ABSTRAK

Kegiatan penipuan telah mencapai ke peringkat kritikal dimana ianya menyebabkan syarikat telekomunikasi mengalami kerugian berjumlah jutaan ringgit dan memaksa syarikat-syarikat tersebut mengguna pakai aplikasi atau sistem (seperti Telekom Malaysia Berhad Sistem Pengenalpastian Kegiatan Penipuan Generasi Baru) untuk mengenalpasti kegiatan tersebut.

Kami memperkenalkan satu algoritma baru yang dapat mengenalpasti kegiatan penipuan dalam industri telekomunikasi (sebagai contoh, pencerobohan penipuan berlaku apabila akaun yang sah diancam oleh penceroboh yang membuat dan menjual panggilan dengan menggunakan akaun tersebut) yang menggunakan Model Bercampur Gauss, satu model kebarangkalian yang seringkali digunakan dalam mengenalpasti kegiatan kecurian melalui pengenalpastian suara. Disebabkan kerumitan yang dipamerkan oleh Model Bercampur Gauss, kami menggunakan Pemaksimum Jangkaan oleh Dempster et al. (1977) untuk mencari Penganggaran Kebolehjadian Maksimum bagi parameter Model Bercampur Gauss. Bersama-sama dengan kaedah inti (sila rujuk kepada Silverman, 1986), kami dapat memperbaiki proses yang berkaitan dengan menentukan bilangan komponen dalam Model Bercampur Gauss. Tambahan pula, kami berjaya menghasilkan Ujian Kebolehjadian Nisbah dalam menentukan bilangan komponen dalam Model Bercampur Gauss dan perbandingan keputusan yang diperolehinya dengan keputusan yang diperolehi oleh Kriteria Maklumat Akaike akan ditekankan dalam tesis ini. Algoritma tersebut juga menggunakan pekali keserupaan untuk mengklasifikasi data sebenar berdasarkan kepada fungsi log-kebolehjadian dan

ianya diperluaskan untuk mengenalpasti panggilan yang mencurigakan yang dikenalpasti oleh syarikat telekomunikasi.

Algoritma yang baru diperkenalkan ini diuji dengan menggunakan data simulasi dan data sebenar dimana keputusannya (daripada ujian tersebut) menunjukkan ia berupaya mengenal pasti kegiatan penipuan. Data sebenar adalah terdiri daripada berapa lama panggilan dibuat dan berapa cajnya, diambil daripada ibusawat Telekom Malaysia Berhad, dan ianya dicemari oleh aktiviti penipuan. Memandangkan format data sebenar adalah berbeza dengan yang biasa digunakan untuk pengenalpastian suara, ianya disusun semula sebelum pengujian dan penganalisaan. Algoritma baru bersetuju dengan apa yang dicurigai oleh syarikat

ABSTRACT

Fraud activities have reached to critical point causing millions of ringgit of losses to telecommunication companies, and as a result, forcing them to employ applications or systems (such as Telekom Malaysia Berhad's Next Generation Fraud Detection System) to detect the said activities.

We introduce a new algorithm that could detect fraud activities in telecommunication industry (e.g. intrusion fraud which occurs when legitimate account is comprised by an intruder who makes or sells calls on this account) that uses Gaussian Mixed Model (or GMM), a probabilistic model normally used in fraud detection via speech recognition. Due to the complexity of GMM, we use Expectation Maximization (or EM) algorithm by Dempster et al. (1977) to obtain the maximum likelihood estimates of the GMM parameters. Together with Kernel method (see Silverman, 1986), we improve the process of finding the number of components in GMM. In addition, we have also successfully derived the likelihood ratio test in the determination of the number of components in GMM and the comparison of its results with those of Akaike Information Criteria (AIC) will also be highlighted in this thesis. The said algorithm uses similarity coefficient to classify the real data based on the log-likelihood function and it's extended to detect incoming fraud calls as suspected by the telecommunication company.

The new algorithm is tested on simulated and real data where the results show it is capable of detecting fraud activities. The real data, which included call charging and duration, are collected from Telekom Malaysia Berhad's exchanges and they are

believed to be contaminated by fraud activities. As the original data are clearly not in the format that is generally used for speech recognition, they are reformatted prior to testing and analysis. The new algorithm is in agreement with those suspected by the company.

ACKNOWLEDGEMENTS

First and foremost, I thank my wife, Mastura Abu Bakar, for her constant love, support and encouragement in this time-consuming and difficult work. I thank my supervisors, Associate Professor Dr Ibrahim Mohamed and Associate Professor Dr Mohd Rizam Abu Bakar, for their advice on the content of this thesis, for their vigilance and energy in reading with great care the drafts of this work, and for their valuable suggestions. Special thanks to Head of Institute of Mathematical Sciences who made it possible for me to carry out my post graduate studies; post graduate students (in particular, Adzhar Rambli, Kamil Khalid, Safwati Ibrahim, Mardziah Nawama, Nurul Najihah) and staff of Institute of Mathematical Sciences (in particular, Budiyah Yeop) for making my stay a pleasant and memorable one.

I thank Telekom Research & Development Sdn Bhd for sponsoring my post graduate studies; Mohd Shafri Kamaruddin and Mohd Daud Jaafar of Telekom Research & Development Sdn Bhd for their cooperation in providing data and information about fraud activities in telecommunication industry.

Finally, I dedicate this thesis to my wife, children, and parents for without them none of this would be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Background of the study

Telecommunication companies (including those that are operating in Malaysia) suffered heavy losses amounting to US$55 Billion per year due to fraud activities. Between 3 and 5 percent of the company's annual revenue would "disappear" due to the said activities. They (i.e. the said companies) will not declare or make public these activities (if they do decide to declare, they will call these activities as bad debt) fearing huge migration of customers to the competitors due to lack of confidence in the services they offered. The fraud activities are expected to increase with the introduction of new services such as 3G and Voice Over Internet Protocol.

