ABSTRACT

DATA MINING USING NEURAL NETWORKS FOR LARGE CREDIT CARD RECORD SETS

by
Wei Wei

Data mining using neural networks has been applied in various financial fields such as risk mitigation, missing data filling, fraud detection, and customer profile classification etc. This master thesis work aims to develop methodologies to mine large sets of records and in particular to fill missing data in these records. The steps include data cleansing, data selection, data preprocessing, data representation, data clustering and finally the missing data filling. Furthermore, this work designs algorithms to evaluate the supervised neural networks' performance, which is helpful for the future research on data prediction and classification. A case study based on a large data set of credit card records, which contains incomplete records, is performed to demonstrate that the proposed algorithms and their implementations accomplish the task of filling missing data in such records.

DATA MINING USING NEURAL NETWORKS FOR LARGE CREDIT CARD RECORD SETS

by
Wei Wei

A Master's Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Masters of Science in Computer Science

Department of Computer and Information Science

May 2001

APPROVAL PAGE

DATA MINING USING NEURAL NEWTWORKS FOR LARGE CREDIT CARD RECORD SETS

Wei Wei

Dr. Constantine N. Manikopoulos, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Jay Jorgenson, Committee Member — Date
Associate Professor of Mathematics, City University of New York

Dr. MengChu Zhou, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

BIOBRAPHICAL SKETCH

Author:	Wei Wei
Degree:	Master of Science
Date:	May 2001

Undergraduate and Graduate Education:

- Master of Science in Computer Science
  New Jersey Institute of Technology, Newark, NJ, 2001
- Bachelor of Arts in Business
  Sichuan Normal University, Sichuan, P. R. China, 1998
- Bachelor of Engineering in Electrical Engineering
  Chengdu University, Chengdu, P. R. China, 1992

Major:	Computer Science

To my beloved parents and Ying who are always
supporting me whenever time is good or bad.
To my dearest grandparents whose care and kindness warm
my heart and never fade away even time pass by.

ACKNOWLEDGEMENT

I would like to express my appreciation to my advisor, Dr. Constantine N. Manikopoulos, as well as Dr. Jay Jorgenson for their guidance and support through this research.

Special thanks are due to Dr. MengChu Zhou who agreed to serve as my defense committee.

I would also like to thank the support from Mutlti-media Research Center and all of the students in the Multi-media Research Center for their assistance, collaboration and friendship in the past year. They are Zheng Zhang, Jun Li and Yuedong Wu.

TABLE OF CONTENTS

Chapter	Page
1 INTRODUCTION	1
1.1 Background	1
1.1.1 Data Mining for Decision-making	1
1.1.2 Neural Networks: A Data Mining Engine	3
1.2 Motivation	4
1.3 Objectives	5
1.4 Organization	5
2 LITERATURE REVIEW	6
2.1 Data Mining and Knowledge Discovery	6
2.1.1 Issues in Feature Selection	7
2.1.2 Issues in Data Clustering	9
2.1.3 Issues in Association Rule	9
2.2 Artificial Neural Networks	16
2.2.1 Back-Propagation Model	16
2.1.2 Self-Organization Model (SOM)	16
2.1.3 Application in Data Mining	18
2.3 Summary	20
3 A GENERIC FRAMEWORK FOR FILLING MISSING DATA USING NEURAL NETWORK	22
3.1 A Solution to Credit Card Problems Using Neural Network	22
3.2 Data Preparation	23

TABLE OF CONTENTS
(Continued)

Chapter **Page**

3.3 Neural Network Models Selection 28

3.4 Training and Testing Neural Network 29

3.5 Measures of Success 30

3.6 Summary 31

4 IMPLEMENTAIONS USING NEURAL NETWORK TO FILL MISSING DATA FOR CREDIT CARD RECORD SETS 33

4.1 System Architecture 33

4.2 Data Preprocessing 36

4.3 Missing Data Filling 45

4.3.1 Default Value Determination 46

4.3.2 Data Cleansing 47

4.3.3 Data Representation 47

4.3.4 Data Filling Sequence Determination 49

4.3.5 Second Pass Filling Missing Data 49

4.4 Neural Network Performance Evaluation 56

4.5 Summary 63

5 CONCLUSIONS AND FUTURE RESEARCH 64

5.1 Conclusions 64

5.2 Future Research 65

REFERENCES 67

LIST OF TABLES

Table		Page
2.1	The comparison of methodologies for sequential ARM	15
4.1	Attribute valid data percent	38
4.2	Default values	46
4.3	Data filling sequences	51
4.4	Learning set 10000 records	61
4.5	Learning set 5000 records	61
4.6	Learning set 1000 records	61
4.7	Accuracy matrix (RMS = 0.5)	62
4.8	Reverse Accuracy matrix	62

LIST OF FIGURES

Figure		Page
2.1	(a) Bookstore Database; (b) Frequent Item Sets; and (c) Strong Rules	10
2.2	Apriori algorithm runs through bookstore database	11
2.3	DHP algorithm runs through bookstore database	12
2.4	Partition algorithm runs through bookstore database	13
2.5	DIC algorithm runs through bookstore database	14
2.6	MaxClique algorithm runs through bookstore database	15
2.7	A typical back-propagation network model	17
2.8	Self-organization network model	17
4.1	System Architecture	34
4.2	System Architecture (cont.)	35
4.3	Original Sample Data	36
4.4	Formatted Sample Data	37
4.5	Strata Sample	40
4.6	Value patterns Sample	41
4.7	Data Type Percentage for Categorical Attributes	42
4.8	Data Type Percentage for Continuous Attribute (64 bins)	43
4.9	SOM Data Sample	50
4.10	Interpretation Sample SOM Data	50
4.11	Sample Cluster report	57
4.12	Winner Vector Sample	58
4.13	Filled Sample Data	59

LIST OF FIGURES
(Continued)

Figure **Page**

4.14 Back-Propagation Model 60

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Data Mining for Decision-making

The last four decades have experienced a revolution in information technology. The use of computer has evolved from the piecemeal automation of certain business operations, such as accounting and billing, into today's integrated computing environments, which offer end-to-end automation of all major business processes [Bigus, 1996]. One problem with the proliferation of computers throughout the business is the large number of database scattered across systems. Increasingly, people want to leverage their investments in this data, to use it as an aid in decision-making, and to turn it into operational applications. Data mining, more than just complex queries, promises to do just that, which is to extract useful regularities from large data archives, either directly in the form of "knowledge" characterizing the relations between the variables of interest, or indirectly as functions that allow to predict, classify or represent regularities in the distribution of the data [Bengio et al., 2000].

A number of business trends have made the usage of data mining tools and services mandatory for companies vying for business in today's competitive market place [Kleissner, 1998]:

- Data explosion: As companies are confronted with the challenge of handling an ever-increasing amount of data, it is becoming more difficult for business professionals to understand the desired information from this data. Data mining promises to alleviate some of this difficulty.

- Business reengineering and organizational decentralization: Over the past few years, corporations have been reengineering their business processes and organizations. This has resulted in flatter and leaner organizations where knowledge workers have the authority and responsibility to implement and recommend business process optimizations and improvements.
- Faster product cycles: Most companies are challenged by the need for faster product and service development cycles in order to take advantage of newly emerging market opportunities. Data mining provides the means for identifying new product chances and cross-selling products and services into existing customer accounts.
- Globalization and enterprise topologies: The globalization of the economy is in part supported and enabled by enterprise information system topologies where distributed computing is becoming the dominant computing paradigm. Data mining technologies, methodologies, tools, and services need to take advantage of and add new components to this enterprise infrastructure.

Data mining manifests a synergy of a diverse set of computational technologies to glean decision-quality knowledge buried in the enormous stocks of collected data, which consists of three major steps as follows:

(1) Data preparation: select, clean, and preprocess data under the guidance and knowledge of a domain expert.

(2) Data mining algorithm: process the prepared data, compressing and transforming it to make it easy to identify any valuable nuggets of information through a data-mining algorithm.

(3) Data analysis: evaluate the data mining outputs, discover additional domain knowledge, and determine the relative importance of the facts generated by the mining algorithms.

1.1.2 Neural Networks: A Data Mining Engine

Widely diverse arrays of developments and inventions in artificial intelligence have laid the groundwork for artificial neural networks. This new approach to intelligent systems involves constructing computers with architectures and processing capabilities that mimic the processing characteristics of the brain. The results are knowledge representations based on massive parallel processing, fast retrieval of large amounts of information, and the ability to recognize patterns based on experience.

An artificial neural system models, in a very simplified way, the biological systems of the human brain, which performs many of the kinds of tasks that humans do [Bigus, 1996]:

- Classification: group items based on a predefined attribute.
- Clustering: group items based on a previously undefined attribute.
- Association: make correlations between items and individuals, and deduce rules that define relationships.
- Modeling: model relationships through a few exact examples and generalize novel cases or problems.
- Prediction: forecast trends based on current information.

- Constraint satisfaction: solve constraint satisfaction and optimization problems with weighted connection and analog computing.

Neural network technology has significant advantages over conventional rules in some applications [Medsker et al., 1996; Trippi and Turban, 1996]:

(1) Fault tolerance: since there are many processing nodes, each with primarily local connections, damage to a few nodes or links does not bring the system to a halt.

(2) Generalization: when a neural network is presented with noisy, incomplete, or previously unseen input, it generates a reasonable response.

(3) Adaptability: since the network learns in new environments, training can occur continuously over its useful life and occur concurrently with the deployment of the network.

1.2 Motivation

As shown in Section 1.1, data mining is a powerful tool for extracting useful information from tons of data [Thuraisingham, 2000]. Then, the extremely large amounts of data are compressed to reveal the inner relationships among the data elements. It is clear that "large collection of data" is the central issue of data mining. Moreover, just collecting the data in one place and making it easily available isn't enough. When operational data from transactions is loaded into the data warehouse, it may contain missing or inaccurate data. To overcome this problem, the operational data must go through a "cleansing" process, which takes care of missing or out-of-range values. Furthermore, how to predict such kind of data based on the existing complete or incomplete raw data set becomes extremely crucial.

1.3 Objective

The goal of this research is to provide a neural-network-based data mining methodology to fill missing data for credit card record sets. Specific objectives are to:

(1) Analyze data set includes: data distribution pattern and data records patterns, etc.

(2) Develop methodologies for data cleansing, data selection, and data preprocessing and data representation.

(3) Design algorithms to accurately fill missing data.

(4) Implement the above concepts and methodologies into software applications.

1.4 Organization

This thesis is organized as follows. Chapter 2 makes a literature review for the current research issues in data mining and neural networks. A generic framework for data mining by using neural network is presented in Chapter 3. Implementation details and final results are addressed in Chapter 4. Finally, Chapter 5 gives the conclusions and future research directions.

CHAPTER 2

LITERATURE REVIEW

As mentioned in Chapter 1, data mining, referred to as knowledge discovery, has become a buzzword in business circles [Frawley et al., 1992]. Significant research efforts have been carried out in industries and academia addressing various problems in these areas. This chapter reviews the major issues in these fields.

2.1 Data Mining and Knowledge Discovery

Data mining (DM) is a folkloric denomination of a complex activity that aims at extracting synthesized and previously unknown information from large data set. DM is sometimes considered as just a step in a broader overall process called knowledge discovery in data set [Olaru et al., 1996]. The tasks of DM are very diverse and distinct because many patterns exist in a large data set. Based on the patterns that are being looked for, tasks in DM can be classified as follows [Fu, 1997]:

(1) Summarization: produce compact and characteristic descriptions for a given set of data.

(2) Classification: derive a function or model to determine the class of an object based on its attributes.

(3) Clustering: identify clusters of similar objects sharing a number of interesting properties.

(4) Association: discover the connection of objects.

(5) Trend analysis: identify patterns in an object's evolution and match objects' changing trends.

2.1.1 Issues in Feature Selection

The classification task in DM is to discover some kind of relationship between the input attributes and the output class, so that the discovered knowledge can be used to predict the class of a new unknown object. One of the first stages of the classification process is the Feature Selection (FS), by means of which the complexity of the problem is reduced by the elimination of irrelevant features to consider later in the classification stage [Martin-Bautista and Vila, 1999].

The first experiments about FS have been realized in machine learning, and most of the methods of classification and reduction of the features set have been proposed [Dash and Liu, 1997]. These methods may be classified into two categories [Langley, 1994]. One is filter method, which is applied before the classification, and its evaluation does not depend on the classification but usually based on different measures of distance between the examples. The other is wrapper method, which is applied while the classification or rather their goodness is based on the result of the classification by a certain method. However, the wrapper methods have some drawbacks as follows:

- In the mining context, the number of features with which people have to deal is quite large. Thus, the complexity and the execution time for the FS process make it unfeasible.
- As the feature selection depends on the classification algorithm, the generality is lost because of the behavior of the classification algorithm in terms of accuracy and

efficiency. Moreover, the selection of a classification algorithm slightly suitable for a certain problem may give a rise to choose features wrongly.

- It is very tough to combine the wrapper methods with soft computing techniques, especially when the set of features are large.

The Generic Algorithm (GA) has been proved as a powerful tool to optimize the process of classification, especially when the domain knowledge is costly to exploit or unavailable, because of its relative insensitivity to noise and the requirement of no domain knowledge [Vafaie and De Jong, 1992]. Furthermore, standard rule induction systems like AQ15 and the standard implementation of ID3, C4.5 [Vafaie and De Jong, 1992, 1995] have been utilized as classifiers to evaluate error rate when the feature set is reduced. An approach combining Gas and the K-NN algorithm to find out for an optimal feature weighting to determine the relative importance of each feature are presented in [Kelly and Davis, 1991; Punch et al., 1993]. A complete different method to this problem is addressed, in which a GA is combined with a production decision rule system [Pei et al., 1997]. Yang and Honavar (1998) studied the feasibility of FS in combination with GA to design neural networks for pattern classification. The main advantage of using GA is akin to the advantage of natural selection in the sense that it is a solid technique that finds a solution if one exists [Kleissner, 1998]. A survey of generic FS in mining issues is presented in [Martin-Bautista and Vila, 1999]. The large over-production of solutions on the other hand requires a lot of computing power and an expert is required to code the problem as well as the artificial environment necessary for the evolutionary game.

2.1.2 Issues in Data Clustering

Clustering studies identify the kind of data that trends to occur together with other data. Data within a cluster have similarities, but different significantly from other data outside the cluster. Artificial neural networks (ANN), as a non-symbolic, inductive paradigm, have been used to perform data clustering in data mining [Craven and Shavlik, 1997].

In the ANN paradigm, typically, unsupervised learning based SOM are used for data clustering, due to their natural propensity to (a) find similarities amongst data items and (b) to group similar data items in proximity [Kohonen, 1982, 1990]. However, SOM may lead to undesirable ambiguity about a cluster's structure and membership when the output of the SOM does not exhibit distinct clusters, rather the boundaries of the emergent 'implied' cluster are rather vague. To solve these problems, Abidi and Ong (2000) proposed a strategy to determine the boundaries of data cluster derived from SOM type ANN. The general ides is to give a high-level topological ordering by a SOM, feed the SOM-driven clustering information to a K-Means algorithm to refine the SOM's output by way of demarcating the output layer of the SOM into distinct clusters. A clustering genetic algorithm designed for rule extraction from supervised neural networks was presented in [Hruschka and Ebecken, 2000].

2.1.3 Issues in Association Rule

Association rule mining (ARM) has become one of the core data mining tasks because of its inception, which includes parallel ARM and distributed ARM. This work uses a bookstore database shown in Fig. 2.1 as an example to illustrate each ARM algorithm.

Items

Jane Austen	A
Agatha Christie	C
Sir Arthur Conan Doyle	D
Mark Twain	T
P. G. Wodehouse	W

Database

Transaction	Items
1	ACTW
2	CDW
3	ACTW
4	ACDW
5	ACDTW
6	CDT

(a)

Frequent Itemsets (min_sup =50%)

Support	Itemsets
100% (6)	C
83% (5)	W, CW
67% (4)	A, T, D, AC, AW, CD, CT, ACW
50% (3)	AT, DW, TW, ACT, ATW, CDW, CTW, ACTW

Maximal frequent itemsets: CDW, ACTW

(b)

Association rules (min_conf=100%)

A->C (4/4)	AC->W (4/4)	TW->C (3/3)
A->W (4/4)	AT->C (3/3)	AT->CW (3/3)
A->CW (4/4)	AT->W (3/3)	TW->AC (3/3)
D->C (4/4)	AW->C (4/4)	ACT->W (3/3)
T->C (4/4)	DW->C (3/3)	ATW->C (3/3)
W->(5/5)	TW->A (3/3)	CTW->A (3/3)

(c)

Figure 2.1 (a) Bookstore Database; (b) Frequent Item Sets; and (c) Strong Rules

A. Apriori

The Apriori algorithm has merged as one of the best ARM algorithms, which uses a complete, bottom-up search with a horizontal layout and enumerates all frequent itemsets

[Fayyad, et al., 1996]. Using the database example in Fig. 2.1, the iterative algorithm can be understood in depth and shown in Fig. 2.2:

(1) Generate candidates of length k from the frequent (k-1) length itemsets. For example, for F_2 = {AC, AT, AW, CD, CT, CW, DW, TW}, C_3 is obtained as C_3 = {ACT, ACW, ATW, CDT, CDW, CTW}.

