  $c$  will be calculated. Each time the candidate item with a minimal impact on the border  $B_d^+$  is deleted until  $Supp(X)$  drops to  $\sigma - 1$ .

Weights are defined as shown below. Let  $D''$  be the database during the process of transformation and  $Supp''(B)$  be the support of  $B$  in  $D''$ . The **weight** of border element  $B$  is defined as:

$$\begin{aligned} w(B) &= (Supp(B) - Supp''(B) + 1) / Supp(B) - \sigma && \text{if } Supp''(B) \geq \sigma + 1 \\ &= \lambda + \sigma - Supp''(B) && \text{if } 0 \leq Supp''(B) \leq \sigma \end{aligned} \quad (12)$$

The larger the weight of a border element  $B$  is having, the more vulnerable  $B$  is to further change, therefore, the lower priority of having  $B$  affected. For a border element  $B$ , when the current support of  $B$ ,  $Supp''(B)$ , is greater than the threshold  $\sigma$ ,  $w(B)$  is no more than 1. When  $Supp''(B)$  equals to  $\sigma$ ,  $w(B)$  is assigned a large integer  $\lambda$ , where  $\infty > \lambda > |Bd^+|$ . The intuition behind this is: if the border element  $B$  is about to be infrequent, a large value is assigned to  $w(B)$ , indicating low priority of being affected. If  $B$  is already overridden ( $Supp''(B) < \sigma$ ),  $B$  should also be avoided for further change. In that case,  $w(B)$  is decided by  $\lambda$  and the amount of  $Supp''(B)$  less than  $\sigma$ . Also, if  $Supp''(B) > \sigma + 1$ , with the decrease of  $Supp''(B)$ ,  $w(B)$  increases under the rate of  $1 / (Supp(B) - \sigma)$ .

There is a term as affected border of  $X$  which is denoted as  $Bd^+|_X$ , and defined as the set of border elements of  $Bd^+$ , which may potentially be affected by hiding  $X$ . Formally,

$$Bd^+|_X = \{B_i \mid B_i \in Bd^+ \wedge B_i \cap X = \emptyset\} \quad (13)$$

Clearly, for evaluating the impact of hiding  $X$  on  $Bd^+$ , only  $Bd^+|_X$  needs to be considered. For a hiding candidate  $u$  of sensitive frequent itemset  $X$ , a set  $S_u$  of border elements that will be affected by deleting  $u$  (note that  $S_u$  is a subset of  $Bd^+|_X$ ) will be determined. The impact of deleting  $u$  on the border should be the sum of the weights of border elements in

$S_u$ . Formally, let  $Bd^+|_X$  be  $\{B_1, \dots, B_n\}$  and a lexicographical order can be imposed among  $B_1, \dots, B_n$ . Given a hiding candidate  $u$  of sensitive frequent itemset  $X$ , we have a **relevance bit vector**  $b_1b_2\dots b_n$  such that  $b_i = 1$  if  $u$  is a hiding candidate of  $B_i$  (i.e., deleting  $u$  will decrease  $Supp(B_i)$ ), otherwise  $b_i = 0$ . The relevance bit vector of  $u$  shows which border element  $B_i$  will be affected if deleting  $u$ . In the running example, for sensitive itemset  $abd$ ,  $Bd^+|_{abd} = \{ab, bd, acd, cde\}$ . The relevance bit vector of hiding candidate  $(T1, a)$  and  $(T3, b)$  are 1010 and 1100 respectively.

Finally, the impact function is defined, which calculates the impact of deleting a hiding candidate on the elements of  $Bd^+|_X$ . It is denoted as  $I(u)$ , and defined as:

$$I(u) = \sum b_i * w(B_i) \quad (14)$$

The value of  $I(u)$  is the sum of the weights of border elements that will be affected by deleting  $u$ .

In the border-based approach basically the hiding candidate is to be found out which put minimal impact on the non sensitive itemsets after its removal. Every time a hiding candidate itemset is selected, the hiding candidate set is to be updated along with the weights of the border elements. Also, after selecting the hiding candidate the database have to be updated.

## 2.5 Research Gaps

After detailed studies of various techniques research papers, the following research gaps are found:

1. Most of the privacy preserving algorithms and techniques has been suggested by researchers are based on boolean databases. Working on 0s and 1s is relatively simple and straightforward. The number of algorithms that are discussed for privacy preserving techniques on quantitative or the numerical database is relatively less.
2. There are cases when a user wants to add some weights to items to prioritize the items. In fuzzy association rule mining field, it is found that there are very few algorithms

that find out weighted fuzzy itemsets.

3. Some approach should be proposed to find out highly predictive fuzzy non-sensitive itemsets from quantitative database which can improve the performance of privacy preservation process.
4. In the field of fuzzy data mining the research work done on privacy preservation is considerably less. In fuzzy association rule hiding, there is one research work by M. Kaya and T. Berberoglu [3] which hides sensitive fuzzy association rules using decreasing its confidence value. Further work needed to be done in this field, which minimizes the effect of data hiding on the database.

# CHAPTER 3

## PROPOSED WORK

---

In this chapter, the proposed algorithm has been discussed. The overall scheme of the work has been shown first. Each module has been described in detail. Here after we will use MF for membership function, ANN for artificial neural network and FWI for fuzzy weighted itemsets.

### 3.1 Overview

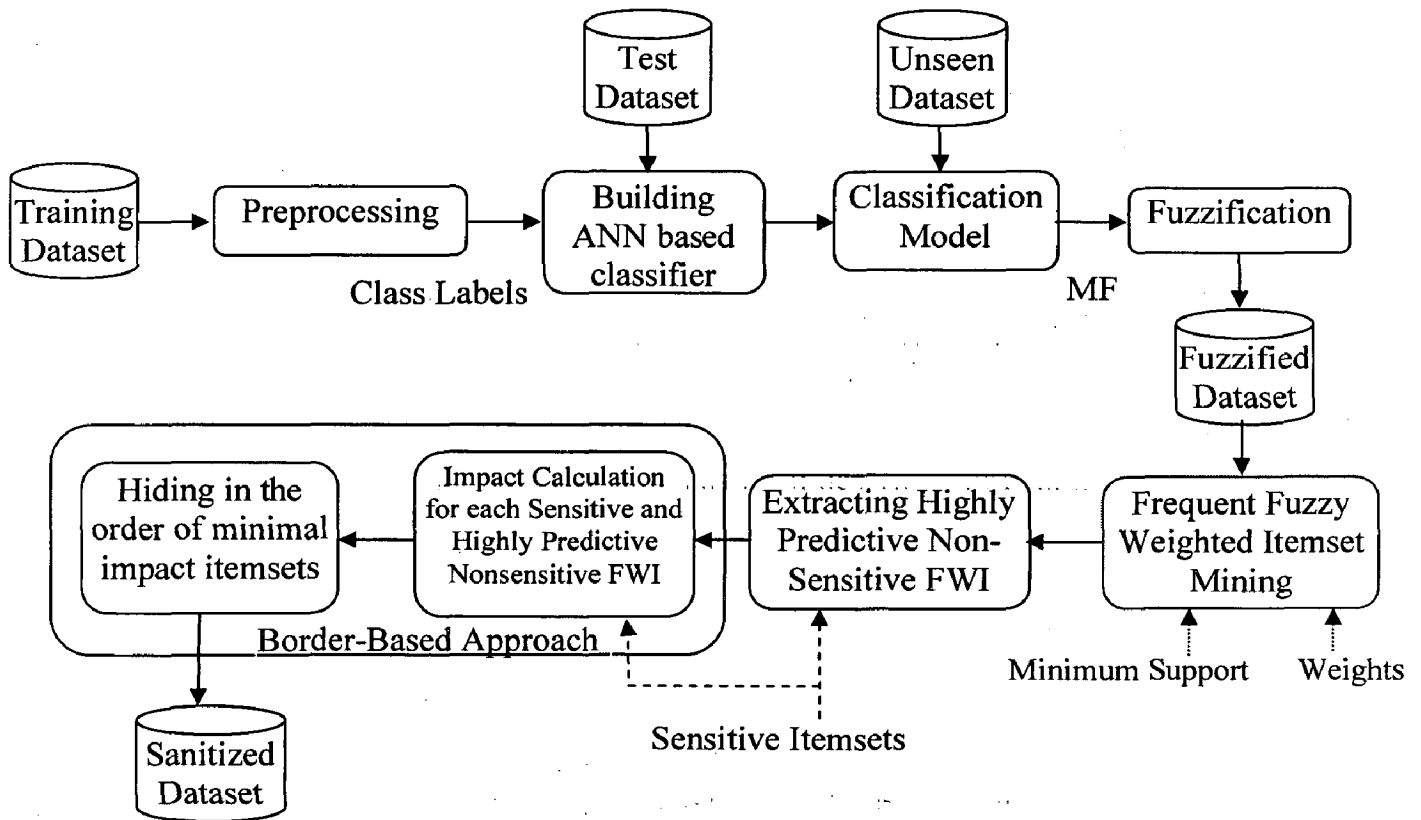


Figure 3.1: Overview of the Work

The above figure shows an overview of the whole work. In the preprocessing step, we have used a modified version of fuzzy c-mean clustering, which is used to obtain the class labels. Using these class labels, and the test set an ANN classifier has been built. This classifier is used to classify the unseen dataset into number of classes. It gives the



membership values of different data points into these classes. Mapping these membership values onto the used activation function, we will obtain the membership function. This membership function has been used in fuzzification process which will provide us the membership values of different attributes in the fuzzy classes. Thus, a fuzzified database will be obtained.

A frequent fuzzy weighted itemset mining process has been proposed, which gives frequent fuzzy weighted itemsets of the fuzzified database. After that, using sensitive fuzzy weighted itemsets, we find out highly predictive non-sensitive fuzzy weighted itemsets. Applying border-based approach on these itemsets, we will obtain a well maintained sanitized database, which does not contain any sensitive fuzzy weighted itemsets and highly predictive non-sensitive fuzzy weighted itemsets. All these modules will be described in detail in the following sub-chapters.

### 3.2 Preprocessing

Conceptually, an artificial neural network is used to obtain the membership values. But since in real life applications the initial knowledge of classes is not defined so we used clustering as a preprocessing step in this work. The basic concept and algorithm of fuzzy c-mean clustering is discussed in previous chapter. In the preprocessing module, a modified fuzzy c-means clustering is being used. The algorithm is composed of the following steps:

Step 1: Initialize  $U = [u_{ij}]$  matrix,  $U^{(0)}$

Step 2: At k-step: calculate the centers vectors  $C^{(k)} = [c_j]$  with  $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^N u_{ij}^m \cdot x_i}{\sum_{i=1}^N u_{ij}^m}$$

Step 3: Update  $U^{(k)}, U^{(k+1)}$

$$u_{ij}^k = \begin{cases} 1 & \text{for max } \frac{1}{\sum_{k=1}^c \left( \frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \\ 0 & \text{otherwise} \end{cases}$$

Step 4: If  $\|U^{(k+1)} - U^{(k)}\| < \epsilon$  then STOP; otherwise return to step 2.