(2) Prune any candidate that has at least one infrequent subset. For example, CDT will be pruned because DT is not frequent.

(3) Scan all transactions to obtain candidate supports. Apriori stores the candidates in a hash tree for fast support counting. In a hash tree, itemsets are stored in the leaves; internal nodes contain hash tables to direct the search for a candidate.

Scan 1

A	4
C	6
D	4
T	4
W	5

Scan 2

AC	4
~~AD~~	~~2~~
AT	3
AW	4
CD	4
CT	4
CW	5
~~DT~~	~~2~~
DW	3
TW	3

Scan 3

ACT	3
ACW	4
ATW	3
CDW	3
CTW	3

Scan 4

ACTW	3

Figure 2.2 Apriori algorithm runs through bookstore database

B. Dynamic Hashing and Pruning (DHP)

Park et al. (1995) proposed the DHP algorithm, which is an extension of the Apriori approach by using a hash table to pre-compute approximate support of 2-itemsets during the first iteration. The second iteration need count only those candidates falling in hash cells with minimum support. The whole steps through the example data shown in Fig. 2.1 are presented in Fig. 2.3.

Hash table (built in Scan 1)

3	7	2	6	8	7
DW	AC	AD	AT	AW	CW
	TW		CD	CT	CT

Figure 2.3 DHP algorithm runs through bookstore database

C. Partition

Savasere et al. (1995) proposed the two-pass partition algorithm, which logically divides the horizontal database into non-overlapping partitions. This algorithm includes the following main steps:

(1) Read each partition, and form vertical tidlists for each item.

(2) Generate all locally frequent itemsets through tidlist intersections.

(3) Merge locally frequent itemsets to form a global candidate set.

(4) Pass all partitions and obtain all candidates' global counts.

Fig. 2.4 shows how partition algorithm works on bookstore database in Fig. 2.1.

D. Dynamic Itemset Counting (DIC)

Brin et al. (1997) proposed the DIC algorithm, which is a generalization of Apriori. The database is divided into p equal-sized partitions so that each partition fits in memory. For partition 1, DIC gathers the supports of single items. Items found to be locally frequent generate candidate 2-itemsets. Then DIC reads partition 2 and obtains support for all current candidates. This process repeats for the remaining partitions. DIC is effective in reducing the number of database scans if most partitions are homogeneous. If data is not homogeneous, DIC might generate many false positives and scan the database more than Apriori does. Fig. 2.5 shows how DIC works on the example database in Fig. 2.1.

Partition

tid: 1-3

A	2
C	3
D	1
T	2
W	3

AC	2
~~AD~~	~~0~~
AT	2
AW	2
CD	2
CT	1
CW	2
~~DT~~	~~0~~
DW	1
TW	2

ACT	2
ACW	2
ATW	2
CDW	1
CTW	2

ACTW	1

tid: 4-6

A	2
C	3
D	3
T	2
W	2

AC	2
AD	2
AT	1
AW	2
CD	3
CT	2
CW	2
DT	1
DW	2
TW	1

ACDT	1
ACDW	1
ACTW	2
ADTW	1
CDTW	1

ACDTW	1

Scan 2 (tid: 1-6)

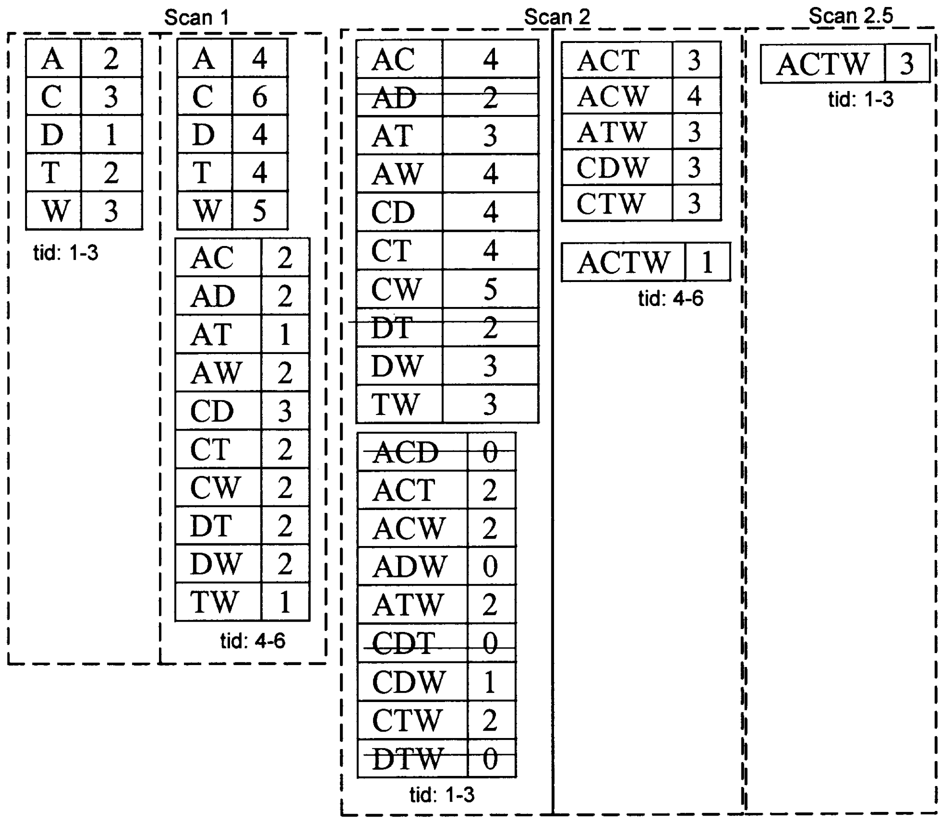
A	4
C	6
D	4
T	4
W	5

AC	4
~~AD~~	~~2~~
AT	3
AW	4
CD	4
CT	4
CW	5
~~DT~~	~~2~~
DW	3
TW	3

ACT	3
ACW	4
ATW	3
CDW	3
CTW	3

ACTW	3

Figure 2.4 Partition algorithm runs through bookstore database



A	2
C	3
D	1
T	2
W	3

A	4
C	6
D	4
T	4
W	5

AC	2
AD	2
AT	1
AW	2
CD	3
CT	2
CW	2
DT	2
DW	2
TW	1

AC	4
~~AD~~	~~2~~
AT	3
AW	4
CD	4
CT	4
CW	5
~~DT~~	~~2~~
DW	3
TW	3

~~ACD~~	~~0~~
ACT	2
ACW	2
ADW	0
ATW	2
~~CDT~~	~~0~~
CDW	1
CTW	2
~~DTW~~	~~0~~

ACT	3
ACW	4
ATW	3
CDW	3
CTW	3

ACTW	1

ACTW	3

Figure 2.5 DIC algorithm runs through bookstore database

E. MaxClique

MaxClique is an equivalence class-based algorithm, which uses a vertical database format and hybrid search, and generates a mix of maximal and non-maximal frequent itemsets [Zaki, *et al.*, 1997]. The use of a vertical database format makes it possible to determine the support of any k-itemset by simply intersecting the tidlists of the lexicographically first two (k-1)-length subsets that share a common prefix. Furthermore, the method breaks the large search space into small, independent, manageable chunks. MaxClique outperforms Apriori and Partition by more than an order of magnitude. Using the example database shown in Fig. 2.1, the algorithm can be understood as follows, Fig. 2.6 gives the detailed information how it works:

(1) Build an association graph from the frequent 2-itemsets;

(2) Find the maximal cliques in the graph (ACTW, CDW); and

(3) Use hybrid search to process these two classes.

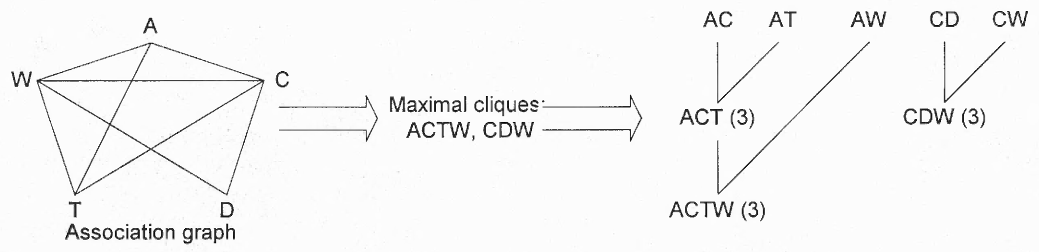


Figure 2.6 MaxClique algorithm runs through bookstore database

Through analysis of these four methodologies, the comparison results for their advantages and disadvantages are concluded in Table 2.1, where K denotes the size of the longest frequent itemset and C_2 array optimization uses a 2D array to count candidate 2-itemsets rather than using hash tree or prefix trees [Zaki et al., 1999].

Table 2.1 The comparison of methodologies for sequential ARM

Algorithm	Datebase Layout	Data Structure	Search	Enumeration	Optimization	# Of Scans
Apriori	Horizontal	Hash tree	Bottom-up	All	C_2 array	K
DHP	Horizontal	Hash tree	Bottom-up	All	C_2 array and hash table	K
Partition	Vertical	None	Bottom-up	All	C_2 array and partitioning	2
DIC	Horizontal	Prefix tree	Bottom-up	All	Count multiple lengths per scan	$\leq$K
MaxClique	Vertical	None	Hybrid	Maximal and non-maximal	C_2 array and clique classes	$\geq$3

2.2 Artificial Neural Networks

In the past few years, neural networks have received a great deal of attention and are being touted as one of the greatest computational tools ever developed [Eberhart and Dobbins, 1990]. Neural networks have been modeled according to the human brain. They consist of a set of nodes (modeled after neurons), which are connected to each other (like neurons are connected to each other by synapses). Typically, a neural network consists of a set of input nodes that receive input signals, a set of output nodes, which give the output signals, and a number of intermediate layers, which connect input and output nodes.

2.2.1 Back-Propagation Model

The back-propagation network model always has an input layer, an output layer and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there will be one or two. Each layer is connected to the succeeding layer. The arrows indicate flow of information during recall. Fig. 2.7 is a typical back-propagation network model. During learning, information is also propagated back through the network and used to update the connection weights. The network can either hetero-associative or auto-associative.

2.2.2 Self-Organization Model (SOM)

Teuvo Kohonen first developed the self-organization map in 1979 and 1982, which was used to visualize topologies and hierarchical structures of higher dimensional input spaces [Kohonen, 1988]. The SOM typically has two layers. The input layer is fully connected to a two-dimensional Kohonen layer. In the SOM layer, none of the process elements (PEs) are connected to each other, regardless of relative position. Fig. 2.8 shows the self-organization network model. The key difference between the SOM and other networks is that the SOM learns without supervision.

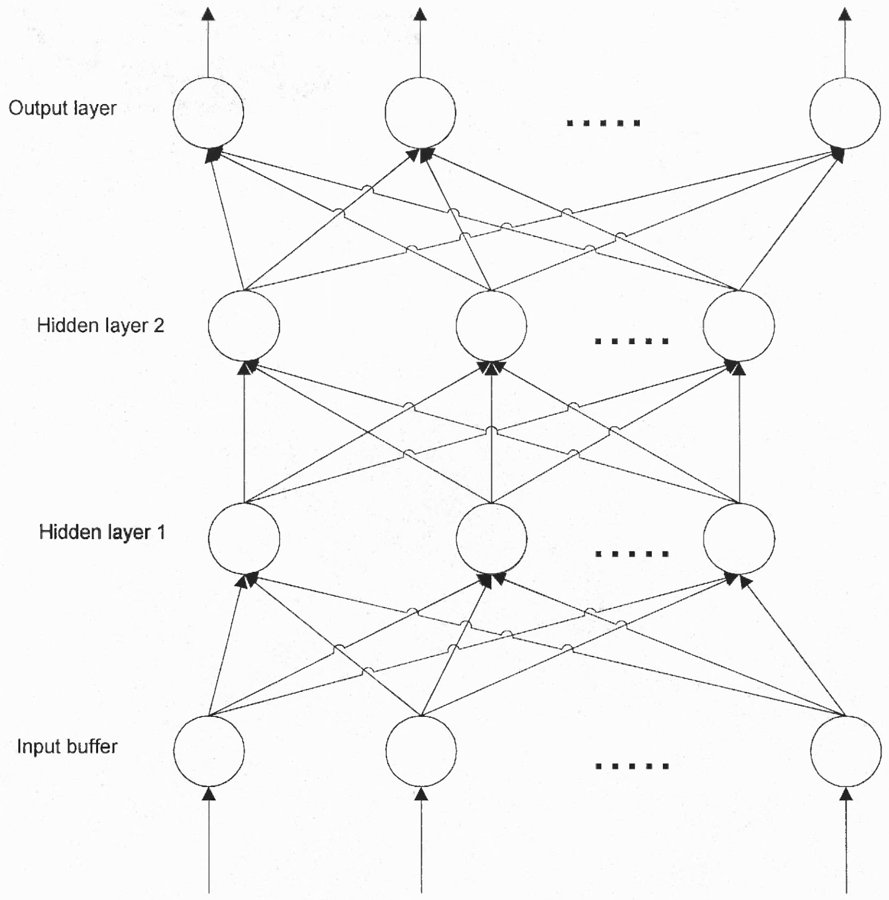


Figure 2.7 A typical back-propagation network model

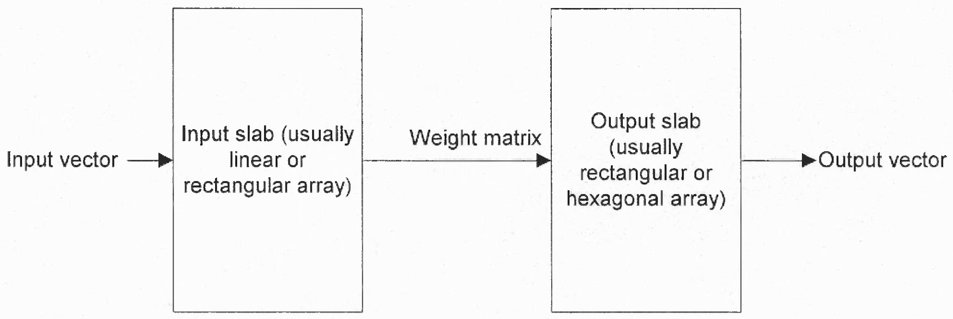

Input vector
Input slab (usually linear or rectangular array)
Weight matrix
Output slab (usually rectangular or hexagonal array)
Output vector

Figure 2.8 Self-organization network model

2.2.3 Applications in Data Mining

Neural networks have the capability of discovering nonlinear, non-obvious, and potentially useful information and knowledge from database [Frayman and Wang, 1998]. For a given set of input-output data in a database, a neural network can be trained using the given data set, then the trained neural network with adjusted numerical weights has learned potential mapping knowledge between input and output data, and finally the neural network is able to apply the discovered knowledge to predict new output data for new input data [Zhang et al., 2000]. Because of strong nonlinear modeling ability of neural network, it becomes a useful tool for data mining. Zhang et al. (2000) proposed a neural-network-based knowledge discovery and data mining methodology based on granular computing, neural computing, fuzzy computing, linguistic computing, and pattern recognition.

When modeling the joint distribution of many random variables with complex interactions, the famous "curse of dimensionality" yields models with exponentially too may parameters. Bengio and Bengio (2000) proposed a method to solve this problem, which is based on a neural network representation of the join distribution of many variables, shows statistically significant improvements with respect to older methods for benchmark data sets of high-dimensional discrete data. Another approach, based on learning maximum entropy models, was proposed as a solution to the same problem, which is very popular in the area of statistical language modeling and information retrieval [Yan and Miller, 2000]. Using the network sensitivity analysis, Kewley et al. (2000) presented a methodology for variable selection. Considering the unequal importance of inputs, Shin et al. (2000) proposed a method, in which ANNs are used to provide weights to the different input features to be used in a K-nearest-neighbors algorithm.

One of the big challenges of data mining is the organization and retrieval of documents from document archives [Bengio et al., 2000]. The advantages of SOM network model have caused much attention in this field. Several researchers have described such new neural architectures, both in the supervised and unsupervised paradigms. Recent works on supervised models have been reported in [Halgamuge and Glesner, 1995; Halgamuge, 1997]. Several extensive reviews on the supervised self-generating neural architectures have done in [Quinlan, 1998; Ash and Cottrell, 1994].

Martinetz and Shulten (1991) developed the Neural Gas algorithm, which can be categorized as an unsupervised self-generating neural network. The network starts with no connections and a fixed number of units floating in the input vector space. When the inputs are presented to the network, units are adapted and connections are created between the winning units and the closest competitor. However, this algorithm uses a fixed number of units, which have to be decided prior to training.

Based on the SOM, Growing Cell Structures algorithm was proposed in [Fritzke, 1991]. A network of nodes whose connectivity defines a system of triangles has replaced the basic two-dimensional grid of the SOM. This algorithm uses a drawing method, which works well with relatively low-dimensional data, but the mapping cannot be guaranteed to be planar for high-dimensional data.

Using the basic concepts of self-organization as the SOM, the Growing Self-organization Map (GSOM) algorithm was developed with a dynamic structure that is generated during the training processes itself [Alahakoon et al., 2000]. The main difference between the two methods is that the SOM attempts to fit in a data set into a predefined structure by self-organizing its node weights as well as possible within its fixed borders. With the GSOM, the borders of the network are expandable and as such the data set can generate new nodes with a flowing out effect, expanding the network outwards. Therefore, the different groupings in the data generate regions for themselves in the network. The self-organization of the weights in the already generated nodes continues at the same time to fine-tune the weights to represent the data better.

Wang et al. (2000) described a new hierarchical visualization algorithm to reduce the dimension of data representation for visualization, which allows the complete dataset to be visualized at the top level with clusters and sub-clusters of data points visualized at lower levels using principal component neural network. Toward similar objectives, a neural network technique to combine SOMs and the nonlinear mappings was proposed in [K   nig, 2000].

2.3 Summary

Data mining, the process of discovering hidden and potentially useful information from very large databases, has been recognized as one of the most promising research topics in the 1990s [Sung et al., 1996]. Classification is one of data mining problems receiving great attention recently in the database community. Various classification algorithms have been designed to tackle the problem by researchers in different fields, such as mathematical programming, machine learning, neural networks and statistics [Lu, et al., 1996]. To choose an association rule for the generation of large items that are presented in at least minimal support of total database tuples is another essential problem faced in the mining. A survey of association rule mining algorithms was presented in [Zaki, 1999]. Besides the popularly studied classification and association, there are many other kinds of data mining tasks to be explored, such as clustering, predictive modeling, time-related pattern analysis etc. A number of directions that may require more in-depth research in data mining were addressed in [Han, 1997].

Artificial neural networks are recognized in the automatic learning framework as universal approximators, with massively parallel computing character and good generalization capabilities, but also as black boxes due to the difficulty to obtain insight into the relationship learned. Recently, applications of neural networks have been increasing in data mining. More and more development tools have emerged on the market. Two kinds of neural network models seem promising for the task of finding data in database, which are SOM and Back-Propagation networks with their corresponding advantages and disadvantages [Stebbins, 1999]. SOM is best used for categorization tasks, because it is self-organization, and resulting categorizations are not necessarily those that would be considered useful. Backpropagation networks are good for prediction, because the designer specifies the structures' of input and output. A historical discussion and a review of important applications of neural networks were presented in [Hecht-Nielsen, 1990].

CHAPTER 3

A GENERIC FRAMEWORK FOR FILLING MISSING DATA USING NEURAL NETWORK

As mentioned in Chapter 2, data mining, the idea of extracting valuable information from data, is widely used, especially in financial field. Financial problems of managerial decisions can be roughly classified into two categories: structured and unstructured [Simon, 1960]. Artificial neural networks are best applied to problem environments that are highly unstructured, require some form of pattern recognition, and may involve incomplete or corrupted data [Trippi and Turban, 1996]. This chapter generally introduces the framework for filling missing data in credit card record sets using neural network.

3.1 A Solution to Credit Card Problems Using Neural Network

The task of approving customers for credit, assigning credit limits and detecting credit fraud is a labor-intensive and time-consuming process that has significant impact on the profitability of most companies [Trippi and Turban, 1996]. How to relieve labors from labor-intensive task and how to find efficient solutions to reduce time complexity become a very challenging and promising field. Furthermore, the process needs to deal with an extremely large amount of data that may be incomplete or inaccurate. How to handle such high-dimensional data is very crucial. An artificial neural network has demonstrated its usefulness in the analysis of such data sets with a distinctive new flavor, which deals with the diversity of input information without requiring that the information be restated in a standard form. It can be trained using customer data as the input vector and the actual decisions of the credit analyst as the desired output vector. This work uses neural network to deal with credit card record sets and develops innovative algorithms to fill missing data. The important steps are presented in the following sections.

3.2 Data Preparation

This is the first step in data mining process. In most cases, the data used for a data mining operation has been just sitting around collecting dust [Bigus, 1996]. There are three important issues need to be concerned before mining these raw material:

- Are these data clean?
- Are these data reliable?
- Are these data sufficient?

A. Data Cleansing

It is often the true fact that not all operational transactions are correct. They might contain inaccurate values, missing data, or other inconsistencies in the data set. Several techniques are being used to clean data. These include rule-based techniques for detecting inaccurate or inconsistent data, which evaluate each data item against metaknowledge (knowledge about the data) about the range of data expected in that field and constraints or relationship to other fields in the record [Simoudis et al., 1995]; Visualization can also be used to easily identify erroneous and out-of-range data. Another way is to use statistical information to replace missing or incorrect field values with neutral, valid values. Data cleansing in this work focuses on filling missing data rather than detecting inaccurate data. Hence, We carefully develop algorithms to determine the default value for each attribute. The work first statistically analyzes each attribute in the data set and gets the probability distributions for all attributes. Then, designs algorithms to determine the default values for all attributes and replaces all missing data with these default values. The details are presented in Chapter 4.

B. Data Selection

Data selection concerns two central issues: one is how to determine the relative importance of each attribute. The other is how to categorize all attributes. A data set may have M attributes, which have the different contribution to the decision-making. How to determine the importance of attributes falls into two ways. Sometimes, experts can do this manually. However, this is a kind of case-by-case solution. Based on statistical analysis, another method can determine the attribute importance by comparing the similarity between target attributes and the uniform distribution functions. If the distribution of an attribute is very similar to the uniform distribution, this attribute has the less importance. The key point is if the distribution of an attribute is uniform, each value has the same probability to appear in a dataset, which makes it useless to predict missing data.

Since some attributes in a data set may have ambiguous value, attribute classification is another important issue in data selection. For instance, value "19" can be either interpreted as categorical data like department number, discrete number value like age or regarded as continuous numeric data. This thesis classifies all attributes into two categories: continuous and categorical data and develops an algorithm to solve this problem based on attributes' output types.

C. Data Preprocessing

Data preprocessing is the step when the clean data, which have been selected, is enhanced. Sometimes this enhancement involves generating new data items from one or more fields, sometimes it means replacing several fields with a single field that contains more information, and sometimes the data needs to be transformed into a form that is acceptable as input to a specific data mining algorithm, such as a neural network. There are several techniques listed below, which are mostly used in the data preprocessing phase:

- Computed attributes: A common requirement is to combine two or more attributes into a new attribute. This is usually in the form of a ratio of each combined value, the sum, the product or other values.
- Scaling: Another transformation involves the more general issues of scaling data feeding to the neural network. Normally, most neural network models adapt Sigmoid (Sig), Hyperbolic Tangent (TanH) or their variants as transfer functions. Equations are:

$$\text{Sigmoid: } f(z) = \frac{1}{1+e^{-z}} \tag{3.1}$$

$$\text{Hyperbolic Tangent: } f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.2}$$

The Sigmoid transfer function requires the input range [0,1] while the Hyperbolic Tangent requires the input range [-1, +1] to avoid “saturation” effect. Thus to scale data into the proper range requires another necessary data preparation step.

In this work, all neural network models such as Back-Propagation (BP), Self-Organization-Map (SOM), etc. use the TanH transfer function; Thus, the original data are scaled into the [-1, +1] range.

- Normalization: Vectors or arrays of numeric data can sometimes be treated as groups of numbers. In these cases, it is necessary to normalize the vectors as a group. There are several ways to handle this. The most common vector normalization method is to sum the squares of each element, take the square root of the sum, and then divide each element by its norm. Another way to normalize vector data is to simply sum up all of the elements in the vector and divide each number by the sum. In this approach, the normalized elements sum to 1.0, and each has a value representing the percentage of contribution they make. The third way is to divide each vector element by the maximum value in the array.

Considering attributes with categorical and numeric data, this work first divides attributes into two categories: continuous data and categorical data by their nature. For continuous data, the normalization equation is introduced as follows:

$$v_{ij}^{'} = \frac{v_{ij} - \mu_i}{\delta_i} \tag{3.3}$$

Where: $v_{ij}^{'}$: Scaled value of the i^{th} attribute in the j^{th} record;

v_{ij} : Unscaled value of the i^{th} attribute in the j^{th} record;

μ_i: Mean value of the i^{th} attribute; and

δ_i : Standard deviation of the i^{th} attribute.

D. Data Representations

Neural networks explore lots of categorical data to do data mining. For categorical data the challenge of representation is to present these variable values in such a way that the network can discern the differences between values and tell the relative magnitude of the differences if that information is available. Various coded data types are used to represent these values. One-of-N code has a length equal to the number of discrete categories allowed for the variable, where every element in the code vector is a 0, except for the single element, which represents the code value. Binary code assigns each category a value from 1 to N and represents by a string of binary digits. Thermometer code is better used when the discrete values are related in some way, usually by increasing or decreasing values. For example, to represent morning, noon and night by 3 bits, noon, morning and night can be represented as [0.5, 1, 0.5], [1, 0.5, 0] and [0, 0.5, 1], respectively. This thermometer encoding scheme not only reveals the difference between categories, but also shows the relationship between each other: Noon is next to morning and night, morning next to noon but far from night etc.

This work designs a different encoding scheme to represent categorical data. The detailed procedures are:

(1) Analyze data types in each attribute and calculate each type's percentage.

(2) Divide all data types into 3 parts, the left and right contain data types which percentage is the lowest and middle part contains the highest percentage data types.

(3) Subgroup all categories into 3 groups, which are encoded as [-0.355, -0.355, +0.355], [+0.355, -0.355, -0.355], and [-0.355, +0.355, -0.355], respectively. The

reason to choose value +/-0.355 is to make sure any pair has the Euclidian distance 1, in other words, to normalize the distance of each pair.

3.3 Neural Network Models Selection

There are many different types of neural network models, which can be categorized by the basic learning paradigm or the approach they use. Supervised neural network is the most common training paradigm, which makes predictions or classifications for a given problem or case. At this point, the "desired" results act as a supervisor to indicate what the answer should be and how to update network itself to achieve this goal. The unsupervised model is often used for clustering and segmentation in data mining, where neural network doesn't have "standard answer", and the training goal is to group similar characteristic vectors together.

In this work, the Back-Propagation Neural Network model is chosen to predict the target data, because the nature of prediction task is supervised, which means there already have some sample cases made by experts in a corresponding field during learning phase. Back-Propagation models are just to learn how to deal with these cases, and simulate an expert to predict target data when a new case without "answer" feeding into a network.

To fill missing data, Self-Organization-Map (SOM) model is selected to cluster all records into some subgroups, where data have the similar characteristics. The key point is if the missing data belong to a subgroup, it is very possible for their values to fall into the value range determined by the subgroup's complete data records. The detailed implementation is presented in Chapter 4.

3.4 Training and Testing Neural Network

Once the data preparation is completed and the neural network model and architecture have been selected, the next step is to train the neural network that includes the following two steps:

(1) Separate training and testing data sets: For most supervised neural network models, networks begin the training process with the connection weights initialized with small random values. The training control parameters are set and the training data patterns are presented to the neural network one after the other. As training progresses, the connection weights are adjusted, and the performance of the network is monitored. In order to evaluate the performance or to determine whether the target model succeed in data prediction or not, implementation needs to test data set, where all weights adjusted at training phase are fixed. This work splits the whole data set into testing and training parts and set the ratio at 1:3.

(2) Set important neural network parameters: To obtain the best results, the critical neural network parameters should be set properly. They are:

- Learn rate: Control the step size for weight adjustments. Decrease over time for some types of neural network. Learn rate is very important because if value is too large, the error factor drop very fast but the side effect is easily falling into "unstable" state. The whole model could never meet the convergence criteria in worst case. On the contrary, whole network will gradually go into a "stable" state but take too much time. It is a trade-off. The proper value based on experience and practice. In the project BP and SOM both need to set this value.

- Momentum: Smooth the effects of weight adjustments over time. BP model need to set this value.
- Error tolerance: Specify how close the output value must be to the desired value before the error is considered to be zero. In BP, the probe using to determine error tolerance usually is Mean Square Root (RMS).
- Activation function and Learning Rule: Select the activation function (transfer function), which is used by the neural processing unit. Most common is the sigmoid or logistic activation function. As mentioned before, in order to get better performance, the project always extend input value range from [-1, +1], so for BP, the activation functions usually used are variants of TanH, learning rules are Delta-Bar-Delta, Ext DBD and Norm-Cum-Delta. For SOM, the functions are logistic or Euclidian.
- Neighborhood: Defines the size or area of units surrounding the winner, which get their weights updated. Neighborhood decreases over time. SOM need to specify this value.
- Number of epochs: determines the number of passes for networks that train for a fixed number of passes through the training data. BP and SOM both need to set this parameter.

3.5 Measures of Success

Once a neural network model is selected and trained with data, it is very important to determine whether the model is well trained or not. The criteria for the measure of the success is different in the following steps:

- Classification: The measure of success in a classification problem is the accuracy of the classifiers, usually defined as the percentage of correct classifications. The algorithms to determine correctness are various. In this work, there are two evaluators. One determines the correctness based on whether the difference between a prediction value and desired value's RMS is lower than a specific threshold or not. The other tool is more sophisticated by using a classification matrix and a reverse matrix. Detailed information is explained in Chapter 4.
- Clustering: Since clustering is an unsupervised usage, where there are no "standard" answers to verify a model. In most cases, the training regimen is determined simply by the number of times data is presented to a network, and how fast the learning rate and the neighborhood decay. A network will be trained for the certain number of epochs specified by a user and then stop.

3.6 Summary

Data mining using Neural Network includes the following four steps:

(1) Data Preparation: it involves data selection, data cleansing, data preprocessing, and data representation. Data selection is a kind of domain knowledge. The domain expert uses his knowledge of a problem and available data to determine attributes' importance and classification. This work classifies all attributes into two categories: categorical and continuous data. Data cleansing in our work focuses on filling missing data. The only purpose of filling missing data in data cleansing step is to gain records vectors, in which every element has a valid value and "erroneous" effects caused by missing values are minimized. In the data preprocessing phase,

this work scales and normalizes continuous data, and transforms categorical data into a numeric normalization form using a specific encoding scheme.

(2) Neural Network Selection: Neural network models are selected based on required tasks. For data prediction purpose, the supervised architecture model Back-Propagation Network is selected. To fill missing data, the unsupervised Self-Organization-Map network is chosen.

(3) Training and testing phase: lists the most important parameters, which directly affect the overall performance of neural network, and explains what these parameters are for.

(4) Evaluation of results from neural network: this work not only uses a common tool such as RMS, but also designs two evaluators: an accuracy matrix and a reverse matrix.

CHAPTER 4

IMPLEMENTAIONS USING NEURAL NETWORK TO FILL MISSING DATA FOR CREDIT CARD RECORD SETS

Chapter 3 generally introduces the background of the work, the given conditions and the targets. Also, the framework of using neural network on data mining is theoretically presented. However, data mining is not a one-hit-all-done process. On the contrary, it is an art. This chapter explores the real implementations of this framework in details.

4.1 System Architecture

This system has the following software packages, which run on Unix and MS Window Operating Systems:

- CREDITCARDPROJ: main software application designed for this work, which is developed using VC++ and deals with data preprocessing, missing data filling and neural network output analyzing etc.; and
- NeuralWare Professional II/Plus: a commercial software, which is mainly used to build Back-Propagation neural network model for data prediction.

The CREDITCARDPROJ includes four modules shown in Figures 4.1 and 4.2:

- Data Preprocessing Model;
- Missing Data Filling Model;
- Neural Network Performance Evaluation Model; and
- Auxiliary Mathematics Tool Module.