Figure 3.2: Modified Fuzzy C-Mean Clustering

In Step 3, the matrix entry will get a 1 for the minimum distance of a data item from the cluster center. Since in this work these clusters have been used for classification this perfect classification will give better results.

### 3.3 Classification and Fuzzification

After get trained and tested from the training dataset and test dataset the neural network is ready to get the membership values of various data points on the number of classes. The sum of the membership of the unseen data points on these classes will be equal to 1. As discussed in the previous chapter, a sigmoid activation function is used to get the membership in the fuzzy sets. When this membership function is drawn, there will be different curve for each class. These curves will be in form of sigmoid function (as shown in results). Using this graph, the membership association of various attributes is found out. This process is called the fuzzification process. These membership associations will result in the fuzzified database.

### 3.4 FWI Mining

By applying weights, a user can add some importance to the items. The application of weights on itemsets can include the process of privacy preservation in which the user can apply comparably less weights to his sensitive itemsets. Another application can be described by an example. In a grocery shop the shopkeeper is used to mine the

transactional database to find the interesting associations among items so that he can use this knowledge for product placement, increasing sales and promotions etc. For profit purpose he can require the information about the items which are associated with costly items. He can do this by applying more weightage to all the costly items. In the proposed work we have been taken the case of applying more weight to items.

The proposed weighted fuzzy mining algorithm first transforms each quantitative value into a fuzzy set with linguistic terms using membership functions. The algorithm then calculates the scalar cardinality of each linguistic term on all the transaction data with weights. In this section, the weighing concept is used in the Fuzzy Apriory data-mining algorithm to discover priority based fuzzy association rules from quantitative values.

Consider  $n$  be the total number of transaction data and  $m$  be the number of attributes.  $\alpha$  is the predefined minimum support and  $\lambda$  is the predefined minimum confidence value and  $g$  is the number of fuzzy region. The set of candidate itemsets with  $r$  attributes is denoted as  $C_r$ , the set of large itemsets with  $r$  attributes is denoted as  $L_r$  and  $W_{m \times g}$  is the weights provided by the user. The proposed weighted fuzzy mining algorithm first transforms each quantitative value into a fuzzy set with linguistic terms using membership functions. The algorithm then calculates the scalar cardinality of each linguistic term on all the transaction data. The main concept here is the calculation of the support count. In general support is just the addition of the membership values of the attributes. In weighted fuzzy mining algorithm the weights provided by the user is used to calculate the support count.

***The Weighted Fuzzy Frequent Itemset Mining Algorithm:***

INPUT: A set of  $n$  transaction data, each with  $m$  attribute values, a set of membership functions, a predefined minimum support value, a predefined confidence value, and a 2-dimensional matrix of weights.

OUTPUT: A set of fuzzy association rules.

STEP 1: For each transaction data  $D^{(i)}$ ,  $i = 1$  to  $n$ , and for each attribute  $A_j$ ,  $j = 1$  to  $m$ , transfer the quantitative value  $v_j^{(i)}$  into a fuzzy set  $f_j^{(i)}$  using the given membership functions.

STEP 2: For each attribute region  $R_{jk}$ , calculate its scalar cardinality on the transactions:

$$count_{jk} = (\sum f_{jk}^i) * W_{jk} \quad (15)$$

where  $f_{jk}^i$  is the membership value of  $v_j^{(i)}$  in Region  $R_{jk}$ .

STEP 3: For each  $R_{jk}$ ,  $1 \leq j \leq m$  and  $1 \leq k \leq |A_j|$ , check whether its  $count_{jk}$  is larger than or equal to the predefined minimum support value  $\alpha$ . If  $R_{jk}$  satisfies the above condition, put it in the set of large 1-itemsets ( $L_r$ ).

STEP 4: Set  $r = 1$ , where  $r$  is used to represent the number of items kept in the current large itemsets.

STEP 5: Generate the candidate set  $C_{r+1}$  from  $L_r$  in a way similar to that in the apriori algorithm except that two regions belonging to the same attribute cannot simultaneously exist in an itemset in  $C_{r+1}$ . Restated, the algorithm first joins  $L_r$  and  $L_r$  under the condition that  $r-1$  items in the two itemsets are the same and the other one is different. It then keeps in  $C_{r+1}$  the itemsets which have all their sub-itemsets of  $r$  items existing in  $L_r$ , and do not have two items  $R_{jp}$  and  $R_{jq}$  where  $p \neq q$ .

STEP 6: For each newly formed  $(r+1)$ -itemset  $s$  with items  $(s_1, s_2, \dots, s_{r+1})$  in  $C_{r+1}$  do the following sub-steps:

a) For each transaction data  $D^{(i)}$ , calculate its fuzzy value on  $s$  as,  $f_s^{(i)} = f_{s_1}^{(i)} \cap f_{s_2}^{(i)} \dots \cap f_{s_{r+1}}^{(i)}$  where  $f_{s_j}^{(i)}$  is the membership value of  $D^{(i)}$  in region  $s_j$ . If the minimum operator is used for the intersection, then  $f_s^{(i)}$  will contain the minimum value from the row multiplied by the respective weights for each  $f_s^{(i)}$ .

b) Calculate the scalar cardinality of  $s$  on the transactions as:

$$count_s = \sum f_s^{(i)} * Max(W_{s_j}) \quad \text{for } i = 1 \text{ to } n \text{ and } j = 1 \text{ to } r+1 \quad (16)$$

c) If  $count_s$  is larger than or equal to the predefined minimum support value  $\alpha$ , put  $s$  in  $L_{r+1}$ .

STEP 7: If  $L_{r+1} = \text{null}$ , then go to the next step; otherwise, Output all the frequent itemsets in  $L_{r+1}$ , set  $r = r + 1$  and repeat STEPS 5-7.

STEP 8: End

We have taken the maximum value of the weight among all the weights of items in an itemset to show the maximum impact of weights.

### 3.5 Identifying Highly Predictive Non-Sensitive FWIs

The inference of sensitive fields with the use of correlations is undesirable from a privacy preservation perspective. Therefore, in order to prevent such inference, it may be desirable to also hide some of the non-sensitive entries. The corresponding tradeoff here is that unnecessary hiding of entries loses information for the purpose of data analysis applications. Therefore, it is important to hide a *minimal* set of entries (i.e. a set of minimum size) in order to prevent such privacy violations. We use the term Inauspicious for these types of itemsets. Inauspicious or Adversarial itemsets are those itemsets, which are itself non-sensitive, but having a strong predictive power so that the values of the hidden sensitive itemsets can be extracted. The basic Framework presented in [22] is extended for the fuzzy databases.

In many real world scenarios, there are some entries in the databases which user want to be hidden. By removing the entry value from table we can say that the private information of user is preserved from data mining results. But it may happen that there are some other frequent itemsets that have a strong predictive power to predict the sensitive value. In this case there is no sense to remove the sensitive entry from its place. Thus, there is a need to extract these itemsets and an effective data hiding technique is also applied on these non-sensitive itemsets in combination of sensitive itemsets.

In this propose work, different constraints are defined for fuzzy itemsets to be inauspicious. And then along with the sensitive itemsets these non-sensitive itemsets are combined and border-based approach for frequent fuzzy itemset hiding is applied. In this sub-chapter the constraints for checking a frequent fuzzy itemset to be inauspicious will be given.

In classical boolean databases, the rules are of the form,  $A \rightarrow x$ , where  $A$  is an attribute and  $x$  is an entry. In fuzzy databases, rules are of the form,  $A \rightarrow A_0$ . In boolean databases, if an entry  $x$  is sensitive in the database then it reflect to only one place of the database i.e.  $(T_x, A)$  where  $T_x$  is the tuple containing  $x$  and  $A$  is the attribute type of  $x$ . Thus the private dataset given by user will be the set of these types of entries.

However, in fuzzy database, one entry  $x$  of the quantitative database reflects  $R$  places on the tuple of  $x$ , where  $R$  is the fuzzy regions. The reason behind this is, in fuzzy database one entry is shown by  $R$  region. All  $R$  regions contain some membership of  $x$ . So, to remove the entry from database, we have to remove all the entries of  $R$  regions corresponding to the tuple  $T_x$ .

One scheme to show the sensitive quantitative entry in the fuzzified database is to remove all the entries of  $R$  regions corresponding to the tuple  $T_x$ . But this technique can result in over-hiding of fuzzy items, since the membership of an entry is not same in all regions. So, the proposed solution is to take the region with highest count among all the regions of the attribute.

To understanding the concept, let the sensitive entries are replaced by a fake character '#' temporarily. Corresponding to the constraints defined for the boolean items, the following constraints are defined for the fuzzy items. The fuzzy tables before and after introducing the fake character '#' (bold items are the terms containing sensitive values),

	A0	A1	A2	B0	B1	B2	C0	C1	C2	D0	D1	D2
<b>T1</b>	0.11	0.33	0.66	<b>0.77</b>	<b>0.11</b>	<b>0.22</b>	0.33	0.33	0.44	0.11	0.55	0.44
<b>T2</b>	0.33	0.44	0.33	0.66	0.11	0.33	<b>0.44</b>	<b>0.66</b>	<b>0.00</b>	0.44	0.55	0.11
<b>T3</b>	0.33	0.11	0.66	0.55	0.33	0.22	0.22	0.55	0.33	0.77	0.00	0.33
<b>T4</b>	0.88	0.11	0.11	<b>0.11</b>	<b>0.44</b>	<b>0.55</b>	0.77	0.11	0.22	0.33	0.77	0.00
<b>T5</b>	0.33	0.33	0.44	0.33	0.66	0.11	0.22	0.33	0.55	<b>0.22</b>	<b>0.22</b>	<b>0.66</b>

Table 3.1: Fuzzified Quantitative Database

	A0	A1	A2	B0	B1	B2	C0	C1	C2	D0	D1	D2
<b>T1</b>	0.11	0.33	0.66	#	0.11	0.22	0.33	0.33	0.44	0.11	0.55	0.44
<b>T2</b>	0.33	0.44	0.33	0.66	0.11	0.33	0.44	#	0.00	0.44	0.55	0.11
<b>T3</b>	0.33	0.11	0.66	0.55	0.33	0.22	0.22	0.55	0.33	0.77	0.00	0.33

<b>T4</b>	0.88	0.11	0.11	0.11	0.44	#	0.77	0.11	0.22	0.33	0.77	0.00
<b>T5</b>	0.33	0.33	0.44	0.33	0.66	0.11	0.22	0.33	0.55	0.22	0.22	#

Table 3.2: Fuzzified Quantitative Temporary Database

Initially a subset of the original database for an itemset, which can be reflected by it, is defined as,

**Projected Database:** Let  $X$  be a fuzzy itemset,  $T$  be a fuzzified table, and  $P$  be a directly private set. For a tuple  $t$  in  $T$ , if  $X$  publicly appears in  $t$ , then the *projection* of  $t$  with respect to  $X$ , denoted by  $(t, X)$ , is the set of entries in  $t$  that are not matched by  $X$ . If  $X$  does not publicly appear in  $t$ , then  $(t, X) = \text{null}$ . The *projected database* with respect to  $X$  is the set of nonempty projections with respect to  $X$  in  $T$ .