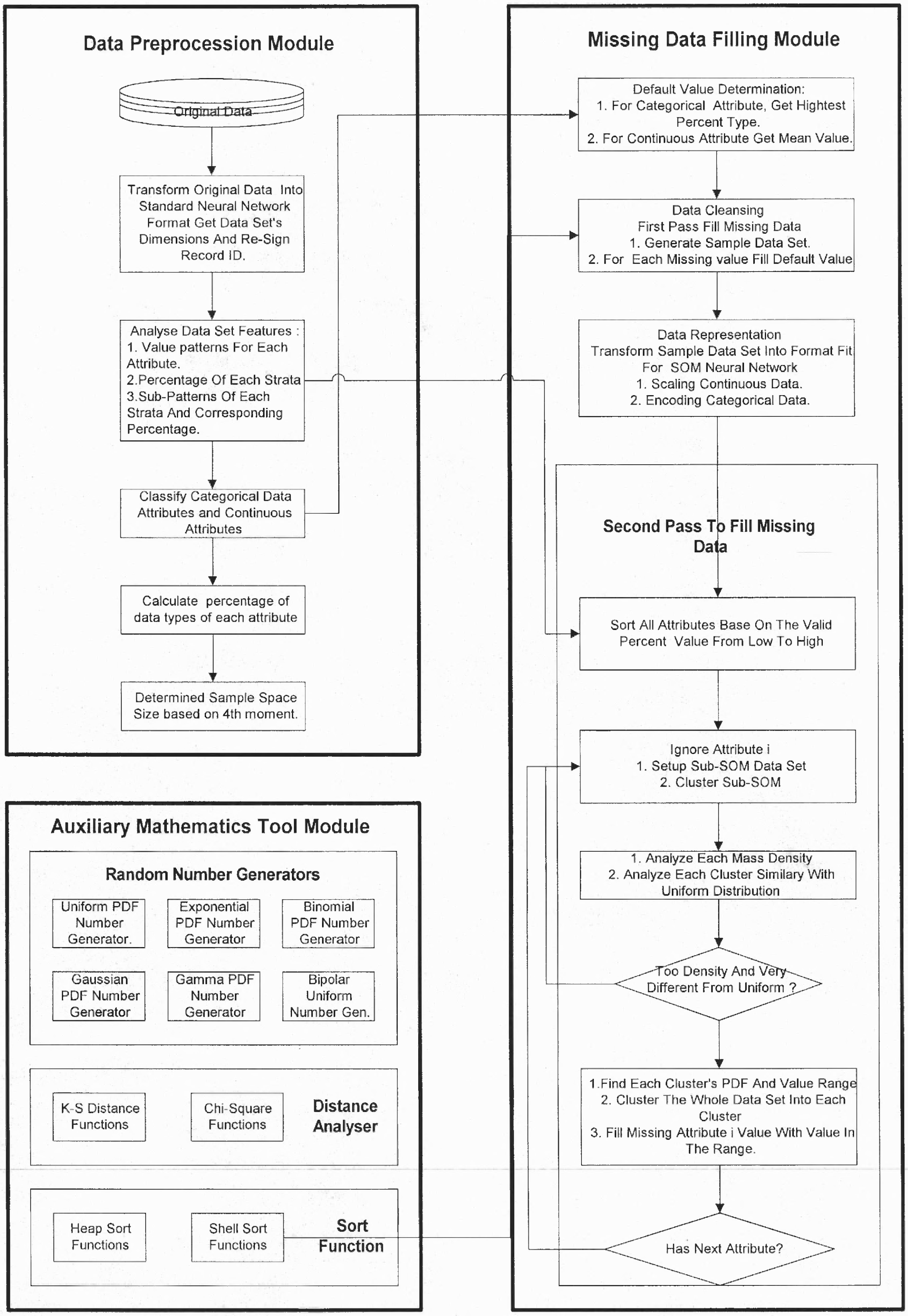

Data Preprocession Module
Original Data
Transform Original Data Into Standard Neural Network Format Get Data Set's Dimensions And Re-Sign Record ID.
Analyse Data Set Features : 1. Value patterns For Each Attribute. 2.Percentage Of Each Strata 3.Sub-Patterns Of Each Strata And Corresponding Percentage.
Classify Categorical Data Attributes and Continuous Attributes
Calculate percentage of data types of each attribute
Determined Sample Space Size based on 4th moment.
Auxiliary Mathematics Tool Module
Random Number Generators
Uniform PDF Number Generator.
Exponential PDF Number Generator
Binomial PDF Number Generator
Gaussian PDF Number Generator
Gamma PDF Number Generator
Bipolar Uniform Number Gen.
K-S Distance Functions
Chi-Square Functions
Distance Analyser
Heap Sort Functions
Shell Sort Functions
Sort Function
Missing Data Filling Module
Default Value Determination: 1. For Categorical Attribute, Get Hightest Percent Type. 2. For Continuous Attribute Get Mean Value.
Data Cleansing First Pass Fill Missing Data 1. Generate Sample Data Set. 2. For Each Missing value Fill Default Value
Data Representation Transform Sample Data Set Into Format Fit For SOM Neural Network 1. Scaling Continuous Data. 2. Encoding Categorical Data.
Second Pass To Fill Missing Data
Sort All Attributes Base On The Valid Percent Value From Low To High
Ignore Attribute i 1. Setup Sub-SOM Data Set 2. Cluster Sub-SOM
1. Analyze Each Mass Density 2. Analyze Each Cluster Similary With Uniform Distribution
Too Density And Very Different From Uniform ?
1.Find Each Cluster's PDF And Value Range 2. Cluster The Whole Data Set Into Each Cluster 3. Fill Missing Attribute i Value With Value In The Range.
Has Next Attribute?

Figure 4.1 System Architecture

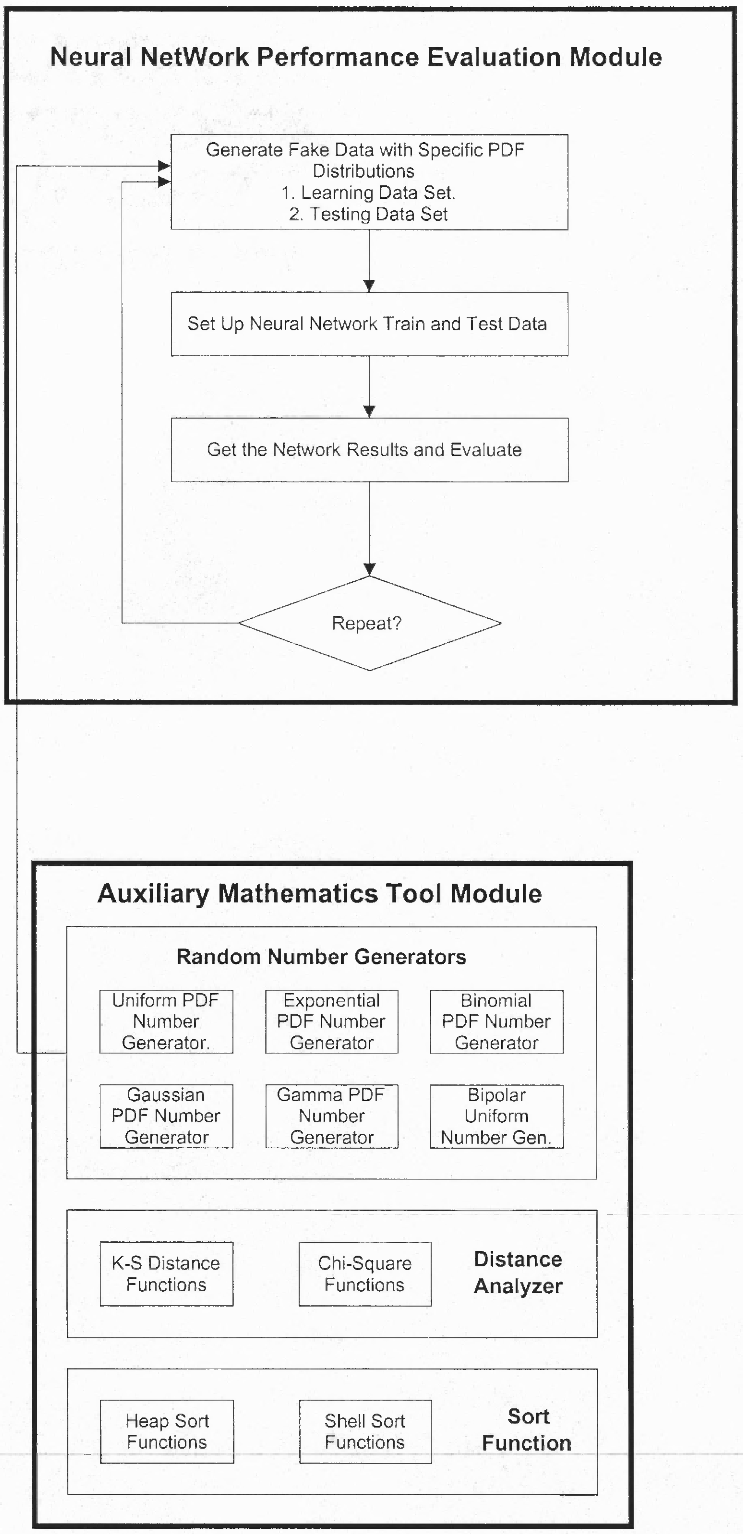

Neural NetWork Performance Evaluation Module
Generate Fake Data with Specific PDF Distributions
1. Learning Data Set.
2. Testing Data Set
Set Up Neural Network Train and Test Data
Get the Network Results and Evaluate
Repeat?
Auxiliary Mathematics Tool Module
Random Number Generators
Uniform PDF Number Generator.
Exponential PDF Number Generator
Binomial PDF Number Generator
Gaussian PDF Number Generator
Gamma PDF Number Generator
Bipolar Uniform Number Gen.
K-S Distance Functions
Chi-Square Functions
Distance Analyzer
Heap Sort Functions
Shell Sort Functions
Sort Function

Figure 4.2 System Architecture (cont.)

4.2 Data Preprocessing

A. Transform Data Format

The original data set is stored in a text file "joint.txt", which is about 150 M unformatted file. The original sample data is shown in Figure 4.3.

```
INDIV_ID    FICO  RESPOND   APPROVE   NET  MULTINOM  DECILE    IND2  IN
D4    IND12 IND14 IND16 IND17 IND23 IND25 IND26 IND35 IND36 IND38B      IND40B
IND37B      H2    H4    H11   H31   H32   H33   H43   H44   H45   H49   H52   H
53    H30B  ECMHZIP1    ZIP2  ZIP8  ZIP11 ZIP13 ZIP16 ZIP18 ZIP19 ZIP20 ZIP21 ZI
P22   ZIP24 ZIP25 ZIP30 ZIP31 ZIP32 ZIP33 ZIP34 ZIP35 ZIP36 ZIP39 ZIP41 ZIP42 ZI
P43   ZIP44 ZIP45 ZIP46 ZIP48 ZIP49 ZIP50 INFO2INFO6INFO7HHINFO1    HHINFO
2     HHINFO3     HHINFO4     HHINFO5     HHINFO45    HHINFO47    HHINFO48    H
HINFO60     HHINFO61    HHINFO69    HHINFO71    LIST  NEWAG13     NEWAG14     N
EWAG15      NEWAG18     NEWAG19     NEWAG20     NEWAG21     NEWAG26     NEWAG
43    NEWAG44     NEWAG45     NEWAG55     NEWAG57     NEWAG63     NEWAG69     N
EWAG72      NEWAG103 NEWAG104 NEWAG105 NEWAG1B      NEWAG2B     NEWAG
3B    NEWAG4B     NEWAG6B     NEWAG8B     PA_FLAG     PROM_FLA DM_FLAG     H
HINFO6B
57843       0                 0     4     MA          0     A     E     J     Y
N     N     41    F                             5     U     H     U     U     D     U
U     U           J     62    42.02       1     174.1 7.19  0.52  21800.38    2
16931.6     1     0.8   1.71  461.21      0.37  0.26  0.2   0.52  158529.33   16
2533.33     0.24  0.1   1     0.57  48971 124   125   244   200   125   64    2.
45    134
                  1011  5     5     5     2     4     4     4     0     0     0
0     0     0     4     4     0     0     0     0           -1.22 -1.22 10.45 -
1.22        0     1     1
58199       0                 0     3     MA          0     A     J     J     Y
N     N     64    C           772.24      772.5 5     U     H     C     N     U
G     D     2     2     G     45    72.03       1     175.74      5.26  0.3   15
293.83      2.43  12905.7     0.75  0.55  1.04  415.53      0.33  0.2   0.01  0.
26    60851.2     72680 0.09  0.04  0.26  0.26  41310 100   106   86    99    82
75    2.02  100
                        1111  4     5     5     1     2     3     3     0     0
1     1     0     0     2     1     0     0     0     0           2.47  -0.82 3.
68    2.47        0     1     1
```

Figure 4.3 Original Sample Data

All these original data used for data mining operations are the collection of symbols, which are unformatted and have variable lengths. The transform function *void* *tranRawToNN()* in system library "INTERFACE_H" analyzes the internal structure of original data set, transforms data into a standard format and fulfills the following functionalities:

(1) Delimitate each attribute value with white space and terminate each record by new line symbol;

(2) Mark all missing value with "-99" symbol;

(3) Resign records index; and

(4) Finally determine the dimension of original data set, which is 49,9992 rows and 110 columns and stored into a file named "NN.dat".

Figure 4.4 shows the formatted sample data after transformation.

```
 45 -99 0 -99 -99 0 1 MA 3 1 D E F Y N N 42 A -99 510.15 -99 05 U U C M M
U U 00 0 D 30 7.99 -99 1 190.00 4.67 0.00 6192.33 0.67 4848.59 0.01 0.01 3.00
57.33 0.33 0.33 0.02 0.00 -99 -99 0.00 0.33 0.67 0.00 21833 96 87 14 92 146 88
1.56 37 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 1011 2 2 2 2
2 2 2 0 2 2 2 0 0 0 0 0 0 0 0 -99 -99 -1.28 -1.28 -99 -99 0 1 1 -99

 274 -99 0 -99 -99 0 3 MA -99 1 D L Q N N N 0 A -99 996.66 -99 10 2 H C N
B L D 00 3 J 62 69.04 -99 1 206.61 3.68 0.11 15318.52 1.96 10727.15 0.58 0.58
1.07 227.95 0.14 0.00 0.00 0.14 50512.75 54999.00 0.21 0.00 0.11 0.14 47708 110
94 49 93 97 95 1.73 63 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -
99 11 2 3 3 0 2 2 2 0 0 0 0 0 0 2 2 0 0 0 0 -99 0.03 0.03 -99 0.03 -99 0 1 0
-99

 366 -99 0 -99 -99 0 99 MA 1 0 X U F N N N 0 U -99 -99 -99 -99 U U U U U
U U -99 -99 D 30 -99 -99 4 232.94 2.24 0.17 5432.09 0.87 5116.48 0.39 0.34 1.06
200.58 0.12 0.10 0.00 0.00 -99 -99 0.10 0.01 0.16 0.00 16641 68 56 10 73 139 95
1.38 29 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 1011 2 2 4 0
1 1 3 0 0 0 2 0 0 3 3 0 0 0 0 -99 0.1 0.1 6.21 0.1 -99 0 1 1 -99

 396 -99 0 -99 -99 0 6 MA -99 0 D C G Y N N 32 E -99 -99 -99 04 U U U U U
U U U -99 U 0 37.97 -99 1 105.40 2.33 1.00 13771.00 1.00 6798.25 1.00 0.75 0.00
0.35 0.02 0.02 0.99 0.33 122238.73 168949.81 0.67 0.00 0.00 0.33 21667 96 91 132
62 50 114 1.95 61 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99
1111 4 4 8 1 2 2 5 0 2 2 6 0 1 4 1 0 0 0 0 6.31 -1.05 -1.05 -1.05 8.12 -1.05 1
1 1 -99
```

Figure 4.4 Formatted Sample Data

B. Analyze Data Set Features

Data set features include the following items:

- Percentage of valid data of each attribute: This is very useful information as shown in Table 4.1, which helps us to get an insight of the whole data set. Furthermore, based on it, the sequence of filling missing data is decided and starts from the lowest one. The function *void rawValuePercent ()* in "INTERFACE_H" deals with this issue. As shown in Table 4.1, more than 70% attributes have over 90% or even 100% valid data, and only 20% attributes miss values more than 90%. Moreover, the attributes from the 80th to the 99th are full.

Table 4.1 Attribute valid data percent

Attribute	0	1	2	3	4	5	6	7	8	9
%	100	1.80	100	2.03	2.03	100	100	100	19.39	99.83
Attribute	10	11	12	13	14	15	16	17	18	19
%	95.59	99.83	95.44	99.83	99.83	99.83	99.83	99.83	0.56	38.55
Attribute	20	21	22	23	24	25	26	27	28	29
%	24.27	78.43	99.87	99.87	99.87	99.87	99.87	99.87	99.87	78.73
Attribute	30	31	32	33	34	35	36	37	38	39
%	48.18	98.52	98.52	50.58	2.78	95.92	95.30	95.92	95.90	95.01
Attribute	40	41	42	43	44	45	46	47	48	49
%	95.92	91.97	91.97	91.97	95.92	90.58	90.58	90.58	46.22	95.83
Attribute	50	51	52	53	54	55	56	57	58	59
%	71.24	71.24	95.90	95.92	95.92	95.91	95.92	95.92	95.92	95.92
Attribute	60	61	62	63	64	65	66	67	68	69
%	95.92	95.92	95.92	95.92	95.92	0.87	0.87	0.87	0.87	0.87
Attribute	70	71	72	73	74	75	76	77	78	79
%	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Attribute	80	81	82	83	84	85	86	87	88	89
%	100	100	100	100	100	100	100	100	100	100
Attribute	90	91	92	93	94	95	96	97	98	99
%	100	100	100	100	100	100	100	100	100	100
Attribute	100	101	102	103	104	105	106	107	108	109
%	10.35	44.19	47.90	34.41	39.89	9.79	100	100	100	0.22

- Percentage of strata: The definition of strata is the number of missing attributes that is denoted as strata_X. For example, if a record has 3 attributes with missing data, the record belongs to strata_3. The detailed report is enclosed in

"BANKDATASTRTA.txt" file. The rough distribution is that most records miss 30 ~ 31 attribute values, very few records have over 70 missing attribute values, and very few records miss less than 22 attribute values. There are no complete records.

- Sub-pattern of strata and its corresponding percentage: Following the above definition, if strata_3 has 2 records, one of which misses attributes 1,2,3 and the other 3,5,7, then there are two sub-patterns. The report is enclosed in "BANKDATASTRTA.txt" file. In order to visualize strata sub-pattern information, this work deploys a Java application "Ploygram", which arranges all attributes on a polygon and connects missing attributes with lines. The color of line stands for percentage of this sub-pattern, and the vertex of a polygon for its corresponding attribute. As shown in Figure 4.5, there are 12 polygons that mean strata_3 has 12 sub-patterns. The deep red color triangle shows that strata_3 sub-pattern has the highest percentage among all strata_3 sub-patterns.
- Value patterns for each attribute: The functions *void rawValuePatternRpt1()* and *void rawValuePatternRpt()* in System Library "INTERFACE_H" work on this issue. The only difference is the former uses static array and the latter uses dynamic linked-list. Report is saved in file " BANKDATAVALUE.txt", and Figure 4.6 shows value patterns sample.

C. Classify Categorical Attributes And Continuous Attribute

As show in Figure 4.4, it is very difficult to determine which attribute belongs to categorical or continuous attribute from the attribute tags. To solve this domain knowledge problem, this work introduces the rules for classification, which is that attributes with alphabetic symbols are categorical data, the ones with numerical symbols whose types are less than a threshold 64 are categorical data, and the rest ones belong to continuous data.

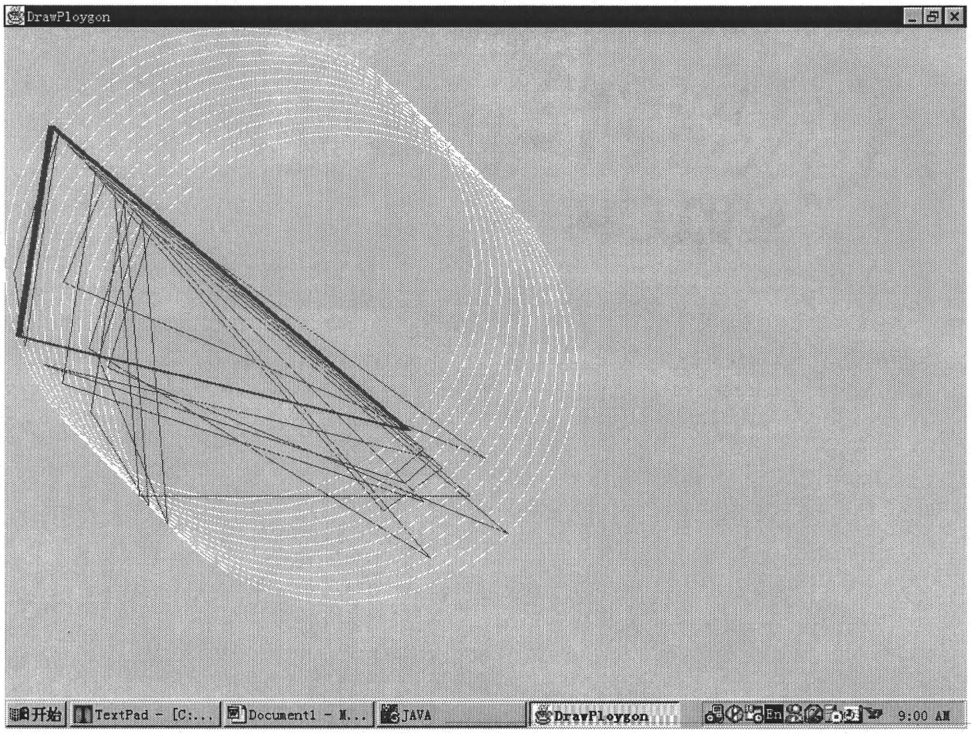


Figure 4.5 Strata Sample

D. Calculate data types percentage for each attribute

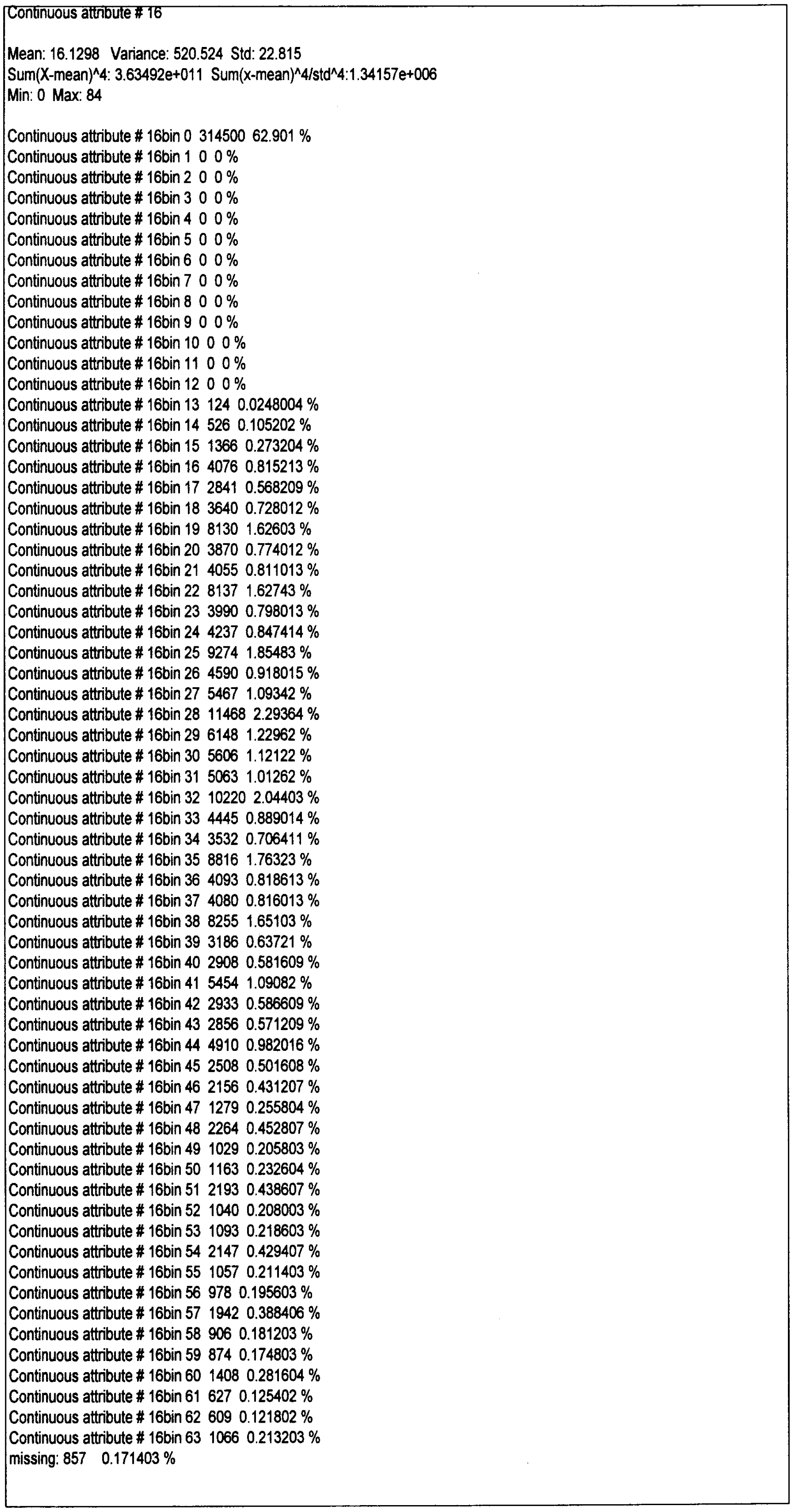
This analysis is necessary because the analysis can help system to classify categorical and continuous attributes, and determine the attributes' default values. Moreover, the peak of data types of each attribute can roughly determine the possible number of clusters, since an attribute may have several steep peaks that are distinguish with each other and may form corresponding clusters. Function *void rawPDFEncoder()* in "INTERFACE_H" deals with this issue and manually adjusts bin number. If an attribute is a continuous type, it calculates data types percentage by dividing attribute value into 64 bins. Otherwise, calculates categorical data types percentage based on how many categorical types. The detailed information is saved in “BANKDATAPDF.txt” file. The samples are shown in Figures 4.7 and 4.8.

```
Attribute # 1value type report:
667 675 595 542 588 565 631 623 585 617 557 641 551 663
740 643 727 695 603 642 644 568 685 628 578 592 584 636
655 621 538 648 489 729 579 674 529 587 637 626 711 598
649 494 645 735 501 594 543 732 718 567 539 666 613 488
624 556 508 582 651 622 604 706 619 552 503 520 632 554
669 634 560 608 687 627 657 678 478 483 583 531 744 671
656 696 724 519 607 658 670 610 689 629 723 571 570 535
533 775 633 576 688 647 618 652 747 593 716 555 700 561
605 699 659 612 753 514 638 614 749 504 713 572 748 705
609 731 676 591 506 664 581 692 625 611 766 710 536 665
602 787 616 734 526 679 680 522 564 599 523 684 751 498
...
Total valid values types are : 371
...
Attribute # 4value type report:
0 1
Total valid values types are : 2
Attribute # 5value type report:
0 1 2
Total valid values types are : 3
Attribute # 6value type report:
4 3 6 5 8 2 99 7 1
Total valid values types are : 9
Attribute # 7value type report:
MA CT PA OH TX SC NY MD NJ VT KS MO VA RI NH ME
FL LA NC CA OK KY AL WA GA OR AZ NV CO TN MN DC
IL MS DE IN WV NM MI MT AR HI ID IA UT NE WY SD ND
AK
Total valid values types are : 50
Attribute # 8value type report:
1 4 3 2 5
```

Figure 4.6 Value patterns Sample

```
Categorical attribute # 7  has  50  types

Type 1: MA  12881  2.57624 %
Type 2: CT  6555  1.31102 %
Type 3: PA  25176  5.03528 %
Type 4: OH  21200  4.24007 %
Type 5: TX  40810  8.16213 %
Type 6: SC  8542  1.70843 %
Type 7: NY  33463  6.69271 %
Type 8: MD  9661  1.93223 %
Type 9: NJ  14520  2.90405 %
Type 10: VT  985  0.197003 %
Type 11: KS  4252  0.850414 %
Type 12: MO  10143  2.02863 %
Type 13: VA  12654  2.53084 %
Type 14: RI  1933  0.386606 %
Type 15: NH  1552  0.310405 %
Type 16: ME  2476  0.495208 %
Type 17: FL  29019  5.80389 %
Type 18: LA  9591  1.91823 %
Type 19: NC  14660  2.93205 %
Type 20: CA  55804  11.161 %
Type 21: OK  5181  1.03622 %
Type 22: KY  8277  1.65543 %
Type 23: AL  8750  1.75003 %
Type 24: WA  9509  1.90183 %
Type 25: GA  15015  3.00305 %
Type 26: OR  5309  1.06182 %
Type 27: AZ  7393  1.47862 %
Type 28: NV  3180  0.63601 %
Type 29: CO  7385  1.47702 %
Type 30: TN  12262  2.45244 %
Type 31: MN  8360  1.67203 %
Type 32: DC  1012  0.202403 %
Type 33: IL  20331  4.06627 %
Type 34: MS  5765  1.15302 %
Type 35: DE  1572  0.314405 %
Type 36: IN  10489  2.09783 %
Type 37: WV  3438  0.687611 %
Type 38: NM  3152  0.63041 %
Type 39: MI  20155  4.03106 %
Type 40: MT  1691  0.338205 %
Type 41: AR  4713  0.942615 %
Type 42: HI  1927  0.385406 %
Type 43: ID  2294  0.458807 %
Type 44: IA  6310  1.26202 %
Type 45: UT  3471  0.694211 %
Type 46: NE  2894  0.578809 %
Type 47: WY  938  0.187603 %
Type 48: SD  1270  0.254004 %
Type 49: ND  1197  0.239404 %
Type 50: AK  875  0.175003 %
missing: 0   0 %
```

Figure 4.7 Data Type Percentage for Categorical Attributes



```
Continuous attribute # 16

Mean: 16.1298  Variance: 520.524  Std: 22.815
Sum(X-mean)^4: 3.63492e+011  Sum(x-mean)^4/std^4:1.34157e+006
Min: 0  Max: 84

Continuous attribute # 16bin 0  314500  62.901 %
Continuous attribute # 16bin 1  0  0 %
Continuous attribute # 16bin 2  0  0 %
Continuous attribute # 16bin 3  0  0 %
Continuous attribute # 16bin 4  0  0 %
Continuous attribute # 16bin 5  0  0 %
Continuous attribute # 16bin 6  0  0 %
Continuous attribute # 16bin 7  0  0 %
Continuous attribute # 16bin 8  0  0 %
Continuous attribute # 16bin 9  0  0 %
Continuous attribute # 16bin 10  0  0 %
Continuous attribute # 16bin 11  0  0 %
Continuous attribute # 16bin 12  0  0 %
Continuous attribute # 16bin 13  124  0.0248004 %
Continuous attribute # 16bin 14  526  0.105202 %
Continuous attribute # 16bin 15  1366  0.273204 %
Continuous attribute # 16bin 16  4076  0.815213 %
Continuous attribute # 16bin 17  2841  0.568209 %
Continuous attribute # 16bin 18  3640  0.728012 %
Continuous attribute # 16bin 19  8130  1.62603 %
Continuous attribute # 16bin 20  3870  0.774012 %
Continuous attribute # 16bin 21  4055  0.811013 %
Continuous attribute # 16bin 22  8137  1.62743 %
Continuous attribute # 16bin 23  3990  0.798013 %
Continuous attribute # 16bin 24  4237  0.847414 %
Continuous attribute # 16bin 25  9274  1.85483 %
Continuous attribute # 16bin 26  4590  0.918015 %
Continuous attribute # 16bin 27  5467  1.09342 %
Continuous attribute # 16bin 28  11468  2.29364 %
Continuous attribute # 16bin 29  6148  1.22962 %
Continuous attribute # 16bin 30  5606  1.12122 %
Continuous attribute # 16bin 31  5063  1.01262 %
Continuous attribute # 16bin 32  10220  2.04403 %
Continuous attribute # 16bin 33  4445  0.889014 %
Continuous attribute # 16bin 34  3532  0.706411 %
Continuous attribute # 16bin 35  8816  1.76323 %
Continuous attribute # 16bin 36  4093  0.818613 %
Continuous attribute # 16bin 37  4080  0.816013 %
Continuous attribute # 16bin 38  8255  1.65103 %
Continuous attribute # 16bin 39  3186  0.63721 %
Continuous attribute # 16bin 40  2908  0.581609 %
Continuous attribute # 16bin 41  5454  1.09082 %
Continuous attribute # 16bin 42  2933  0.586609 %
Continuous attribute # 16bin 43  2856  0.571209 %
Continuous attribute # 16bin 44  4910  0.982016 %
Continuous attribute # 16bin 45  2508  0.501608 %
Continuous attribute # 16bin 46  2156  0.431207 %
Continuous attribute # 16bin 47  1279  0.255804 %
Continuous attribute # 16bin 48  2264  0.452807 %
Continuous attribute # 16bin 49  1029  0.205803 %
Continuous attribute # 16bin 50  1163  0.232604 %
Continuous attribute # 16bin 51  2193  0.438607 %
Continuous attribute # 16bin 52  1040  0.208003 %
Continuous attribute # 16bin 53  1093  0.218603 %
Continuous attribute # 16bin 54  2147  0.429407 %
Continuous attribute # 16bin 55  1057  0.211403 %
Continuous attribute # 16bin 56  978  0.195603 %
Continuous attribute # 16bin 57  1942  0.388406 %
Continuous attribute # 16bin 58  906  0.181203 %
Continuous attribute # 16bin 59  874  0.174803 %
Continuous attribute # 16bin 60  1408  0.281604 %
Continuous attribute # 16bin 61  627  0.125402 %
Continuous attribute # 16bin 62  609  0.121802 %
Continuous attribute # 16bin 63  1066  0.213203 %
missing: 857    0.171403 %
```

Figure 4.8 Data Type Percentage for Continuous Attribute (64 bins)

E. Determine Sample Space

The whole data set has around 500,000 records; in the meanwhile, each record holds 110 attributes. To implement any algorithm on such a huge data set is infeasible since the time-complexity and space-complexity are overwhelming. Based on the data analysis presented in Chapter 3, the average traverse time of the whole data set is about 70 minutes on PC Pentium II 350MHz, 128M RAM. On the other hand, dealing with the whole data set doesn't guarantee the best filling quality because if the original data contain "erroneous" values, the larger the number of records, the more the "dirty" values. Moreover, in a large data set, there are lots of similar or duplicated records together. Hence, how to obtain the good trade-off among time and space complexity and quality is very important. The key point to determine sample space is to obtain the smallest subset, which can best represent the original data set. Following the line of this thought, *Estimate Representative Small Sets* algorithm is developed as follows, which has two functions *Fourth Moment* and *Sample Size Determination*:

Estimate Representative Small Sets Algorithm:

Fourth Moment ():

$$\mu = \sum_{1}^{N_b} v_i \times p_i \tag{4.1}$$

$$\delta = \sum_{1}^{N_b} (v_i - \mu)^2 \times p_i \tag{4.2}$$

$$M = \sum_{1}^{N_b} (v_i - \mu)^4 \times p_i \tag{4.3}$$

Sample Size Determination ():

$$T = \left[{(t \times \sigma)}\big/{(r * \mu)} \right]^2 \tag{4.4}$$

$$S = {T}\big/{(1 + T/N)} \tag{4.5}$$

Where

N_b: Total number of bins;

v_i: Value of the i^{th} bin;

p_i: Possibility value of the i^{th} bin;

δ : Standard deviation;

μ: Mean value;

M: Fourth moment value;

S: Sample space size;

r: Desired error; and

t: Appropriate number from the standard normal table.