**Privacy-Free Fuzzy Itemsets:** A fuzzy itemset is *privacy-free* if its projected database for all  $t$  does not contain any fuzzy region of the directly private entry at all. We can check it by first finding the projected database of the itemset. If it does not contain at least one '#', then it will be called as a privacy-free fuzzy itemset.

**Non-Discriminative Fuzzy Itemsets:** A fuzzy itemset  $X$  is *non-discriminative* if every tuple in the projected database of  $X$  contains directly private entries in the same region of attribute(s). We can check it by checking for '#' in the entire initial fuzzy region of an attribute for all tuples. If the initial fuzzy region is having '#' then there is no need to check the other fuzzy regions for that attribute.

**Contrast Fuzzy Itemsets:** A fuzzy itemset  $X$  is said to be a *contrast itemset* if for any entry  $y$  belongs to  $P$  such that  $X \rightarrow y_m$ , where  $y_m = \max(y_i)$  for  $i = 1, 2, 3, \dots, R$ , appears in some tuples in  $T$ ; all the three rule have a public confidence of 0.

**Discriminative Itemsets:** An itemset  $X$  is *discriminative*, if  $X$  is the antecedent of some inauspicious rules. This can be determined by checking the projected database of  $X$ . Technically, if there is a value  $y$  such that  $X \rightarrow y_m$ , where  $y_m = \max(y_i)$  for  $i = 1, 2, 3, \dots, R$

has public and hidden confidence of at least  $\delta$  with respect to the projected database of  $X$ , then  $X$  is discriminative. From hidden confidence is the confidence of the rule  $P$  where  $P$  is the rule used to predict the value  $y_R$ . Here  $y_R$  is a fuzzy value for region  $R$  for item  $y$ .

### 3.6 Border-Based Approach for Hiding Sensitive FWIs

Border theory is very useful in determining the effect of hiding the sensitive itemsets on the non-sensitive itemsets. Thus, helps in maintaining the quality of the sanitized database. In this proposed work, the basic border-based approach which was applicable on the boolean databases is extended to the border-based approach which will be applicable for the fuzzy databases. Using this approach, one can hide sensitive fuzzy itemset while maintaining the quality of the database. A new heuristic is also proposed for candidate generation algorithm for fuzzy items.

The fuzzy terminology is used is same as described in section 2.3. In addition, consider  $r$  is the number of fuzzy regions. A transaction  $T$  is a pair  $(T_{id}, X)$  where  $T_{id}$  is a unique identifier of a transaction and  $X$  is an itemset. Given a fuzzy database  $D$ , the support of an itemset  $X$ , denoted as  $Supp(X)$  is the count of each attribute region.  $\sigma$  is the minimum support threshold. An itemset  $X$  is called an  $\sigma$ -frequent itemset if  $Supp(X) \geq \sigma$ .

Suppose,  $L$  be the complete set of  $\sigma$ -frequent itemsets in  $D$  and  $\Delta L$  be the set of sensitive itemset that needed to be hidden.  $D'$  is our sanitized database in which any  $X$  that belongs to  $\Delta L$  is not a  $\sigma$ -frequent itemset. Also, Let  $L'$  is the set of  $\sigma$ -frequent itemsets in  $D'$  and  $L_r$  is the set of all non-sensitive frequent itemsets. Our main aim is to try to minimize  $|L_r - L'|$ .

The key idea is to use the borders of non-sensitive itemsets to track the impact on the sanitized database during the hiding process and maintain its quality by selecting the modifications with minimal impact at each step. The concept of borders has been described in the last chapter (section 2.5). Let us consider there are four attributes as  $A, B,$



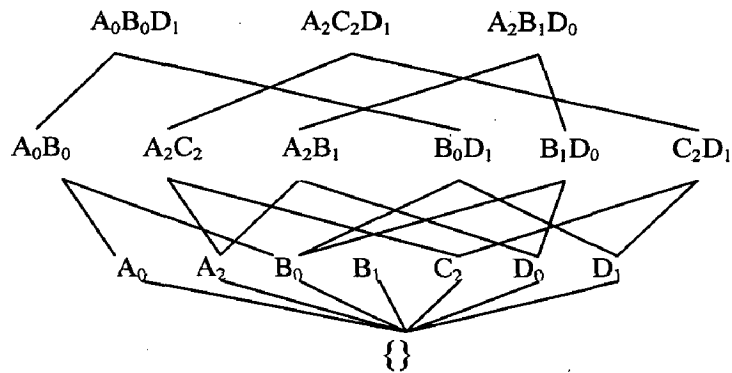


Figure 3.3: Frequent Fuzzy Itemset Lattice

$C$  and  $D$  in our database with three fuzzy sets. Thus, after fuzzification the attributes in the database will be  $A_0, A_1, A_2, B_0, B_1, B_2$  and so on. The following fig is showing the lattice of frequent fuzzy itemsets.

In this lattice, we can see that the fuzzy items from different region but same attribute can never present together. This is a fuzzy concept because after the fuzzification process each attribute gets divided into fuzzy fields. Thus, in a rule one attribute can imply one fuzzy field.

The negative border of the sensitive frequent itemset will be denoted as  $B^-$  and positive border of non-sensitive itemsets will be denoted as  $B^+$  and these two borders have been taken into consideration. For each itemset  $X$  in  $B^+$ , there will be a set of hiding candidate  $C$ , which is a set of all the transactions that contain  $X$ .

In the previous work, since the itemset are defined by either 0 or 1, the hiding candidate set can be a subset of the transaction database by the number of rows but in fuzzy database each fuzzy attribute is having some membership in each transaction, so here  $C$  will not have the number of rows less than the number of transactions in the database. The proposed work will suggest a factor, which decides the selection of transactions from this hiding candidate set.

The weight calculations will be the same process as suggested for the boolean dataset. For revision purpose, the weight is calculated to prioritize the itemsets in  $B^+$  from being affected by itemset deletion. Also, the definition of affected border of itemset  $X$  and the relevance bit vector will be the same (described in section 2.5).

The main approach is, for each element in  $B^-$ , the affected itemset of  $B^+$  and weights are found out. Then, a candidate is selected for deletion and the item which puts minimal impact will be deleted. In each iteration,  $C$ , weights and the database also updated. The proposed algorithm is shown in the given fig.

<p><i>Input:</i> A database <math>D</math>, the set <math>L</math> of <math>\sigma</math>-frequent itemset in <math>D</math> and the set of sensitive itemsets <math>L'</math></p> <p><i>Output:</i> <math>D'</math> with the maintained quality</p> <p><i>Method:</i></p> <p>    Compute <math>Bd^-</math> and <math>Bd^+</math>;</p> <p>    Sort itemsets in <math>Bd^-</math> in descending order of length and ascending order of support;</p> <p>    for each <math>X</math> in <math>Bd^-</math> do</p> <p>        Compute <math>Bd^+ x</math> and <math>w(B_j)</math> where <math>B_j</math> in <math>Bd^+ x</math>;</p> <p>        Initialize <math>C</math> (<math>C</math> is the set of hiding candidates of <math>X</math>);</p> <p>        for(<math>i = 0</math>; <math>i &lt; Supp(X) - \sigma</math>; <math>i++</math>) do</p> <p>            Sort the candidate set, according to the decreasing order of fuzzy membership values for <math>x_i</math>;</p> <p>            Find <math>u_i = (T_i, x_i)</math> from the sorted <math>C</math> such that <math>I(u_i) = \text{Min} \{I(u) \mid u \text{ in } C\}</math>;</p> <p>            Update <math>C = C - \{(T, x) \mid T = T_i\}</math>;</p> <p>            Update <math>w(B_j)</math> where <math>B_j</math> in <math>Bd^+ x</math>;</p> <p>            Update database <math>D</math>;</p> <p>        Output <math>D' = D</math>;</p>
--

Figure 3.4: Border-based Approach for Hiding Sensitive Fuzzy Weighted Itemsets

The candidate selection step gives the core of the border-based approach, which is to efficiently find the hiding candidate with minimal impact on border. The proposed

approach finds  $u_i$  in an efficient manner for fuzzy itemsets. The justification is given below.

In boolean databases, we calculate support by the ratio of itemset count and the number of total transactions. We can conclude from this that, since all the counted itemset are having a "1" on the place it is present; it gives its 100% membership for the support count Thus the support-deciding-factor is 1 for all elements. However, this is not the case with fuzzy items. In fuzzy databases, the support-deciding-factor is different due to membership of the itemsets so in the candidate selection step we have taken the greatest value so that we can distinguish the candidates which affect the selection most.

## CHAPTER 4

# SYSTEM DESIGN AND IMPLEMENTATION

---

A complete description of the system design and implementation has been given in this chapter. First of all, the used database has been described followed by a brief description of the used platform. All the basic modules and the procedures used in the coding have also been described.

### 4.1 Database Used

There are a large number of algorithms that are implemented on the Boolean database. The basic concept of algorithms for quantitative databases works in different manner. The database used in this dissertation work is of quantitative type.

The breast cancer is one of the most common tumor related disease among women in Korea and throughout the world. We have used well known WBCD which is provided by the University of Wisconsin Hospitals based on microscopic examination of breast masses with fine needle aspirate tests. It is the breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The database is denoted on 15<sup>th</sup> July, 1992 to the UCI Machine Learning Repository.

#### *Basic Details:*

- Title: Wisconsin Breast Cancer Database (January 8, 1991)
- Sources:
  - Dr. William H. Wolberg (physician)
  - University of Wisconsin Hospitals
  - Madison, Wisconsin
- Number of Instances: 699
- Number of Attributes: 10 plus the class attribute

**Attribute Information:**

	<b>Attributes</b>	<b>Domain</b>
1.	Sample Code number	Patient ID number
2.	Clump Thickness	1-10
3.	Uniformity of Cell Size	1-10
4.	Uniformity of Cell Shape	1-10
5.	Marginal Adhesion	1-10
6.	Single Epithelial Cell Size	1-10
7.	Bare Nuclei	1-10
8.	Bland Chromatin	1-10
9.	Normal Nucleoli	1-10
10.	Mitoses	1-10
11.	Class of Tumor	2 for benign 4 for malignant

Table 4.1: Database Details

## 4.2 Code Platform

For coding and testing purposes, two most suitable platforms have been chosen. These two platforms are JAVA and MATLAB. Java is used for the basic implementation while Matlab is used to generate the fuzzy membership values. A brief description about each platform will be given below. NetBeans IDE 6.5.0 is used for using JAVA. Netbeans provide application programming interface for java.

*JAVA:* Object-orientation is the core of Java. The features of Java make it the most suitable platform for this work. Inheritance and Abstraction is the main feature of object orientation which is used in Java during implementation. Classes like Itemset defines the basic functionality in a program while hiding its implementation details. Some of classes are derived from the super class to add new functionalities to the class. Like Apriori is written in a class and then its main features is used in another class called Fuzzy-Apriori to add the new features of fuzzy concept.

Large databases are normally stored in separate spreadsheet like MS Excel or MS Access. Java provides easy connectivity using JDBC database connectivity. Only few lines of codes have to be written to access the database.