The function *void fourthMoment()* deals with this issue to determine the thickness of a distribution's head and tail. In the case r = 0.01 and t =2.48, the desired error is not higher than 1.0% and the sample space size is about 8000 records.

4.3 Filling Missing Data

While the goal of data mining is to extract valuable information from data, it is an undeniable fact that the quality of the results is related directly to the quantity and quality of the data being mined [Bigus, 1996]. As mentioned in Chapter 3, most records in the data set miss 30 attributes and there are no complete records. Thus, how to properly fill missing data is the critical point to guarantee enough and high quality data for data mining. Furthermore, filling missing data is not a simple top-to-down process; instead it is a recursive routine and art.

4.3.1 Default Value Determination

The most straightforward way to fill missing hole is using the most feasible values. Based on the possibility, this way guarantees the highest "hit-ratio". For a categorical attribute, the highest percentage type is selected as a default value. While for a continuous attribute, the mean value is chosen as a default value. Table 4.2 lists all attributes' default values.

Table 4.2 Default values

Att.	0	1	2	3	4	5	6	7	8	9
Def	N/A	615.98	0	0	0	0	99	CA	1	0
Att	10	11	12	13	14	15	16	17	18	19
Def	A	U	G	N	N	N	16.13	U	536.88	603.10
Att	20	21	22	23	24	25	26	27	28	29
Def	562.32	U	U	U	U	U	U	U	U	U
Att	30	31	32	33	34	35	36	37	38	39
Def	0	U	26.76	48.13	1	1	157.08	4.44	0.35	20012
Att	40	41	42	43	44	45	46	47	48	49
Def	1.80	12290.7	1.03	0.85	1.01	503	0.39	0.28	0.18	0.29
Att	50	51	52	53	54	55	56	57	58	59
Def	88413.5	98874.8	0.28	0.04	0.29	0.37	42469.8	110	100.32	99.57
Att	60	61	62	63	64	65	66	67	68	69
Def	98.69	99.09	100	1.95	99.07	M	U	0.4	66.01	U
Att	70	71	72	73	74	75	76	77	78	79
Def	U	U	U	U	15	S	6	U	N	U
Att	80	81	82	83	84	85	86	87	88	89
Def	10	0	0	0	0	0	0	0	0	0
Att	90	91	92	93	94	95	96	97	98	99
Def	0	0	0	0	0	0	0	0	0	0
Att	100	101	102	103	104	105	106	107	108	109
Def	1.91	1.52	0.88	4.29	1.89	2.93	0	1	1	1961.89

4.3.2 Data Cleansing:

With the sample space representing the whole data set with small record numbers and the default value to fill missing holes, the procedures for data cleansing are listed as follows:

- Randomly generate indexes within the whole data set with range [1, 49993] and set a flag for each candidate record that will be contained in the sample data set.
- Open the source standard data set file, retrieve one record and check whether it is a candidate record. If not, this record is ignored. And the next record is retrieved. This step continues until the end of the data file is reached.
- For each candidate record, traverse its attributes. For each attribute, check whether the value is missing (missing flag is "-99") or not. If the attribute is missing, replace the missing hole with the default value.
- Store all filled sample records into a new file and data into another one.

4.3.3 Data Representation

After the data cleansing, all missing holes have been replaced with the default value. However, this complete sample data set cannot be directly fed into neural network because some attributes are categorical data, such as "A, E …" or "Y, N" etc. Moreover, neural network requires that its input data are ranged between –1 and +1. Thus, to convert categorical data into numeric format by specific encoding scheme and scale all numeric data into value range [-1, +1] is our next step to avoid possible "saturation" state in neural network and get the best performance

For categorical attribute, the encoding scheme algorithm is developed as follows:

Categorical Attribute Encoding Scheme Algorithm:

```
If (categorical attribute type = 1)
```

```
        Ignore this attribute;
Else If (categorical attribute type = 2)
        Then encode two types as (-0.355, +0.355) and (+0.355, -0.355).
Else
    i)  Centralize all types so that the higher percent the near to center;
    ii) Divide all types into 3 sub-group, which are left, middle and right group.
    iii) If (group = left group)
            Code is (+0.355, -0.355, -0.355);
         Else if (group = middle group)
            Code is (-0.355, +0.355, -0.355);
         Else
            Code is (-0.355, -0.355, +0.355).
```

For continuous data the scaling scheme is:

$$V_{ij}' = \frac{V_{ij} - \mu_i}{\delta_i}$$

Where

V_{ij}': The scaled and normalized value for the i^{th} attribute of the j^{th} record;

V_{ij}: The original value of the i^{th} attribute of the j^{th} record;

δ_i: The standard deviation of the i^{th} attribute; and

μ_i: The mean of the i^{th} attribute.

Function *void SOM()* is used to convert encoded attribute values to fit the Self-Organization-Map neural network input format, and outputs two files: one is SOM format data file shown in Figure 4.9, the other is SOM interpretation format data file shown in Figure 4.10.

The symbols in the sample are listed as follows:

- $ and !: indicate unfilled categorical data alternately;
- &: indicates unfilled continuous data;
- * and ^: indicate filled categorical data alternately;
- #: indicates filled continuous data;

4.3.4 Data Filling Sequence Determination

After the data cleansing, all missing "holes" are replaced with the default value. In most data mining case studies, this way is good enough for filling missing data [Trippi and Turban, 1996]. However, it assumes that every record in the whole data set has the similar characteristics, which may be incorrect. Thus, to make the filling more accurate, an innovative algorithm *Second Pass Filling Missing Data* is developed and presented in the next section. The first step of this algorithm is to determine the filling sequence. In this work, the sequence is determined on attribute valid data percentage, and ordered from the lowest to the highest. It is fulfilled by function *void SOM ()* in "PHASE2_H" library. Table 4.3 shows the results.

4.3.5 Second Pass Filling Missing Data

Based on the sequence in Table 4.3, filling missing data use the following algorithm:

Second Pass Filling Missing Data Algorithm:

i) Based on the filling sequence and partially filled data set ignore the attribute one time and create a subset which contains the rest attribute value.

```
SNNS pattern definition file V4.1
generated at Thu Apr 21 19:56:37 2001


No. of patterns : 8000
No. of input units : 230

# Input pattern 45:
0 -0.17395 0.17395 -0.15265 0.15265 -0.15265 0.15265 -0.23075 0.23075 -0.23075 0.1349 -0.1349 -0.1349 -0.1704 0.1704 -0.1704
0.15265 -0.15265 -0.15265 -0.25915 0.25915 -0.25915 -0.24495 0.24495 -0.24495 -0.18105 0.18105 -0.18105 -0.15265 0.15265 -0.15265
0.08875 -0.08875 -0.1491 0.1491 -0.1633 0.1633 1.15648 -0.15975 -0.15975 0.15975 0 -0.411924 0 -0.1846 0.1846 -0.1846 -0.25205
0.25205 -0.25205 -0.1775 0.1775 -0.1775 -0.28755 0.28755 -0.28755 -0.1917 0.1917 -0.1917 -0.20235 0.20235 -0.20235 -0.2911 0.2911 -
0.2911 -0.3053 0.3053 -0.3053 -0.2272 0.2272 -0.2272 -0.1917 0.1917 -0.1917 -0.1846 0.1846 -0.1846 0.137598 -1.00652 -0.25205
0.25205 -0.25205 0.510305 0.112652 -0.713579 -1.10521 -0.924837 -0.834309 -1.38872 -1.27312 1.95466 -0.636926 -0.0826133
0.116667 -0.391182 -0.930763 0 0 -0.77337 2.12029 0.88486 -0.917734 -0.83269 -0.15975 0.15975 -0.15975 -0.446551 -1.14105 -
0.0996102 0.522938 0.17395 -0.17395 -0.17395 -0.864094 -1.21331 -0.13135 0.13135 -0.13135 -0.1562 0.1562 -0.1562 -0.142 0.142 -
0.142 0 -0.2556 0.2556 -0.2556 -0.24495 0.24495 -0.24495 -0.25915 0.25915 -0.25915 -0.26625 0.26625 -0.26625 -0.2485 0.2485 -
0.2485 -0.1278 0.1278 -0.1278 -0.15975 0.15975 -0.15975 -0.1207 0.1207 -0.1207 -0.20945 0.20945 -0.20945 -0.12425 0.12425 -0.2556
0.2556 -0.2556 -0.16685 0.16685 -0.16685 -0.2698 0.2698 -0.2698 -0.26625 0.26625 -0.26625 -0.2698 0.2698 -0.2698 -0.25915 0.25915 -
0.25915 -0.25915 0.25915 -0.25915 -0.2556 0.2556 -0.2556 -0.25915 0.25915 -0.25915 -0.30885 0.30885 -0.30885 -0.28755 0.28755 -
0.28755 -0.2911 0.2911 -0.2911 -0.28755 0.28755 -0.28755 -0.30175 0.30175 -0.30175 -0.30885 0.30885 -0.30885 -0.25915 0.25915 -
0.25915 -0.26625 0.26625 -0.26625 -0.25205 0.25205 -0.25205 -0.30175 0.30175 -0.30175 -0.3124 0.3124 -0.3124 -0.30885 0.30885 -
0.30885 0 0 -0.599792 -1.14572 0 0 -0.15265 0.15265 0.0639 -0.0639 0.09585 -0.09585 0
```

Figure 4.9 SOM Data Sample

```
 0 #0 $-0.355 $0.355 *-0.355 *0.355 ^-0.355 ^0.355 !-0.355 !0.355 !-0.355 $-0.355 $-0.355 $0.355 !-0.355 !0.355 !-0.355 *-
0.355 *0.355 *-0.355 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !0.355 !-0.355 !-0.355 $0.355 $-
0.355 !-0.355 !0.355 $-0.355 $0.355 &1.11416 !-0.355 !0.355 !-0.355 #0 #0 #0 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355
$-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !-0.355 !-0.355 !0.355 $-0.355 $0.355 $-0.355 !-0.355
!0.355 !-0.355 $-0.355 $0.355 $-0.355 ^-0.355 ^0.355 ^-0.355 !-0.355 !0.355 !-0.355 &1.45774 &-0.162296 $-0.355 $0.355 $-
0.355 &0.26785 &1.20017 &0.500938 &0.142154 &0.182534 &0.523925 &-0.0321248 &-0.0786546 &0.99783 &-0.0679357 &-
0.055429 &-0.0664852 &0.0538864 &0.754906 &0.920553 &0.758755 &-0.115874 &0.589314 &2.23375 &0.509541 &0.270454 !-
0.355 !-0.355 !0.355 &0.871755 &1.97312 &1.77604 &0.277347 $0.355 $-0.355 $-0.355 &1.64635 &0.690411 *-0.355 *0.355 *-
0.355 ^-0.355 ^0.355 ^-0.355 *-0.355 *0.355 *-0.355 #0 ^-0.355 ^0.355 ^-0.355 *-0.355 *0.355 *-0.355 ^-0.355 ^0.355 ^-0.355 *
0.355 *0.355 *-0.355 ^-0.355 ^0.355 ^-0.355 *-0.355 *0.355 *-0.355 ^-0.355 ^0.355 ^-0.355 *-0.355 *0.355 *-0.355 ^-0.355 ^0.3
^-0.355 *-0.355 *0.355 ^-0.355 ^0.355 ^-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355
$0.355 $-0.355 !-0.355 !0.355 !-0.355 $0.355 $-0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-
0.355 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !-
0.355 !0.355 !-0.355 $0.355 $-0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355 $0.355 $-0.355 !-0.355 !0.355 !-0.355 $-0.355
$0.355 $-0.355 #0 &-0.799221 &-0.634348 &1.49601 &-0.952405 #0 !-0.355 !0.355 $0.355 $-0.355 !0.355 !-0.355 #0
```

Figure 4.10 Interpretation Sample SOM Data

Table 4.3 Data filling sequence

Sequence	0	1	2	3	4	5	6	7	8	9
Attribute	109	18	66	68	70	67	72	73	74	75
Seq	10	11	12	13	14	15	16	17	18	19
Attribute	76	77	65	79	71	69	78	1	3	4
Seq	20	21	22	23	24	25	26	27	28	29
Attribute	34	105	100	8	20	103	19	104	101	48
Seq	30	31	32	33	34	35	36	37	38	39
Attribute	102	30	33	51	50	21	29	47	45	46
Seq	40	41	42	43	44	45	46	47	48	49
Attribute	42	43	41	39	36	12	10	49	38	52
Seq	50	51	52	53	54	55	56	57	58	59
Attribute	55	53	54	59	56	35	64	58	44	61
Seq	60	61	62	63	64	65	66	67	68	69
Attribute	57	62	40	60	37	63	31	32	13	16
Seq	70	71	72	73	74	75	76	77	78	79
Attribute	15	17	9	11	14	28	22	26	27	23
Seq	80	81	82	83	84	85	86	87	88	89
Attribute	25	24	2	6	7	80	81	82	84	85
Seq	90	91	92	93	94	95	96	97	98	99
Attribute	86	5	0	93	94	95	83	97	98	99
Seq	100	101	102	103	104	105	106	107	108	109
Attribute	87	88	89	90	91	92	106	107	108	96

```
ii)  For (i = 0; i < p; i++)
     {
         Randomly generate vector $\nabla i$, which contain $N_a$ attributes.
     }// Initialize cluster vector.
iii) For (l=0; l<C; l++)
     {
         For (i = 0; i< $N_b$; i++)
         {
             Retrieve a record vector $\eta_i$ from sample data set.
             For (j=0; j<p; j++)
             {
```

```
            Calculate Euclidian Distance between $\nabla i$ and $\eta_i$.
        }
        For (k=0; k < p; k++)
        {
            If ( $\nabla_{ij}$ has the smallest Euclidian Distance value)
            {
                A) Update $\nabla i$ as: $\nabla_i^{new} \equiv \alpha\nabla_i^{old} + (1-\alpha)\eta_i$
                B) Mark $\nabla i$ as winner vector.
            }
            Else
                Continue;
        }
    }
}//Get all winner vectors after C loops update, suppose the number is P;
iv) For (i = 0; i< $N_b$; i++)
    {
        Retrieve a record vector $\eta_i$ from sample data set.
        For (j=0; j<P; j++)
        {
            Calculate Euclidian Distance between $\eta_i$ and $\nabla_i$;
        }
        If (The vector $\nabla_i$ has the smallest Euclidian Distance)
        {
            For (k=0; k<$N_a$; k++)
            {
                If (attribute k has valid data value)
                    Save the valid data value into $Matrix_{ij}$;
                Else
                    Continue;
            }
        }
        Else
            Continue;
    } // Get valid data value from sample data set for each winner cluster's every
attribute.
v)  For (i = 0; i<$N_b$; i++)
    {
        Retrieve a record vector $\eta_i$ from sample data set.
        For (j=0; j<P; j++)
        {
            Calculate Euclidian Distance between $\eta_i$ and $\nabla_i$;
        }
        If (The vector $\nabla_i$ has the smallest Euclidian Distance)
        {
            For (k=0; k< $N_a$ ; k++)
            {
                If (attribute k miss data value)
                    Randomly select a valid data value from $Matrix_{ij}$ and
                    Replace the missing hole;
                Else
                    Continue;
            }
        }
    } // Partially filling missing data in sample data set.
    If (there still has attributes need to fill)
        Go back to step i);
    Else
        Continue;
vi) For (i = 0; i< $N_c$ ; i++)
    {
        Retrieve a record vector $\eta_i$ from original data set.
        For (j=0; j<P; j++)
        {
            Calculate Euclidian Distance between $\eta_i$ and $\nabla_i$ ;
        }
        If (The vector $\nabla_i$ has the smallest Euclidian Distance)
        {
            For (k=0; k<$N_a$; k++)
            {
                If (attribute k miss data value)
                    Randomly select a valid data value from $Matrix_{ij}$ and
                    Replace the missing hole;
                Else
                    Continue;
            }
        }
    } // Fill missing data to original data set.
```

Where:

N_a: Total number of attributes;

N_b: Total number of sample set records;

N_c: Total number of original data set records;

C: The number of loops;

∇i: the i^{th} Vector;

η_i: the i^{th} record;

α: Learning Rate;

p: Number of initialization vectors;

P: Number of winner vectors; and

$Matrix_{ij}$: Valid value for the i^{th} cluster of the j^{th} attribute.