Java Swing has been used for designing GUI. It provides a lot of classes and interfaces using which an attractive and user friendly GUI can be designed.

*MATLAB*: MATLAB (MATrix LABoratory) is popular for its large repository of inbuilt functions. Very large programs get converge to a few lines of code using Matlab functions. MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages.

In this implementation, Fuzzy Toolbox and Neural Network Toolbox are used to get the membership from the database. These inbuilt functional toolboxes help in the process simulation and analysis. Also to draw the membership function a toolbox named as Membership Function Editor is used.

### **4.3 Modules and Procedures**

In this work, a complete fuzzy system is designed which, as an input takes a quantitative database, the sensitive itemsets, minimum support and weights and provide a well maintained sanitized database which is free from all the sensitive itemsets and highly predictive non-sensitive itemsets. This section describes each module and the classes and functions used to implement these modules.

#### ***Clustering:***

Clustering is used to classify the data objects into classes or clusters. These clusters will be used by artificial neural network for classification. The dataset we have used is having 10 attributes. The number of the class in which the dataset is to be classified is two. Fuzzy C-means clustering is used here to get classes.

A function FCM.m is used to get the clusters. prototype of the function is as follows:

$$[\text{CENTER}, \text{U}, \text{OBJ\_FCN}] = \text{FCM}(\text{DATA}, \text{N\_CLUSTER})$$

finds N\_CLUSTER number of clusters in the data set DATA. DATA is size M-by-N, where M is the number of data points and N is the number of coordinates for each data point. The coordinates for each cluster center are returned in the rows of the matrix CENTER. The membership function matrix U contains the grade of membership of each DATA point in each cluster. The values 0 and 1 indicate no membership and full membership respectively. Grades between 0 and 1 indicate that the data point has partial membership in a cluster. At each iteration, an objective function is minimized to find the best location for the clusters and its values are returned in OBJ\_FCN.

### ***Classification:***

Classification process is done by using artificial neural network. Feed forward multilayer neural network can be used to generate the membership functions from the labeled data. As given in [technique MF], the output values of a sigmoid activation function of a neuron are quite similar to the membership values. A sigmoid function is given as,

$$f(t) = 1 / (1 + e^{-t}) \quad (17)$$

The number of neurons in the input layer is set to the number of features or attributes and the number of neurons in the output layer is set to the number of classes we want to classify the data points.

In order to generate the class membership values, the multilayered network must be trained using a suitable algorithm. I have used the Feed forward backpropagation algorithm to train the network. This algorithm not only sends the output of one layer to the input of next layer, but also refines the network by reducing the learning errors in each iteration.

The following figure shows the basic design of the neural network used in this work. The sigmoid activation function is used in each neuron, since as pointed out earlier, sigmoid

activation function gives values similar to membership function. 10 neurons are used in first layer and 2 neurons in the last layer since we require 10 attributes in the input data

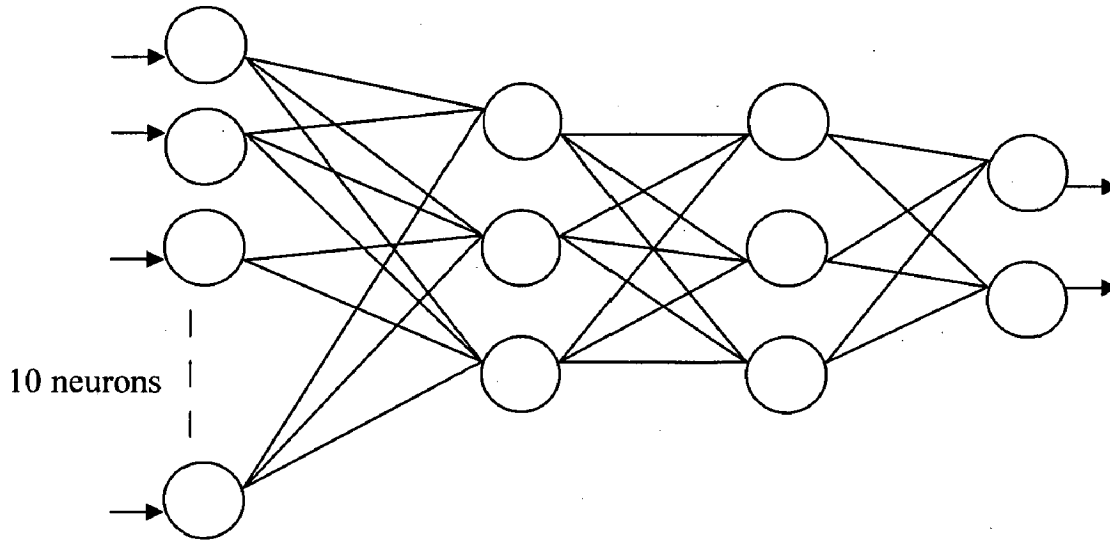


Figure 4.1: Multilayer Feed-forward Neural Network

and 2 classes. There were 16 instances in the database with missing value. These tuples are removed to construct a new dataset with 683 instances. The first 400 instances in the new dataset are chosen as the training set and the remaining 283 as the test set. The maximum number of epochs is set to 500 and the learning rate is set to 0.03.

After getting membership values for each class, the membership function for each class is defined. There are several types of membership functions such as triangular, trapezoidal, and Gaussian, to name a few. The triangular membership function is used in this work. NNTOOL of MATLAB is used to design the desired neural network.

***Fuzzification:***

Artificial Neural Network gives the membership values of each data point for each class. For the fuzzification of the database it is needed to draw the membership function. Since the sigmoid activation function is used to get the membership values the membership function we get, should be in the form of sigmoid function. We will get fuzzy regions in the form of sigmoid function for each class.



In the fuzzification process, the quantitative data values of the database for each item is entered into a fuzzy set represented as,  $(f_{j1} / R_{j1}, f_{j2} / R_{j2}, \dots, f_{jl} / R_{jl})$  by using the given membership function for the item quantities, where  $l$  is the number of fuzzy regions.

A class is written in JAVA named as fuzzification.java. Function establishes JDBC connectivity from the database (MS Access). It contains the following functions:

***Dataptdistance***(int [] *datapoint*) : calculates the distance of the membership value from the centroids of two classes.

### ***Fuzzy Frequent Weighted Itemset Mining:***

This is one of the proposed works. This process gives frequent weighted itemsets using the proposed algorithm. Inputs are two files, one for the input transactions and another for the weights which is to be applied to the items. Minimum support and confidence thresholds are also taken from the user.

A class is written in java named as Apriori.java which is used to find out number of frequent itemset in boolean databases. A second class is derived from it named as FuzzApriori.java. Using inheritance a large amount of duplicacy is removed. Itemset.java is a class which is used to show an itemset and all the functions defined on it.

The main functions used in these classes are namely:

***countSingles***() : count the number of the single itemsets.

***orderArray***() : order the items in the itemset.

***combinations*** (Itemset *itemset*) : returns all the combinations of a set except null.

***resizedata*** (Itemset[][] *itemset*) : resizes the 2D array of itemsets by deleting the itemsets having support lower than minimum threshold.

***FreqItemset*** (Itemset[][] *itemset*) : returns all the frequent itemsets.

***combine*** (Itemset [][] *itemset*) : combine two large itemsets to get candidate itemsets.

***largeItemset*** (Itemset [][] *itemset*) : returns all itemsets having support greater than minimum support threshold.

***pruneitems***(Itemset *itemset*): delete all items having lower support than the minimum support threshold.

***chkRegions***(Itemset *itemset*): checks whether two fuzzy candidate itemsets are belonging to same attribute or not.

***support***(Itemset *itemset*): calculates support of the itemset.

### ***Highly Predictive Non-Sensitive Itemsets Extraction:***

This module use to find out the inauspicious itemsets that are having a strong predictive power. Using three constraints, as given in the proposed work, an itemset is checked for being an inauspicious itemset. *Adversarial.java* is the class name used to implement.

The main functions used in this class are:

***fuzzconfidence***(Itemset *itemset*, Double *val*, String *attr*): calculate confidence.

***projectDB***(Itemset *itemset*, String *tuple*): returns the projected database for tuple *tuple*.

***PDB\_X***(Itemset *itemset*): returns the projected database for itemset *itemset*.

***checkPFree***(Itemset *itemset*, String *tuple*, Itemset *PrivateSet*): checks for privacy free.

***chknonDisc***(Itemset *itemset*, Itemset *PrivateSet*): checks for non-discriminative.

***chk\_Contrast***(Itemset *itemset*, Itemset *PrivateSet*): check for contrast.

***chkDiscriminative***(Itemset *itemset*, Itemset *PrivateSet*, Double *delta*): check for discriminative with respect to hidden confidence *delta*.

### ***Border-Based Approach for Hiding Frequent Fuzzy Weighted Itemsets:***

This is also one of the proposed works. This process takes the sensitive fuzzy weighted itemsets as well as the inauspicious non-sensitive fuzzy weighted itemsets and hides the sensitive fuzzy weighted itemsets in such a way that the quality of sanitized database is maintained. *Border.java* is the class used for it.

The main functions used in this class are:

***PostiveBorder***(): calculates the positive border of an itemset.

***NegativeBorder***(): calculates the negative border of an itemset.

***HidingCandidate***(): returns all the transactions selected to for hiding.

***AffectedBorder\_Candidate***(Itemset *itemset*): returns all the itemsets of positive borders that are affected by itemset.

***RelevanceBitVector***(Itemset *itemset*, char *c*): returns the relevance bit vector for itemset.

***Impact***( int[] *RelBitVector*, Double [] *weights*): calculates the impact.

***Weights***(): calculates weight.

In addition to these functions other utility functions are also present like ***combination***(Itemset *itemset*) which is used to find all the combinations of an itemset, ***Support***(Itemset *itemset*) which is used to find out the support of an itemset, ***StrtoArr***(String *str*) and ***ArrtoStr***(Char[] *str*) used to convert character array to String and vice-versa.

## CHAPTER 5

### RESULTS AND ANALYSIS

---

Various experiments have been conducted on the proposed work using a real life dataset. Results obtained are shown in this chapter to show the effectiveness of the work. A brief analysis of these results has also been performed and discussed. We have taken minimum support as 0.2.

#### 5.1 Results

##### 1. Membership function

After getting class labels using clustering and building the artificial neural network, the membership values of different data points is mapped graphically. Since sigmoid function is used as activation function, the resultant membership functions obtained for the two classes are the two sigmoid curves. The membership function for one of the two classes is shown in fig. 5.1. mf1 is showing the membership function for class 1. For class 2 function curve obtain is complementary to the function curve of class 1.

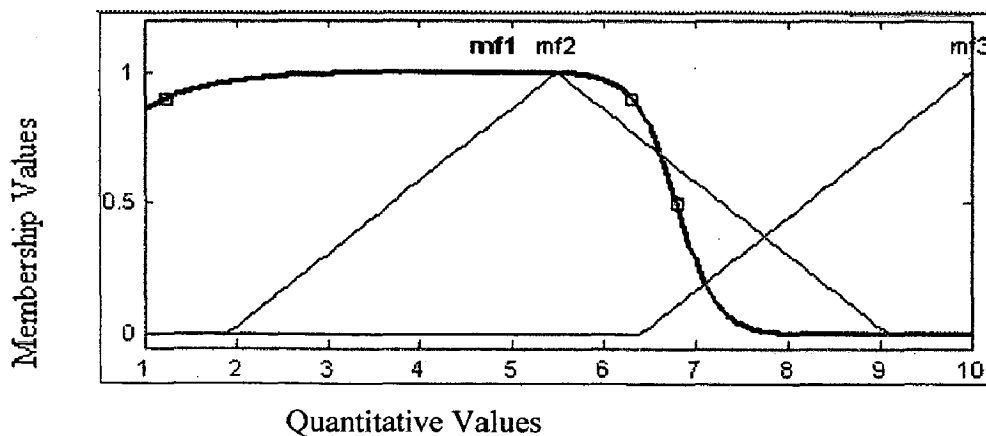


Figure 5.1: Membership Function obtained for Class 1

##### 2. Frequent Fuzzy Weighted Itemset Mining (FFWIM)

Various experiments were conducted to test the performance of the proposed algorithm. We have taken popular breast cancer database from UCI data repository. The unseen

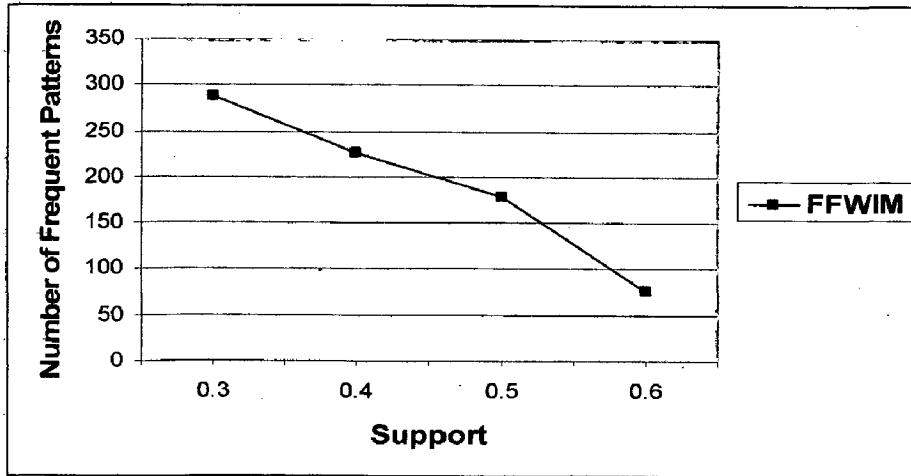


Figure 5.2: Effect of Support on Number of Frequent Weighted Itemsets

dataset contained 450 records. All the values are of quantitative type. Fig. 5.2 shows the effect of the minimum support threshold over the number of frequent patterns.

Weight applied to an item shows its importance to the user. An infrequent itemset will become frequent if extra weight is applied to it. Since support count is a major metric to quantify the importance of an itemset, the weights applied affect the support count of an itemset. A relative comparison has been shown in fig 5.3. Graph shows the affect of support count on the number of frequent itemsets in the frequent fuzzy weighted itemset mining and frequent fuzzy itemset mining.

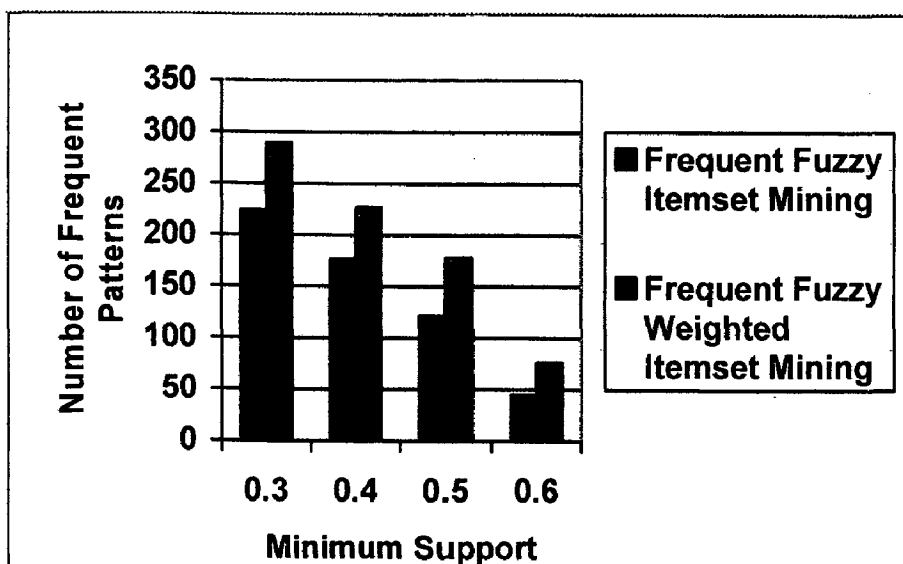


Figure 5.3: Comparison between FFIM and FFWIM

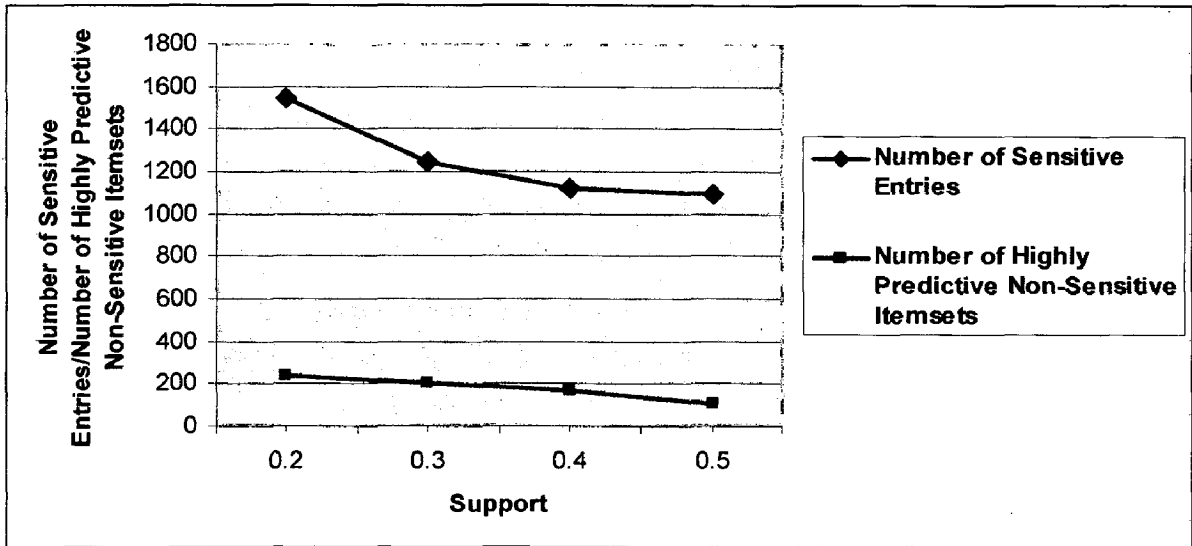


Figure 5.4: Effect of Support & no. of Sensitive entries on no. of highly predictive non-sensitive itemsets

### 3. Highly Predictive Non-Sensitive Itemsets Extraction

Graph in fig. 5.4 shows the affect of the support and the number sensitive entries to the number of highly predictive non-sensitive itemsets.

### 4. Border-Based Approach for Hiding Frequent Sensitive Fuzzy Weighted Itemsets

In fig. 5.5 we have shown the comparison between the proposed border-based approach for data hiding and the previous data hiding approach [3]. To measure the effectiveness,

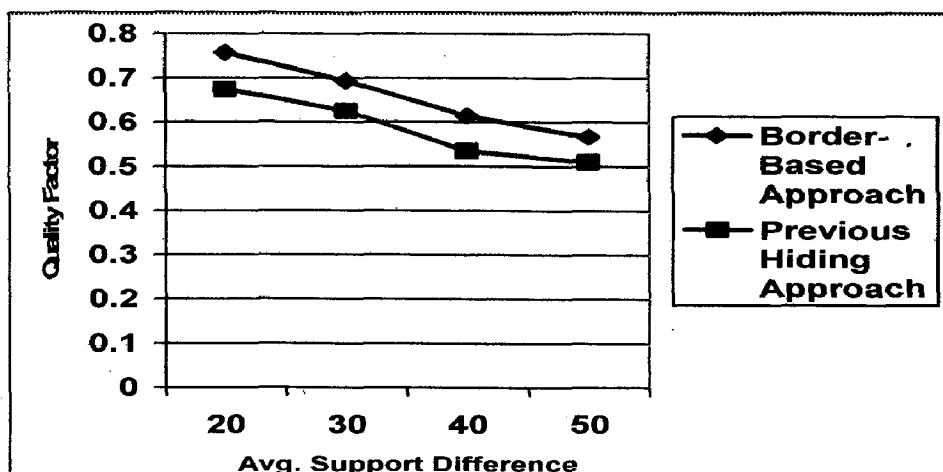


Figure 5.5: Effect of Average Support Difference on Quality Factor

we have compared the set of non-sensitive frequent itemset  $L_r$  with the set  $L'$  of frequent itemsets in  $D'$ . The *quality*  $Q$  of the result dataset  $D'$  could be measured as:  $Q = |L'|/|L_r|$ . A new term, average support difference introduced in [19] is taken as a factor to compare the results. The average support difference considers the support counts for negative border of the sensitive frequent itemsets. It is defined as,  $avg\_suppdiff = \sum (Supp(X_i) - \sigma) / |B|$  for all  $X_i$  belongs to  $B$  and  $\sigma$  is the support count.

## 5.2 Analysis

We have proposed a frequent fuzzy weighted itemset mining algorithm on quantitative database. Since, the work is a modification in the basic algorithm there will not be any change in the performance parameter. However, comparing these two algorithms it can be observed that the memory usage and execution time of the proposed algorithm exceeds that of the basic algorithm by a small factor as shown in fig 5.6.

Fig. 5.4 shows the effect of increasing support count and the number of fuzzy weighted sensitive entries on the number of highly predictive fuzzy weighted non-sensitive itemsets. The number of frequent patterns will decrease as the support count increases. Also we can analyze from the graph that the number of highly predictive non-sensitive itemsets decreases according to the decrement of sensitive entries.