Functions *void Cluster()* and *void finalFill()* in "PHASE_2.H" fulfill the algorithm. Figures 4.11 and 4.12 present the sample report and sample winner vectors after cluster, respectively. And Figure 4.13 shows the sample data after the second filling, where all filled data are marked with "*" in order to distinguish them with the original values.

4.4 Neural Network Performance Evaluation

Given a set of complete data, the other important task of the work is to predict target data. To achieve this task, Back-Propagation neural network model is selected. Before building up Back-Propagation model to predict credit card target data, preliminary evaluation is performed to test whether this model works and how well it is:

- Generate learning, testing complete data set: generate learning and testing set at specific ratio, whose attributes, records number and output results are manageable. This task is handled by function *void RandomDataGenerator()* in "SUBFUNCTION_H" library, which invoke functions in "AUXMATH_H" library listed as follows:

  (1) *Float ran1 (long *)* generates uniform distribution value between [0,1];

  (2) *Float expdev(long *)* generates exponential distribution value;

  (3) *Float gasdev(long *idum)* generates gaussian or normal distribution value;

  (4) *Float gamdev(int ia, long * idum)* generates gamma distribution value; and

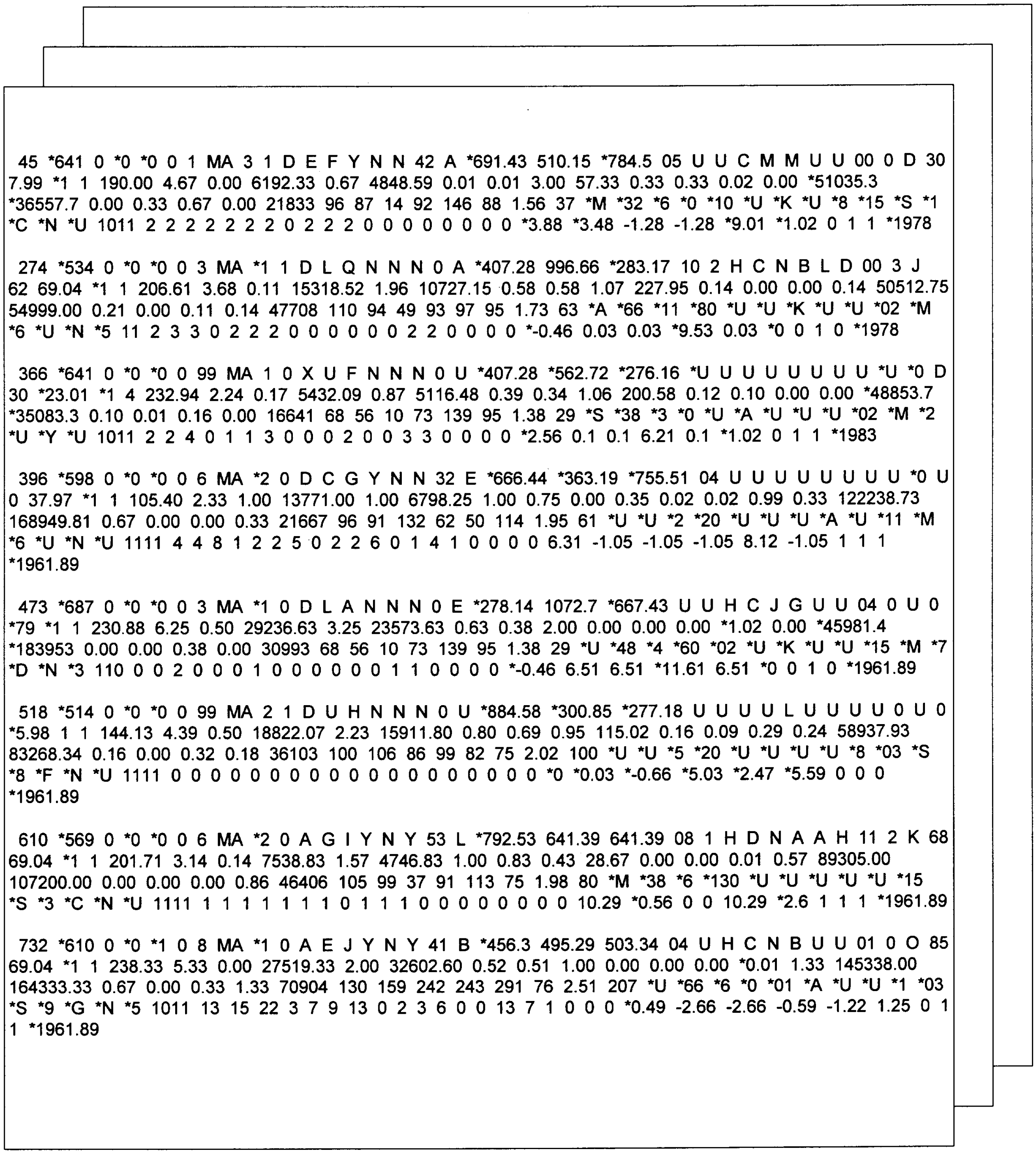
  (5) *Float bnldev(float pp, int n, long *idum)* generates binomial distribution value.

```
Second Pass Filling  Missing Data Function

Attribute 109 Total 17 Clusters

Cluster 0 has 74    Cluster 1 has 292    Cluster 2 has 689  Cluster 3 has 561   Cluster 4 has 152
Cluster 5 has 397  Cluster 6 has 1211   Cluster 7 has 161  Cluster 8 has 1619 Cluster 9 has 108
Cluster 10 has 719Cluster 11 has 119   Cluster 12 has 426Cluster 13 has 301 Cluster 14 has 474
Cluster 15 has 477Cluster 16 has 109   Cluster 17 has 111

Cluster Vadlid Data Of Attribute 109

cluster 0 has0 valid data

cluster 1 has5 valid data
1960  1978  1975  1960  1986
cluster 2 has0 valid data

cluster 3 has2 valid data
1984  1940
cluster 4 has1 valid data
1941
cluster 5 has0 valid data

cluster 6 has2 valid data
1953  1910
cluster 7 has1 valid data
1959
cluster 8 has3 valid data
1951  1983  1978
cluster 9 has0 valid data

cluster 10 has1 valid data
1983
cluster 11 has0 valid data

cluster 12 has4 valid data
1971  1956  1993  1985
cluster 13 has1 valid data
1958
cluster 14 has3 valid data
1994  1992  1995
cluster 15 has0 valid data

cluster 16 has0 valid data

cluster 17 has1 valid data
1939
```

Figure 4.11 Sample Cluster report

```
Attribute 109
Vector Number 17
Vector  Size 229.

  0 |     | 229:-0.04133, 228: 0.02271, 227: 0.00425, 226:-0.02069, 225: 0.03213, 224:-0.03672, 223:
0.02288, 222: 0.05486,
          221:-0.05929, 220: 0.03831, 219:-0.04518, 218: 0.03659, 217:-0.11528, 216: 0.05014, 215:-
0.04020, 214:-0.04790,
          213: 0.12835, 212:-0.13904, 211:-0.06395, 210: 0.05362, 209:-0.11137, 208:-0.10531, 207:
0.03213, 206:-0.08993,
          205:-0.06842, 204: 0.05639, 203:-0.04293, 202:-0.07297, 201: 0.08200, 200:-0.03965, 199:-
0.04761, 198: 0.11049,
          197:-0.13133, 196:-0.05247, 195: 0.05153, 194:-0.11321, 193:-0.11351, 192: 0.12280, 191:-
0.06423, 190:-0.04724,
          189: 0.04183, 188:-0.11632, 187:-0.07537, 186: 0.11017, 185:-0.10028, 184:-0.04364, 183:
0.11206, 182:-0.05211,
          181:-0.04013, 180: 0.01647, 179:-0.09612, 178:-0.06731, 177: 0.06052, 176:-0.03372, 175:-
0.03646, 174: 0.04137,
          173:-0.07954, 172:-0.09723, 171: 0.07732, 170:-0.11937, 169:-0.04831, 168: 0.08119, 167:-
0.08572, 166:-0.05801,
          165: 0.01661, 164:-0.06663, 163:-0.11796, 162: 0.10308, 161:-0.07712, 160:-0.03033, 159:
0.02187, 158:-0.02736,
          157:-0.04540, 156: 0.02985, 155:-0.05388, 154: 0.04207, 153: 0.01156, 152:-0.08956, 151:
0.05579, 150:-0.01758,
          149:-0.08261, 148:-0.01136, 147:-0.01936, 146:-0.01582, 145: 0.03599, 144:-0.00406, 143:-
0.06949, 142: 0.02308,
          141:-0.00733, 140:-0.09072, 139: 0.12001, 138:-0.10923, 137:-0.03803, 136: 0.07811, 135:-
0.10824, 134:-0.04290,
          133: 0.12422, 132:-0.08287, 131:-0.04712, 130: 0.06692, 129:-0.04956, 128:-0.06253, 127:
0.09790, 126:-0.12478,
          125:-0.00164, 124:-0.00267, 123: 0.00017, 122:-0.02752, 121:-0.08331, 120: 0.02533, 119:-
0.01253, 118:-0.06479,
          117: 0.05101, 116:-0.00226, 115:-0.08656, 114:-0.04237, 113:-0.02720, 112: 0.01381, 111:-
0.05686, 110:-0.03022,
          109:-0.04186, 108:-0.02638, 107: 0.01406, 106:-0.02438, 105:-0.00362, 104:-0.04037, 103:-
0.09266, 102:-0.02614,
          101:-0.04794, 100: 0.01113,  99: 0.00344,  98: 0.05198,  97:-0.01340,  96: 0.00350,  95:-
0.02172,  94:-0.03625,
           93: 0.02360,  92: 0.02631,  91: 0.06442,  90: 0.03404,  89:-0.01295,  88: 0.08826,  87:
0.08334,  86:-0.02672,
           85:-0.08223,  84: 0.01574,  83: 0.02337,  82:-0.09113,  81: 0.08241,  80:-0.06680,  79:
0.25024,  78: 0.16346,
           77:-0.00171,  76: 0.01697,  75: 0.00053,  74:-0.07232,  73: 0.01273,  72:-0.01823,  71:-
0.08699,  70: 0.05411,
           69:-0.09246,  68:-0.10449,  67: 0.07589,  66:-0.06186,  65:-0.04037,  64: 0.05932,  63:-
0.12516,  62:-0.03540,
           61:-0.01018,  60:-0.03786,  59:-0.05000,  58: 0.00706,  57:-0.00122,  56:-0.09274,  55:
0.07594,  54:-0.10012,
           53:-0.07724,  52: 0.04324,  51:-0.00039,  50:-0.03620,  49: 0.03327,  48:-0.02697,  47:-
0.02240,  46: 0.07419,
           45:-0.01546,  44:-0.00267,  43: 0.13828,  42:-0.04162,  41:-0.01073,  40:-0.00746,  39:-
0.03227,  38: 0.20579,
           37: 0.06884,  36:-0.01737,  35: 0.08420,  34: 0.00303,  33: 0.01999,  32:-0.01405,  31:-
0.02207,  30: 0.05611,
           29: 0.00724,  28:-0.01789,  27: 0.02833,  26:-0.03820,  25:-0.10115,  24: 0.08367,  23:-
0.02185,  22:-0.03750,
           21: 0.05873,  20:-0.07095,  19:-0.06224,  18: 0.01268,  17:-0.08087,  16: 0.00822,  15:
0.00646,  14: 0.00446,
           13: 0.00442,  12:-0.00126,  11: 0.02376,  10:-0.03107,   9: 0.03389,   8:-0.09208,   7: 0.01359,
6:-0.05616,
            5: 0.00643,   4:-0.02425,   3: 0.06419,   2:-0.09067,   1:-0.04152
```

Figure 4.12 Winner Vector Sample



```
 45 *641 0 *0 *0 0 1 MA 3 1 D E F Y N N 42 A *691.43 510.15 *784.5 05 U U C M M U U 00 0 D 30
7.99 *1 1 190.00 4.67 0.00 6192.33 0.67 4848.59 0.01 0.01 3.00 57.33 0.33 0.33 0.02 0.00 *51035.3
*36557.7 0.00 0.33 0.67 0.00 21833 96 87 14 92 146 88 1.56 37 *M *32 *6 *0 *10 *U *K *U *8 *15 *S *1
*C *N *U 1011 2 2 2 2 2 2 2 0 2 2 2 0 0 0 0 0 0 0 *3.88 *3.48 -1.28 -1.28 *9.01 *1.02 0 1 1 *1978

 274 *534 0 *0 *0 0 3 MA *1 1 D L Q N N N 0 A *407.28 996.66 *283.17 10 2 H C N B L D 00 3 J
62 69.04 *1 1 206.61 3.68 0.11 15318.52 1.96 10727.15 0.58 0.58 1.07 227.95 0.14 0.00 0.00 0.14 50512.75
54999.00 0.21 0.00 0.11 0.14 47708 110 94 49 93 97 95 1.73 63 *A *66 *11 *80 *U *U *K *U *U *02 *M
*6 *U *N *5 11 2 3 3 0 2 2 2 0 0 0 0 0 0 2 2 0 0 0 0 *-0.46 0.03 0.03 *9.53 0.03 *0 0 1 0 *1978

 366 *641 0 *0 *0 0 99 MA 1 0 X U F N N N 0 U *407.28 *562.72 *276.16 *U U U U U U U U *U *0 D
30 *23.01 *1 4 232.94 2.24 0.17 5432.09 0.87 5116.48 0.39 0.34 1.06 200.58 0.12 0.10 0.00 0.00 *48853.7
*35083.3 0.10 0.01 0.16 0.00 16641 68 56 10 73 139 95 1.38 29 *S *38 *3 *0 *U *A *U *U *U *02 *M *2
*U *Y *U 1011 2 2 4 0 1 1 3 0 0 0 2 0 0 3 3 0 0 0 0 *2.56 0.1 0.1 6.21 0.1 *1.02 0 1 1 *1983

 396 *598 0 *0 *0 0 6 MA *2 0 D C G Y N N 32 E *666.44 *363.19 *755.51 04 U U U U U U U U *0 U
0 37.97 *1 1 105.40 2.33 1.00 13771.00 1.00 6798.25 1.00 0.75 0.00 0.35 0.02 0.02 0.99 0.33 122238.73
168949.81 0.67 0.00 0.00 0.33 21667 96 91 132 62 50 114 1.95 61 *U *U *2 *20 *U *U *U *A *U *11 *M
*6 *U *N *U 1111 4 4 8 1 2 2 5 0 2 2 6 0 1 4 1 0 0 0 0 6.31 -1.05 -1.05 -1.05 8.12 -1.05 1 1 1
*1961.89

 473 *687 0 *0 *0 0 3 MA *1 0 D L A N N N 0 E *278.14 1072.7 *667.43 U U H C J G U U 04 0 U 0
*79 *1 1 230.88 6.25 0.50 29236.63 3.25 23573.63 0.63 0.38 2.00 0.00 0.00 0.00 *1.02 0.00 *45981.4
*183953 0.00 0.00 0.38 0.00 30993 68 56 10 73 139 95 1.38 29 *U *48 *4 *60 *02 *U *K *U *U *15 *M *7
*D *N *3 110 0 0 2 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 *-0.46 6.51 6.51 *11.61 6.51 *0 0 1 0 *1961.89

 518 *514 0 *0 *0 0 99 MA 2 1 D U H N N N 0 U *884.58 *300.85 *277.18 U U U U L U U U U 0 U 0
*5.98 1 1 144.13 4.39 0.50 18822.07 2.23 15911.80 0.80 0.69 0.95 115.02 0.16 0.09 0.29 0.24 58937.93
83268.34 0.16 0.00 0.32 0.18 36103 100 106 86 99 82 75 2.02 100 *U *U *5 *20 *U *U *U *U *8 *03 *S
*8 *F *N *U 1111 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 *0 *0.03 *-0.66 *5.03 *2.47 *5.59 0 0 0
*1961.89

 610 *569 0 *0 *0 0 6 MA *2 0 A G I Y N Y 53 L *792.53 641.39 641.39 08 1 H D N A A H 11 2 K 68
69.04 *1 1 201.71 3.14 0.14 7538.83 1.57 4746.83 1.00 0.83 0.43 28.67 0.00 0.00 0.01 0.57 89305.00
107200.00 0.00 0.00 0.00 0.86 46406 105 99 37 91 113 75 1.98 80 *M *38 *6 *130 *U *U *U *U *U *15
*S *3 *C *N *U 1111 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 10.29 *0.56 0 0 10.29 *2.6 1 1 1 *1961.89

 732 *610 0 *0 *1 0 8 MA *1 0 A E J Y N Y 41 B *456.3 495.29 503.34 04 U H C N B U U 01 0 O 85
69.04 *1 1 238.33 5.33 0.00 27519.33 2.00 32602.60 0.52 0.51 1.00 0.00 0.00 0.00 *0.01 1.33 145338.00
164333.33 0.67 0.00 0.33 1.33 70904 130 159 242 243 291 76 2.51 207 *U *66 *6 *0 *01 *A *U *U *1 *03
*S *9 *G *N *5 1011 13 15 22 3 7 9 13 0 2 3 6 0 0 13 7 1 0 0 0 *0.49 -2.66 -2.66 -0.59 -1.22 1.25 0 1
1 *1961.89
```