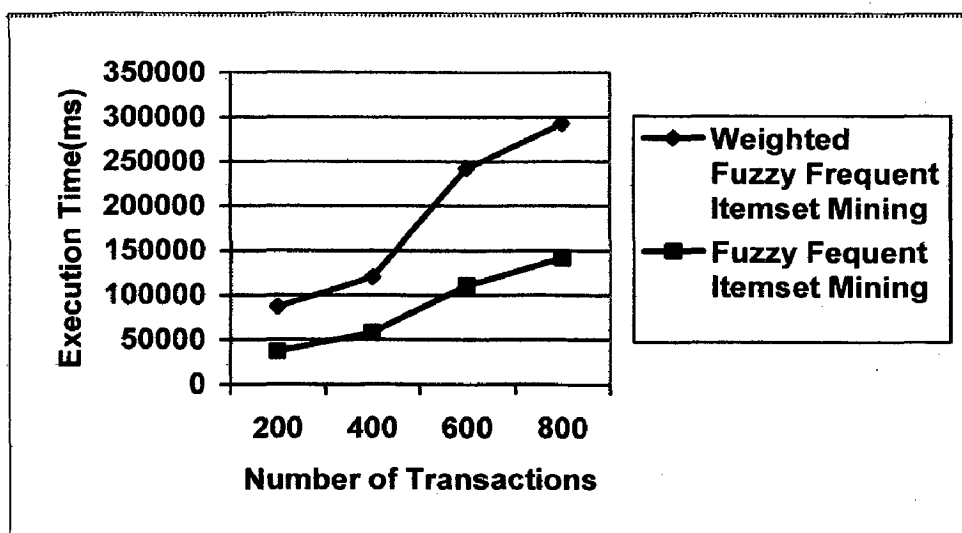


Figure 5.6: Effect of Number of Transaction on Execution Time

In the border-based approach, to measure the effectiveness of the approach, it is natural to compare the set of non-sensitive frequent itemset  $L_r$  with the set  $L'$  of frequent itemsets in  $D'$ . As this approach does not introduce new frequent itemsets in  $D'$ , the *quality*  $Q$  of the result dataset  $D'$  could be measured as:  $Q = |L'| / |L_r|$ . At any instant, the average support difference will be given by,  $avg\_suppdiff = \sum ((supp(X_i) - \sigma) / |B'|$  where  $X_i$  belongs to  $B'$  and  $\sigma$  is the support count [15]. Here we have only considered the itemsets in  $B'$  since it shows all the sensitive itemsets on the negative border.



In the border-based approach, to measure the effectiveness of the approach, it is natural to compare the set of non-sensitive frequent itemset  $L_r$  with the set  $L'$  of frequent itemsets in  $D'$ . As this approach does not introduce new frequent itemsets in  $D'$ , the *quality*  $Q$  of the result dataset  $D'$  could be measured as:  $Q = |L'| / |L_r|$ . At any instant, the average support difference will be given by,  $\text{avg\_suppdiff} = \sum ((\text{supp}(X_i) - \sigma) / |B|$  where  $X_i$  belongs to  $B$  and  $\sigma$  is the support count [15]. Here we have only considered the itemsets in  $B$  since it shows all the sensitive itemsets on the negative border.

## CHAPTER 6

# CONCLUSION AND FUTURE WORK

---

In this chapter, the work is concluded. Some future works are also suggested so that some more research works can be initiated in the field of privacy preserving fuzzy itemset mining.

### 6.1 Conclusion

A generalized framework has been proposed for hiding fuzzy sensitive weighted itemsets in quantitative databases. Proposed works shows the affect of prioritized itemsets on the process of mining frequent fuzzy weighted itemsets. In order to be able to use frequent itemsets in practical applications, like targeted marketing or customer retention, one must be able to prioritize between various frequent itemsets mined with respect to the magnitude of the effect they produce on the outcome.

There are cases where some non-sensitive itemsets can be used to predict to sensitive entries. In the proposed work we have consider these types of non-sensitive fuzzy itemsets to improve the accuracy of privacy preservation.

In order to hide sensitive fuzzy itemsets, a border-based approach is proposed. Using borders, the quality of the sanitized databases can be preserved from the itemset hiding process. The contribution of this work includes the minimization of the side effect on the sanitized databases.

A number of experiments have be en conducted to evaluate the effectiveness of the proposed algorithms. A real life case data has been taken to conduct experiments. Respective results have been shown for quantitative dataset.

## **6.2 Future Work**

To obtain class labels for the classifiers, we have used Fuzzy C-Means clustering. While other techniques like K-Means Clustering can also be used. Choice of this clustering algorithm should be done on the basis of low learning error of classifier.

We have used artificial neural network as classifier. Other techniques can also be used to explore the area of automated generation of membership functions.

Proposed work used a static approach of applying weights. However, other approaches can also be explored to generate weights dynamically.

## REFERENCES

---

- [1] V. Verkios, E. Bertino, I. G. Fovino, L. P. Provenza, Y. Saygin and Y. Theodoris, "State-of-the-art in Privacy Preserving Data Mining," *Special Interest Group on Management Of Data (SIGMOID)*, March 2004, Vol.33, No. 1, pp.50-57.
- [2] H. Mannila. "Local and Global Methods in Data Mining: Basic Techniques and Open Problems," *Proc. 29th Int'l Colloquium on Automata, Languages, and Programming, ICALP 2002*, pp. 57-68
- [3] T. Berberoglu and M. Kaya, "Hiding Fuzzy Association Rules in Quantitative Data," *Proc. 3rd Int'l Conf. on Grid and Pervasive Computing*, 2008, Vol. 33, pp. 387-392.
- [4] S. Agrawal, V. Krishnan and J.R. Haritsa, "On Addressing Efficiency Concerns in Privacy Preserving Mining," *Proc. 9th Int'l Conf. on Database Systems for Advanced Applications (DASFAA-2004)*, Jeju Island, Korea. March 2004, pp. 113-124.
- [5] M. Gibbs, G. Shanks and R. Lederman, "Data Quality, Database Fragmentation and Information Privacy," *Journal of Surveillance and Society*, January 2005, Vol. 3, No.1, pp. 45-58.
- [6] R. Agrawal and R. Srikant, "Privacy Preserving Data mining," *Proc. ACM SIGMOID Conference on Management of Data*, Dallas, Texas, May 2000, pp. 439-450.
- [7] L.A. Zadeh, "Fuzzy sets," *Journal of Information and Control*, Vol. 8, 1965, pp. 338-353.
- [8] C. Czarnecki, R. John and S. Bennett, "The Application of Fuzzy Logic to Real Time Multiple Robot Collision Avoidance," *Proc. ICSC Fuzzy Logic Symposium, 1995*, pp.116-121.
- [9] S. Medasani, J.Kim and R. Krishnapuram, "An Overview of Membership Function Generating Techniques for Pattern Reorganization," *Int'l Journal of Approximate Reasoning*, 1998, pp. 391-417.

- [10] J.C. Can and P.A. Nava, "A Fuzzy Method for Automatic Generation of Membership Function using Fuzzy Relations from Training Examples," *Proc. Annual Meeting of the North American Fuzzy Information Processing Society*, 2002, pp. 158-162.
- [11] A. Carling, "Introducing Neural Networks," *Sigma Press*, Wilmslow, UK, 1992, pp. 233-234.
- [12] Kuok, Chan Man, Fu Ada Wong, Man Hon, "Mining Fuzzy Association Rules in Databases," *Special Interest Group on Management Of Data (SIGMOD)*, Vol. 27, 1998, pp. 41-46.
- [13] Dubois Didier, Eyke Hullermeier, Prade Henri, "A Note on Quality Measures for Fuzzy Association Rules," *Proc. Tenth Int'l Fuzzy Systems Association World Congress on Fuzzy Sets and Systems*, Springer, 2003 pp. 356-353.
- [14] Srikant R, Agrawal R., "Mining quantitative association rules in large relational tables," *Proc. ACM SIGMOD Int'l Conf. on Management of Data, Monreal, Canada*, 1996, pp. 1-12.
- [15] T.P. Hong, C.S. Kuo, S.C. Chi, "Mining Association Rules from Quantitative Data," *Journal of Intelligent Data Analysis*, Vol. 3(5), 1999, pp. 363-376.
- [16] T.P. Hong, C.S. Kuo and S.L. Wang, "A Fuzzy AprioriTid Mining Algorithm with Reduced Computational Time," *Proc. Fourth Int'l Conference on Machine Learning and Cybernetics*, Guangzhou, August 2005, pp. 1812-1815.
- [17] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim and V.S. Verykios, "Disclosure Limitation of Sensitive rules," *Proc. IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, 1999, pp. 45-52.
- [18] C. Aggrawal and P. Yu, "Privacy Preserving Data Mining: Models and Algorithms," Vol. 34, pp. 267-268.
- [19] X. Sun and P.S. Yu, "A Border-Based Approach for Hiding Sensitive Frequent Itemsets," *Proc. Fifth IEEE Int'l Conf. Data Mining (ICDM '05)*, 2005, pp. 426-433.
- [20] G. Moustakides and V.S. Verykios, "A Max-Min Approach for Hiding Frequent Itemsets," *Proc. Sixth IEEE Int'l Conf. Data Mining (ICDM '06)*, 2006, pp. 502-506.

- [21] A.G. Divanis and V.S. Verykios, "An Integer Programming Approach for Frequent Itemset Hiding," *Proc. ACM Conf. Information and Knowledge Management (CIKM '06)*, Nov. 2006, pp. 748-757.
- [22] C.C. Aggarwal, J. Pei and B. Zhang, "On Privacy Preservation against Adversarial Data Mining," *Knowledge Discovery and Data Mining (KDD, 06)*, Philadelphia, USA, August 2006, pp. 510-516.
- [23] [http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Original\)](http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)).  
(Dataset)

- [21] A.G. Divanis and V.S. Verykios, "An Integer Programming Approach for Frequent Itemset Hiding," *Proc. ACM Conf. Information and Knowledge Management (CIKM '06)*, Nov. 2006, pp. 748-757.
- [22] C.C. Aggarwal, J. Pei and B. Zhang, "On Privacy Preservation against Adversarial Data Mining," *Knowledge Discovery and Data Mining (KDD, 06)*, Philadelphia, USA, August 2006, pp. 510-516.
- [23] [http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Original\)](http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)).  
(Dataset)

## LIST OF PUBLICATIONS

---

[1] Mridula Verma, Durga Toshniwal, "Fuzzy Weighted Sensitive Itemset Mining in Quantitative Databases", 1<sup>st</sup> International Conference of Multi-Label Data, Slovenia (under communication).

[2] Mridula Verma, Durga Toshniwal, "A Border Based Approach for Hiding Fuzzy Weighted Sensitive Itemsets", International Journal of Knowledge and Information Systems, to be published by **Springer** (under communication).