Figure 4.13 Filled Sample Data

- Build neural network model for evaluation: build Back-Propagation, training and testing models using software NeuralWare. Figure 4.14 shows the evaluation Back-

Propagation Model built by NeuralWare. All parameters in Back-propagation model are:

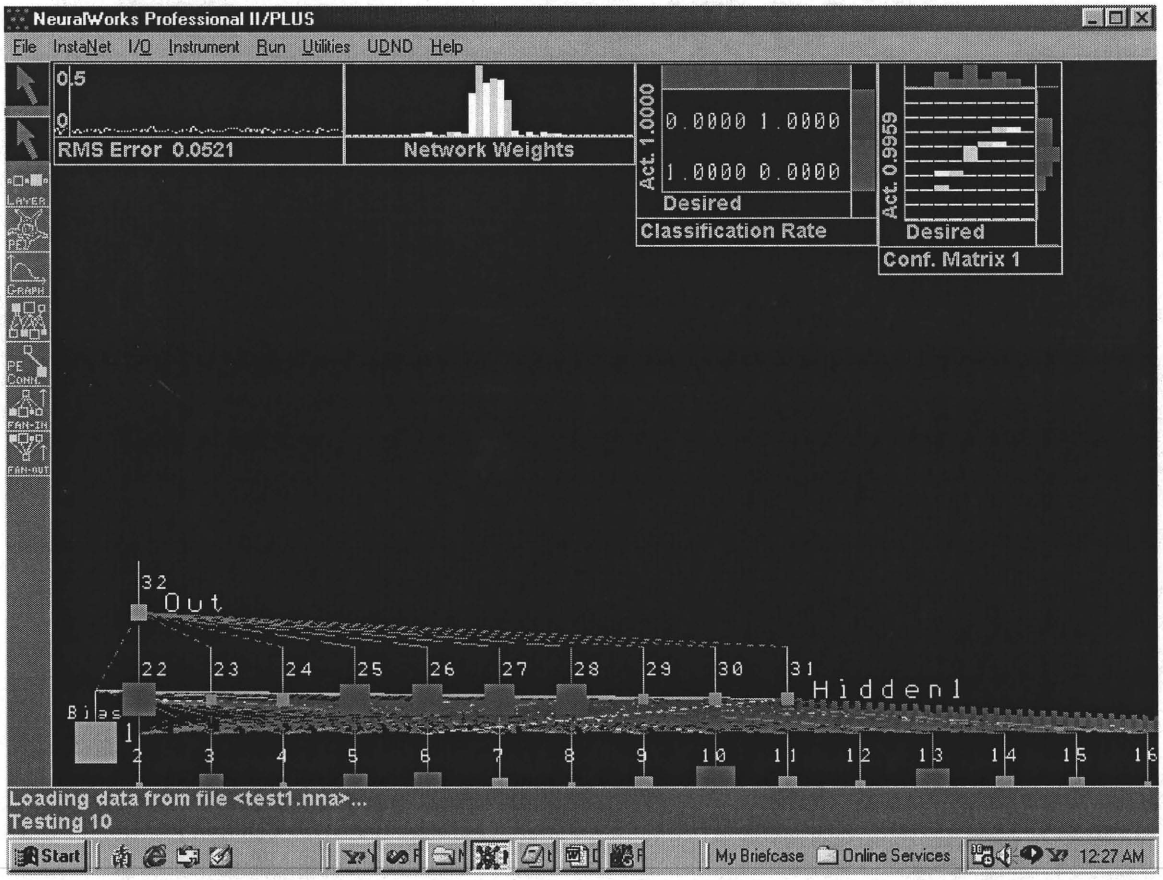
a) One hidden layer with 10 Process Elements;

b) Transfer function: TanH;

c) RMS = 0.02;

d) Learning rule: Norm-Cum-Delta.

e) Add Norm distribution noise into each input.



Loading data from file <test1.nna>...
Testing 10

Figure 4.14 Back-Propagation Model

- Analyze model's performance: analyze the performance using error range. The definition of accuracy is the percentage of predicating data, which fall into the "desired" values' error range. For example, if predict data is 9.7 and desired data is 10, it has an error range 0.05 that means the predict value within the range [10.5, 9.5] is regarded as correct. Of course, value 9.7 is correct. Experiments results based on above model are listed in Tables 4.4, 4.5, and 4.6.

Table 4.4 Learning set 10000 records

Accuracy Error range	Test1 (10 samples)	Test2 (100 samples)	Test3 (500samples)
0.05	90	90	94.4
0.03	70	78	81.6
0.1	100	99	99.8

Table 4.5 Learning set 5000 records

Accuracy Error range	Test1 (10 samples)	Test2 (100 samples)	Test3 (500samples)
0.05	90	88	90.4
0.03	60	69	71.4
0.1	100	98	99

Table 4.6 Learning set 1000 records

Accuracy Error range	Test1 (10 samples)	Test2 (100 samples)	Test3 (500samples)
0.05	70	73	84.4
0.03	40	62	58.8
0.1	100	97	98.8

To make the evaluation more accurate, two RMS evaluation matrixes are both used in this work. The procedures for the first matrix are: (1) for each record, determine its output category from its "desired" output value; (2) evaluate its predicted output value based on a specific RMS value; and (3) determine whether the predicted output value falls into the right category or not. The evaluation of the second matrix, called "reverse matrix", is to determine which output category the predicative value belongs to based on the predicative value and the specific RMS value, and compare this type with "desired" output type and determine whether it falls into the right category or not.

The next experiment generates 6600 training and 3300 testing records. Each record has 40 inputs and 5 output types. Each type has equal testing and training records. Results are shown in Tables 4.7 and 4.8.

Table 4.7 Accuracy matrix (RMS = 0.5)

	Class 0:	Class 1:	Class 2:	Class 3:	Class 4:	Unknown
Class 0:	**0.995**	0	0	0.005	0	0
Class 1:	0.003	**0.997**	0	0	0	0
Class 2:	0.005	0	**0.77**	0.08	0	0.144
Class 3:	0.003	0	0.09	**0.75**	0	0.15
Class 4:	0	0	0	0	**0.998**	0.002

Table 4.8 Reverse Accuracy matrix

	Class 0:	Class 1:	Class 2:	Class 3:	Class 4:
Class 0:	**0.989**	0 .003	0.005	0.003	0
Class 1:	0	**1**	0	0	0
Class 2:	0	0	**0.895**	0.105	0
Class 3:	0.005	0	0.096	**0.899**	0
Class 4:	0	0	0	0.0015	**0.998**
Unknown	0	0	0.482	0 .512	0 .005

In terms of error range, when error tolerance value is +/- 10%, neural network model can correctly predict over 95% target values. In terms of accuracy matrix, based on experiments, when RMS = 0.5, neural network can correctly predict at least 75% data. From the analysis, Back-Propagation neural network is proved effective on data prediction.

4.5 Summary

Following the data-mining framework in Chapter 3, this chapter first describes the implementation architecture with 4 modules, which are: Data Preprocessing Module, Missing Data Filling Module, Auxiliary Mathematics Tool Module and Neural network Performance Evaluation Module.

- Data Preprocessing Module: mainly deals with data format transformation, and data set features analysis etc.
- Missing Data Filling Module: determines the sample size and the default value, and fills all "holes" in the original data set through clustering, data cleansing and the second filling.
- Neural network Performance Evaluation Module: focuses on evaluating performance of supervised neural network model ---- Back-Propagation Model, which is proved to be effective on data prediction and classification.
- Auxiliary Mathematics Tool Module: a supporting model for the whole implementation, which implements some distribution function generators, sort method, K-S and Chi-Square algorithms etc.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

Data mining on financial field is a very promising research area and lots of efforts have been on it. Neural network, as a data-mining engine, is best applied to problem environments that are highly unstructured, require some form of pattern recognition, and may involve incomplete or corrupted data. After a review of recent methodology and technology development activities in data mining and neural networks areas, it is clear that neural network has many advantages over other methods in financial problems, such as fault tolerance, generalization and adaptivity. Credit card data mining problems include filling missing data and predicting target value etc. Based on this observation, this thesis first proposes a generic framework for filling missing data using neural network. Following this framework, two innovative algorithms are proposed to fill missing data. Finally, the detailed implementations and results are presented.

This work has the following contributions:

(1) The methodology provides a mechanism to fill missing data in credit card record sets using neural network. Two-level filling procedures guarantee all "holes" filled with more accurate and reliable data.

(2) Applications of the methodology finally get the complete clean, high quality data sets, which found a very solid foundation on credit card approval and fraud detection research.

(3) The evaluation for neural network Back-Propagation Model is performed, which provides a strong proof for its further use in target data prediction.

5.2 Future Research

This work also has some limitations, which need to solve in the further research.

- Detect inaccurate and inconsistent value. This work assumes all exist data are valid data in data cleansing step, which may not true. It is necessary to develop methods to exclude out-of-range data or erroneous data from data set and further improve the data set's quality.
- Determine attribute weight. In the real world, when an expertise makes' credit card approval decisions, he or she will not consider all fields in the application forms equally. In case of approval or not, applicants' revenue, credit history, occupation etc., are the most important fields, while gender or marital status etc., are relevant but less important, and name, social security number, telephone number etc., are irrelevant to decision-making. In this research, all attributes have the same importance. This may not fit the real situation. There is a need to design algorithm to evaluate the importance of each attribute and optimize the quality of filled data set.
- Make criteria to determine the necessity of further sub-cluster. This project only clusters sample data set once. From report, it shows clusters are denser than other clusters in other word; some clusters have record much more than others. It is necessary to make a guideline or rule to judge whether a further cluster is needed or not.

- Reduce the dimensionality of data set. All operations on data filling are based on all attributes, which is not reasonable. To solve this problem, determine attribute weight is a usual way, but not enough. Thus, the further study to find a useful method for eliminating some attributes deserves more effort.

REFERENCES

[1] Bengio Y., Buhmann, J. M. and Embrechts, M. J., “Introduction to the special issue on Neural Networks for data mining and knowledge discovery,” *IEEE Trans. On Neural Networks,* V. 11, No. 3, June 2000, pp. 545-548.

[2] Bigus, J. P., *Data Mining with Neural Networks*, McGraw-Hill, New York, 1996.

[3] Craven, M. and Shavlik, J., “Using neural networks for data mining,” *Feature Generation Computer Systems,* Vol. 13, Issue 2-3, Nov. 1997, pp. 211-229.

[4] Dash, M. and Liu, H., “Feature selection for classification,” *Intelligent Data Analysis,* Vol. 1, No. 3, 1997, pp. 131-156.

[5] Fu, Y. J., “Data mining,” *IEEE Potentials*, V.16, Issue 4, Oct.-Nov. 1997, pp. 18-20.

[6] Han, J. W., “Data Mining: Where Is It Heading?” *Proc. of 13th Int. Conf. on Data Engineering*, 1997, pp. 508-508.

[7] Kelly, J. D. and Davis, L., “Hybridizing the genetic algorithm and the K-nearest neighbors classification algorithm”, *Proc. of the fourth Int. Conf. on Genetic Algorithms and their Applications,* 1991, pp. 377-383.

[8] Kleissner, C., “Data mining for the enterprise,” *Proc. of the 31st Hawaii Int. Conf. on System Science,* V.7, 1998, pp. 295-304.

[9] Kohonen, T, “Self-organized formation of topologically correct feature maps,” *Biological Cybernetics,* Vol. 43, 1982, pp. 59-69.

[10] Kohonen, T, “The self-organized maps,” *Proc. of the IEEE,* Vol. 78, No. 9, 1990, pp. 1464-1480.

[11] Langley, P. and Blum, A. L., "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, Vol. 97, No. 1-2, Dec. 1997, pp. 245-271.

[12] Lu, H. J., Setiono, R. and Liu, H., "Effective data mining using neural networks," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 8, Issue 6, Dec. 1996, pp. 957-961.

[13] Martin-Bautista, M. J. and Vila, M. A., "A survey of genetic feature selection in mining issues," *Proc. of the 1999 Congress on* Evolutionary Computation, Vol. 2, 1999, pp. 1999-1321.

[14] Medsker, L., Trippi, R. R. and Turban, E., *Expert Systems and Applied Artificial Intelligence,* Macmillan Publishing Co., New York, 1992.

[15] Olaru, C. and Wehenkel, L., "Data mining," *IEEE Computer Applications in Power*, Vol. 12, Issue 3, July 1999, pp. 19-25.

[16] Pei, M., Goodman, E. D. and Punch, W. F., "Pattern discovery from data using genetic algorithms," *Proc. of the First Pacific-Asia Conf. on Knowledge Discovery and Data Mining,* 1997.

[17] Punch, W. F., Goodman, E. D., Pei, M., Chia-Sun, L., Hovland, P. and Enbody, R., "Further research on feature selection and classification using genetic algorithms," *Proc. of the Fifth Int. Conf. on Genetic Algorithms and their Applications,* San Mateo, CA, 1993, pp. 557-564.

[18] Stebbins, G., "Seeking signals in data bases using neural networks," *Proc. of the Fifth Int. Symp. on Signal Processing and Its Applications,* Vol. 1, 1999, pp. 167-170.

[19] Sung, S. Y., Wang, K. and Chua, K. L., "Data mining in a large database environment," *Proc. of IEEE Int. Conf. on System, Man and Cybernetics,* Vol. 2, 1996, pp. 988-993.

[20] Thuraisingham, B., "A primer for understanding and applying data mining," *IT Professional*, V.2, Issue 1, Jan.-Feb. 2000, pp. 28-31.

[21] Trippi, R. R. and Turban, E., *Neural Networks in Finance and Investing,* Irwin Professional Publishing Co., Chicago, 1996.

[22] Vafaie, H. and De J. K., "Genetic algorithms as a tool for feature selection in machine learning," *Proc. of the 4th Int. Conf. on Tools with Artificial Intelligence,* Arlington, VA, Nov. 1992, pp. 200-204.

[23] Vafaie, H. and De J. K., "Genetic algorithms as a tool for restructuring feature space representation," *Proc. of the 7th Int. Conf. on Tools with Artificial Intelligence,* Herndon, VA, Nov. 5-8 1995, pp. 8-11.

[24] Yang, J and Honavar, V., "Feature subset selection using a genetic algorithm," In Liu, H. and Motoda, H. (Eds.), *Feature Extraction Construction and Selection: A Data Mining Perspective,* Kluwer Academic Publishers, Massachusetts, 1998.