## APPENDIX: SOURC CODE LISTING

---

### 1. Fuzzy Weighted Itemset Mining (FWI Mining)

#### 1. *Counting the number of single itemsets*

```
protected int[] [] countSingles()
{
    int[] [] countArray = new int[numCols+1][2];
    for (int index=0 ; index<countArray.length ; index++)
    {
        countArray[index][0] = index;
        countArray[index][1] = 0;
    }

    for(int rowIndex=0 ; rowIndex<dataArray.length ; rowIndex++)
    {
        if (dataArray[rowIndex] != null)
        {
            for (int colIndex=0;colIndex<dataArray[rowIndex].length;
                colIndex++)
                countArray[dataArray[rowIndex][colIndex]][1]++;
        }
    }

    return(countArray);
}
```

#### 2. *Order items in itemsets according to their support count*

```
private void orderCountArray(int[] [] countArray)
{
    int attribute, quantity;
    boolean isOrdered;
    int index;

    do
    {
        isOrdered = true;
        index = 1;
        while (index < (countArray.length-1))
        {
            if (countArray[index][1] >= countArray[index+1][1])
                index++;
            else
            {
                isOrdered = false;
                attribute = countArray[index][0];
                quantity = countArray[index][1];
                countArray[index][0] = countArray[index+1][0];
                countArray[index][1] = countArray[index+1][1];
                countArray[index+1][0] = attribute;
                countArray[index+1][1] = quantity;
            }
        }
    }
}
```

```

        index++;
    }
}
while (isOrdered==false);
}

```

### 3. *Combinations of items in itemsets*

```

String[] combinations(Itemset itemset)
{
    int n;
    int [] x = new int[20];
    int len = itemset.length;
    int com = (int)Math.pow(2,len);
    NumComb = com;
    String [][]combi = new String[com-1][len];
    char [][] ex = new char[com-1][len];

    for(int j = 0 ; j < com-1 ; j++)
    {
        n = j+1;
        for(int i = 0 ; i<len ; i++)
        {
            x[i] = n % 2;
            ex[j][i] = Integer.toString(x[i]).charAt(0);
            n = n / 2 ;
        }
    }
    int r=0;
    for(int i=0;i<com-1;i++)
    {
        r=0;
        for(int j=0;j<len;j++)
            if(ex[i][j]=='1')
            {
                combi[i][r]=s[j];
                r++;
            }
    }
    String [] C = new String[com-1];

    for(int i=0;i<com-1;i++)
        C[i] = StrArrToStr(combi[i]);

    return C;
}

```

### 4. *Resize Dataset According to Minimum Threshold*

```

public void resizeInputData(double percentage)
{
    numRows = (int) ((double) numRows*(percentage/100.0));
    short [][] trainingSet = new short[numRows][];
}

```

```

for (int index=0;index<numRows;index++)
    trainingSet[index] = dataArray[index];

dataArray = trainingSet;
minSupport = (numRows * support)/100.0;
}

```

## 5. *Frequent Itemsets*

```

private int outputFrequentSets(int number, short[] itemSetSofar,
    int size, TtreeNode[] linkRef)
{
    if (linkRef == null)
        return(number);

    for (short index=1; index < size; index++)
    {
        if (linkRef[index] != null)
        {
            if (linkRef[index].support >= minSupport)
            {
                short[] newItemSet = realloc2(itemSetSofar, index);
                outputItemSet(newItemSet);
                number = outputFrequentSets(number + 1, newItemSet, index,
                    linkRef[index].childRef);
            }
        }
    }
    return(number);
}

```

## 6. *Combine two large itemsets*

```

protected FuzzyDataItem[] realloc1(FuzzyDataItem[] oldItemSet,
    FuzzyDataItem newElement)
{
    if (oldItemSet == null)
    {
        FuzzyDataItem[] newItemSet = new FuzzyDataItem[1];
        newItemSet[0] = new FuzzyDataItem(newElement);
        return(newItemSet);
    }
    int oldItemSetLength = oldItemSet.length;
    FuzzyDataItem[] newItemSet = new
        FuzzyDataItem[oldItemSetLength+1];

    int index;
    for (index=0; index < oldItemSetLength; index++)
        newItemSet[index] = new FuzzyDataItem(oldItemSet[index]);
    newItemSet[index] = newElement;

    return(newItemSet);
}

```

## 7. *Returns all the Large Itemsets*

```
private int outputFrequentSets(int number, short[] itemSetSofar,
    int size, TTreeNode[] linkRef)
{
    if (linkRef == null) return(number);

    for (short index=1; index < size; index++)
    {
        if (linkRef[index] != null)
        {
            if (linkRef[index].support >= minSupport)
            {
                short[] newItemSet = realloc2(itemSetSofar, index);
                outputItemSet(newItemSet);
                number = outputFrequentSets(number + 1, newItemSet, index,
                    linkRef[index].childRef);
            }
        }
    }
    return(number);
}
```

## 8. *Prune items with low support from Tree*

```
protected boolean pruneLevelN(TTreeNode [] linkRef, int level)
{
    int size = linkRef.length;
    if (level == 1)
    {
        boolean allUnsupported = true;
        for (int index1=1; index1<size; index1++)
        {
            if (linkRef[index1] != null)
            {
                if (linkRef[index1].support < minSupport)
                    linkRef[index1] = null;
                else
                {
                    numFrequentSets++;
                    allUnsupported = false;
                }
            }
        }
        return(allUnsupported);
    }
    for (int index1=level; index1<size; index1++)
    {
        if (linkRef[index1] != null)
        {
            if (linkRef[index1].childRef != null)
            {
                if (pruneLevelN(linkRef[index1].childRef, level-1))
                    linkRef[index1].childRef=null;
            }
        }
    }
}
```

```

    }
  }
  return(false);
}

```

## 2. Highly Predictive Non-Sensitive FWI Extraction

### 1. Projected Database:

```

String [][] project_DB(Itemset itemset, String tuple)
{
  String [][] Projdb = new String [2][4];
  int i = 0 , j = 0 , x = 0 , count = 0 , k=0;
  int [] itemset_index = itemset.getIndex();

  try
  {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection
      ("Jdbc:Odbc:Dissertdb");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from Table2
      where TID = '" + tuple + "'");

    while(rs.next())
    {
      for(i=0;i<4;i++)
      {
        if (i != itemset_index[j])
        {
          Projdb[0][count] =String.valueOf
            (rs.getDouble(i+2));
          Projdb[1][count] = itemset.getClassName(i);
          count++;
        }
        j++;
        if(j==itemset_index.length)
          j--;
      }
    }
    con.close();
  }
  catch(Exception exp)
  {
    exp.printStackTrace();
  }

  c = count;
  return Projdb;
}

```

```

String [][] PDB_X(Itemset itemset)
{
  int i = 0, j = 0, count = 0 , k = 0;
  int x =0;

```

```

String TID;
int itemset_len = itemset.attrname.length;
int m = 5 - itemset_len;
String [][] PDB = new String [10] [5];
int [] itemset_index = itemset.getIndex();
Double str =null;

try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection
        ("Jdbc:Odbc:Dissertdb");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from Table2");

    while(rs.next())
    {
        count =0;
        TID = rs.getString(1);
        for(i = 0 ; i<itemset_len ; i++)
        {
            str = rs.getDouble(itemset_index[i]+2);
            if(itemset.value[i].equals(str))
                count++;
            if(count==itemset_len)
            {
                x++;

                String [][] ProjDB_X = project_DB(itemset,TID);

                PDB[k] [0] = TID;
                PDB[k+1] [0] = TID;
                for(j = 1 ; j < m ; j++)
                {
                    PDB[k] [j] = ProjDB_X[0] [j-1];
                    PDB[k+1] [j] = ProjDB_X[1] [j-1];
                }
                k = k + 2 ;
            }
        }
        System.out.println();
    }
    c = 2*x;
    con.close();
}
catch(Exception exp)
{
    exp.printStackTrace();
}
return PDB;
}

```

## 2. *fuzzConfidence*:

```

boolean confidence (Itemset itemset, Double val , String attr)
{
    boolean conf = true;

```

```

Double [][] rsltset = new Double[20][20];
int i = 0 , j = 0 , x = 0 ;
int [] itemsetindex = itemset.getIndex();
int index = itemset.getIndex(attr);

try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection
        ("Jdbc:Odbc:Dissertdb");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from Table2");

    while(rs.next())
    {
        for(j=1;j<5;j++)
            rsltset[i][j-1] = rs.getDouble(j+1);
        i++;
    }

    for(i=0;i<5;i++)
        for(j=0;j<4;j++)
            {
                for(int k=0;k<itemset.attrname.length;k++)
                    if(rsltset[i][itemsetindex[k]].equals(itemset.value[k]))
                        x++;

                if(x==itemset.attrname.length)
                    if(rsltset[i][index].equals(val))
                        conf = false;
                break;
            }
        con.close();
    }
catch(Exception exp)
{
    exp.printStackTrace();
}
return conf;
}

```

### 3. *Checking for Privacy Free Itemsets:*

```

Boolean chk_PFree(Itemset itemset, String tuple, Itemset [] P)
{
    Boolean b = true;
    String [][] ProjDB_X = project_DB(itemset,tuple);
    int j = 0 ;
    Double [][] PrivateSet = new Double [5][5];
    int [] numItemsinPrivate = new int[5];
    int TotnumPrivateitems = P.length;

    for(int i = 0 ; i < TotnumPrivateitems ; i++)
        if(P[i].tuple.equals(tuple))
            {
                PrivateSet[j] = P[i].value;
            }
}

```

```

        numItemsinPrivate[j] = P[i].value.length;
        j++;
    }

    int numPrivateSelected = j;
    for(j = 0 ; j < numPrivateSelected ; j++) //for j tuple
    {
        System.out.println("");
        for(int i=0;i<numItemsinPrivate[j];i++) //
            for(int t=0 ; t<ProjDB_X[0].length ; t++)

                if(PrivateSet[j][i].equals(Double.parseDouble(Proj
                    DB_X[0][t])))
                {
                    b=false;
                    break;
                }
    }
    return b;
}

```

#### 4. Checking for Non-Discriminative Itemsets:

```

Boolean chk_nonDiscrri(Itemset itemset , Itemset [] P)
{
    Boolean b = false;
    int i = 0 , x = 0 , j = 0;
    int len = itemset.attrname.length;
    int cnt = 0;
    int t = 0;

    String [][] PDB = PDB_X(itemset);
    int numrowPDB = PDB[0].length;
    int numrow = c/2;
    int numcol = 4 - len;
    int [][] countj = new int[numrow][numcol];

    System.out.println("Numrows = " + numrow + "Numcol=" + numcol);

    for(i = 0 ; i<numrow ; i++)
        for(j = 0 ; j<numcol ; j++)
            countj[i][j] = 0;

    for(int k=0; k<P.length;k++) //for every private element
    {
        t=0;
        for(i = 1 ; i<c ; i = i + 2)
        {
            for(j = 1 ; j<=numcol ; j++)
            {
                if(P[k].tuple.equals(PDB[i][0]))
                {
                    if(P[k].attrname[0].equals(PDB[i][j]) &&
                        P[k].value[0].equals(Double.parseDouble(PDB[i-1][j])))
                    {
                        countj[t][j]++;
                    }
                }
            }
        }
    }
}

```





```

}
if(r==t)
    b=true;

return b;
}

```

### 5. *Checking for Discriminative Itemsets:*

```

String [][] chk_Discriminative(Itemset itemset, Itemset [] P,
    double delta)
{
    String [][]PDB = PDB_X(itemset);
    String [][]Pattr = new String[c][2];
    int r = 0 , i = 0 , j = 0 , cnt = 0;
    double []conf = new double[10];
    String [][] disc = new String[20][20];
    int numAI = 0;
    int len = itemset.attrname.length;
    int n = 6-len;

    for(i=0;i<10;i++)
        conf[i]=0;

    for(i=0;i<c;i++)
        for(j=0;j<n;j++)
            if(PDB[i][j].equals("#"))
            {
                Pattr[r][0] = PDB[i][0];
                r++;
            }
    cnt = r;
    r=0;

    for(i=0;i<cnt;i++)
        for(j=0;j<3;j++)
            if(Pattr[i][0].equals(P[j][0]))
                Pattr[i][1]= P[j][1];

    for(i=0;i<cnt;i++)
        Pattr[i][0]=Pattr[i][1];

    for(i=0;i<cnt;i++)
    {
        conf[i] = confidence(X, Pattr[i][1]);
        if(conf[i]==delta)
        {
            disc[numAI] = X;
            numAI++;
        }
    }
    return disc;
}

```

### 7. *Itemset Class:*

```

class Itemset
{
    String[] Attributes = { "class1" , "class2" };
    String [] attrname = new String [100];
    Double [] value = new Double [100];
    double support ;
    String tuple;

    Itemset()
    {
        attrname[0] = "";
        support = 0.0;
        value[0] = 0.0;
    }

    Itemset(String[] itemset)
    {
        attrname = itemset;
        support = support(attrname);
    }

    Itemset(String[] itemset, Double [] val)
    {
        attrname = itemset;
        value = val;
        support = support(attrname);
    }

    Itemset(String [] itemset, String t, Double [] val)
    {
        attrname = itemset;
        tuple = t;
        value = val;
    }

    Itemset(String [] itemset, String t)
    {
        attrname = itemset;
        tuple = t;
    }

    String getClassname(int index)
    {
        return(Attributes[index]);
    }

    int [] getIndex()
    {
        int len = this.attrname.length;
        int [] index = new int[len];

        for(int i=0;i<len;i++)
            index[i] = 0;

        for(int j=0 ; j<len ; j++)
            for(int i=0 ; i<Attributes.length ; i++)
                if(this.attrname[j].equals(Attributes[i]))
                    index[j] = i;
    }
}

```

```

        return index;
    }

    int getIndex(String a)
    {
        int index = 0;
        for(int i=0 ; i<Attributes.length ; i++)
            if(a.equals(Attributes[i]))
                index = i;
        return index;
    }

    Double [] valueof(String tuple)
    {
        int [] index = this.getIndex();
        int len = index.length;
        Double [] value = new Double [len];

        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection
                ("Jdbc:Odbc:Dissertdb");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from Table2
                where TID = '"+tuple+"'");

            rs.next();
            for(int i=0;i<len;i++)
                value[i] = rs.getDouble(i+2);

            con.close();
        }
        catch(Exception exp)
        {
            exp.printStackTrace();
        }
        return value;
    }

    double support(String [] X)
    {
        int len = X.length;
        double sup = 0 , supi = 0 , min = 10000;
        int i = 0;
        String items = X[0];
        Double [] row = new Double [4];

        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection
                ("Jdbc:Odbc:Dissertdb");
            Statement stmt = con.createStatement();

            if(len == 1)
            {
                ResultSet rs = stmt.executeQuery("select " + items + "
                    from Table2");
            }
        }
    }

```

```

        while(rs.next())
            sup += rs.getDouble(items);
    }
    else
    {
        for(i=0;i<len-1;i++)
            items = items + ", " + X[i+1];
        ResultSet rs = stmt.executeQuery("select " + items + "
            from Table2");

        while(rs.next())
        {
            min = 10000;
            for(i=0;i<len;i++)
            {
                row[i] = rs.getDouble(i+1);
            }

            for(i=0;i<len;i++)
                if(min>row[i])
                    min = row[i];

            sup += min;
        }
    }
    sup = sup/(double)5;
    con.close();
}
catch(Exception exp)
{
    exp.printStackTrace();
}
return sup;
}

void dispItemset()
{
    System.out.println("Itemset : count =");

    for(int i=0;i<attrname.length;i++)
        System.out.println(attrname[i]);

    System.out.println(" : " + support);
}
}

```

### 3. Border-based Approach for Hiding Sensitive FWI

#### 1. Positive Border:

```

String [] PositiveBorder()
{
    String [] PBorder = new String [100];
    FreqItemset [] f ;
}

```

```

f = new FreqItemset[17];
int p=0;

for(int i=0;i<17;i++)
    f[i] = new FreqItemset();

int len = X.length;

for(int i=0;i<len;i++)
{
    f[i].Itemset=X[i];
    f[i].sup = Y[i];
}
String []CombItem;

for(int i=0;i<len-1;i++)
{
    cnt =0;
    CombItem = combinations(StrToStrArr(f[i].Itemset));

    for(int j=i+1 ; j<len ; j++)
    {
        for(int k=0;k<NumComb-1; k++)
            if(CombItem[k].equals(f[j].Itemset) && f[i].SupSet)
            {
                f[j].SupSet=false;
                if(cnt==0)
                {
                    PBorder[p] = f[i].Itemset;
                    p++;
                }
                cnt++;
                break;
            }
    }
}
NumPBorders = p;
for(int i=0 ; i<NumPBorders ; i++)
{
    InitSup_PBorder[i] = support(StrToStrArr(PBorder[i]));
}

String [] ABX = AffectedBorder(PBorder, "cd");
double [] W_x = Weights();
int [] ABu = RelevanceBitVector(ABX, 'a');

String [][]W = AffectedBorder_Candidate("ab");

double impact_delu = Impact(ABu, W_x);
return PBorder;
}

```

## 2. *Hiding Candidate:*

```

String [][] Hiding_Candidate(String Itemset)
{
    int len = Itemset.length(), i=0, j=0,p=0,k=0;

```

```

String [][] rsltset = new String [20][20];
int [] index = new int [len];
String[] X = StrToStrArr(Itemset);
String [][] C = new String[NumRows][len+1];

for(i=0;i<len;i++)
    for(j=0;j<9;j++)
        if(X[i].equals(Attr[j]))
            {
                index[p]=j;
                p++;
            }
i=0;
try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection
        ("Jdbc:Odbc:BorderApp");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from Table1");

    while(rs.next())
    {
        for(j=1;j<=9;j++)
            rsltset[i][j-1] = rs.getString(j);
        i++;
    }
    NumRows = i;

    for(i=0;i<NumRows;i++)
    {
        cnt = 0;
        for(j=0;j<p;j++)
            if(rsltset[i][index[j]].equals("1"))
                cnt++;

        if(cnt==len)
        {
            C[k][j] = rsltset[i][0];
            k++;
        }
    }

    con.close();
}

catch(Exception exp)
{
    exp.printStackTrace();
}
return C;
}

```

### 3. *Relevance Bit Vector:*

```

int [] RelevanceBitVector(String []ABX, char c)
{

```

```

int numAffItem = ABX.length;
int [] RelBit = new int[numAffItem];
int lenAB = 0, p = 0;
char [] ABXarr = new char[lenAB];

for(int i=0;i<numAffItem;i++)
    RelBit[i]=0;

for(int i=0;i<numAffItem && ABX[i]!=null;i++)
{
    ABXarr = ABX[i].toCharArray();
    lenAB = ABXarr.length;
    for(int j=0;j<lenAB;j++)
        if(ABXarr[j]==c)
            {
                RelBit[i]=1;
                break;
            }
}
return RelBit;
}

```

#### 4. Impact Calculation:

```

double Impact(int []ABu, double [] W_x)
{
    double impact_delu = 0;
    for(int i=0 ; i<W_x.length ; i++)
        impact_delu = impact_delu + ABu[i]*W_x[i];

return impact_delu;
}

```

#### 5. Negative Border:

```

String [] NegativeBorder()
{
    String []NBorder = new String [100];
    FreqItemset [] f ;
    f = new FreqItemset[17];
    int p=0;

    for(int i=0;i<17;i++)
        f[i] = new FreqItemset();
    int len = X.length;

    for(int i=0;i<len;i++)
    {
        f[i].Itemset=X[i];
        f[i].sup = Y[i];
    }

    String []CombItem;
    String X1,X2;
    int k = 0;
}

```



```

for(int i=0;i<len-1;i++)
{
    cnt =0;
    CombItem = combinations(StrToStrArr(f[i].Itemset));
    for(int j=i+1 ; j<len ; j++)
    {
        X2 = f[j].Itemset;
        for(k=0;k<NumComb-1; k++)
        {
            X1 = CombItem[k];
            if(X1.equals(X2))
            {
                if(cnt==0)
                    NBorder[p] = f[j].Itemset;

                cnt++;
                p++;
                break;
            }
        }
        if(k==NumComb-1)
        {
            NBorder[p] = f[i].Itemset;
            p++;
        }
    }
}
return NBorder;
}

```

## 6. *Weights:*

```

double [] Weights()
{
    double [] wt = new double[NumPBorders];
    double [] CurSup_PBorder = new double[NumPBorders];

    for(int i=0;i<NumAffBorX;i++)
        CurSup_PBorder[i] = InitSup_PBorder[i];

    for(int i=0;i<NumAffBorX;i++)
    {
        if(CurSup_PBorder[i] >= minsup)
            wt[i] = (double)1;
        else if(CurSup_PBorder[i]>=0&&CurSup_PBorder[i]<=minsup)
            wt[i] = lambda + minsup - InitSup_PBorder[i];
    }
    return wt;
}

```

## 7. *Affected Border:*

```

String[] AffectedBorder(String []PBorder, String X)
{

```

```

String [] AffBorX = new String [10];
String [] ArrX = StrToStrArr(X);
String [] combX = combinations(ArrX);
String [] Combi = null;
int len1 = combX.length;
int len2 = 0, p=0;
boolean flag = false;

for(int i=0;i<NumPBorders;i++)
{
    Combi = combinations(StrToStrArr(PBorder[i]));
    len2 = Combi.length;
    flag=false;

    for(int j=0;j<len1;j++)
    {
        if(flag)
            break;

        for(int k=0;k<len2;k++)
            if(combX[j].equals(Combi[k]) && PBorder[i]!=null)
            {
                AffBorX[p] = PBorder[i];
                System.out.println("Aff Bor="+ AffBorX[p]);
                p++;
                flag = true;
                break;
            }
    }
}
NumAffBorX = p;
int x=0;
String [] A = new String [p];
for(int i=0;i<p;i++)
{
    A[x]=AffBorX[i];
    x++;
}
return A;
}
}

```

### 8. *Combinations:*

```

String[] combinations(Itemset itemset)
{
    int n;
    int [] x = new int[20];
    int len = itemset.length;
    int com = (int)Math.pow(2,len);
    NumComb = com;
    String [][] combi = new String[com-1][len];
    char [][] ex = new char[com-1][len];

    for(int j = 0 ; j < com-1 ; j++)
    {
        n = j+1;
        for(int i = 0 ; i<len ; i++)

```

```

    {
        x[i] = n % 2;
        ex[j][i] = Integer.toString(x[i]).charAt(0);
        n = n / 2 ;
    }
}
int r=0;
for(int i=0;i<com-1;i++)
{
    r=0;
    for(int j=0;j<len;j++)
        if(ex[i][j]=='1')
        {
            combi[i][r]=s[j];
            r++;
        }
}
String [] C = new String[com-1];

for(int i=0;i<com-1;i++)
    C[i] = StrArrToStr(combi[i]);
//    System.out.println(C[i]);

return C;
}

```