Although the number of fraudulent calls is small when compared to the overall call volume but the cost incurred is huge (or significant) amounting to, based on estimation by analysts for telecommunication industry in U.S., $1 Billion a year as mentioned by Cox et al. (1997). Cahill et al. (2002) reported fraud activities eroded between 4% and 6% of U.S. telecom's revenue; and suggested that the degree of "erosion" is much worse at international level where several new service providers reported losses greater than 20%. In terms of losses to fraud, Moreau et al. (1996) estimated several millions European Currency Units (ECUs) per year. Bolton and Hand (2002) gave figures representing losses to fraud each year of £13 Billion and $13

Billion (U.S.); and estimated it could reach $28 Billion within 3 years. Generally, the loss is significantly large and warrants serious action to manage the problem.

Becker et al. (2010) mentioned the Communications Fraud Control Association (cfca.org) periodically estimates the extent of worldwide telecommunications fraud. In 1999 this estimate was $12 billion, in 2003 it was between $35 and $ 40 billion, in 2006 it was between $55 and $60 billion, and in 2009 it was between $70 and $78 billion.

1.1.1 Modus Operandi

Becker et al. (2010) gave examples of some common varieties of fraud in telecommunication and the one that is of our interest is intrusion fraud. This occurs when an existing, otherwise legitimate account, typically a business, is compromised in some way by an intruder, who subsequently makes or sells calls on this account. In contrast to subscription calls, the legitimate calls may be interspersed with fraudulent calls, calling for an anomaly detection algorithm.

For mobile telecommunication, the perpetrator would normally hack into a network and use false identity to access services for free. Nowadays the perpetrator would use cloned phones (or SIM) to make free international and expensive roaming calls (see Bihina Bella et al., 2005). This activity would involve the duplication of customer's hardware and firmware thus allowing the perpetrators to make calls on their account and consequently inflating their monthly bill. Other types of fraud are prepaid and interconnect.

Premium Rate Service (PRS) involves high number of calls made to the PRS number from customer's line network without their knowledge or from a number where there is no intention to pay for the outgoing calls using auto-dialers. Other types of fraud are Private Automatic Branch Exchange (PABX) for international calls, network's personnel provide an assigned number to a user that does not have an account with the network (this activity is called stolen line unknown) and international roaming manipulation which is similar to subscription fraud where the perpetrator exploits the time delay of high rate identification and notification to home network when roaming on another network.

There are cases where perpetrators are the subscribers themselves that performed call back (with the intention to get cheaper international calls from call back operator usually in another country via dialing out, regular dial tone availability, call booked via other channels, and call initiated via international toll free number) and illegal schemes (e.g. reselling calling cards to other subscribers, forgery of vouchers, recharge, counterfeit and stolen cards to pay goods over the phone).

Several types of telecom fraud are listed in Shawe-Taylor et al. (2000), but the one, which is of our interest, is called superimposed or "surfing" fraud. Superimposed fraud refers to the use of a service without permission and it would appear as phantom calls on a bill. It will generally occur at the level of individual calls where fraudulent will be mixed together with the legitimate calls.

1.1.2 Methods used to detect fraud activities

Data Mining is one of the applications capable of detecting fraud in telecommunication and Malaysian companies (especially telecommunication ones) are using the said application. IBM developed Data-Mining application called "Intelligent Miner" (amongst its users are Australia's Health Insurance, John Hancock Life Insurance, "Saveway" Mart in UK and banking industry in USA) whereas SAS "Enterprise Miner", SPSS, Lotus, SGI and Hitachi are not far behind in the "race" (in developing the said application). In Malaysia, companies such as Aetna Insurance, Astro, Celcom and Franks & Small have been using Data-Mining application since 1996 and SAS has developed Data-Mining application called SISWANG to manage corporate type data for Telekom Malaysia Berhad.

Briefly, techniques used in Data-Mining can be divided into two: (1) Preparation of Data and (2) Model of Data. Techniques (1) and (2) are equally important where according to Pyle (1999), the data must be well managed so that process with regard to modeling could be performed smoothly and quickly. Zuber et al. (2013) on the other hand concentrated on (2) and gave six models used in Data-Mining: (1) Classification, (2) Regression, (3) Time Series, (4) Cluster, (5) Association Analysis and (6) Sequence Discovery.

SAS has developed a data mining analysis cycle known by the acronym SEMMA (Rohanizadeh and Moghadam, 2009). This acronym stands for the five steps of an analysis that are ordinarily a part of a data mining projects. Those five steps are:

Sample: First step of data mining is to create one or more data tables by sampling data from the data warehouse. The samples should be big enough to contain significant information, yet small enough to process quickly hence reducing the processing time required to get critical business information. This approach uses a reliable, statistical representative sample of the entire database.

Explore: After sampling the data, they would be explored visually and numerically (i.e. using statistical techniques including Factor Analysis, Corresponding Analysis and Clustering) for inherent trends or groupings.

Modify: Based on the discoveries in the exploration phase, modification may be needed:

(1) To include information such as grouping of customers and significant subgroups, or

(2) To introduce new variables such as ratio obtained by comparing two previously defined variables.

Modification process also involves looking for outliers, reduce the number of variables to narrow them down to the most significant ones and modify data when previously mined data change in some way.

Model: After the data have been assessed and modified, data modeling techniques (e.g. neural networks, tree-based models, logistic models, and other statistical models such as time series analysis and survival analysis) are used to construct models that explain "pattern" in the data and each of them (i.e. data modeling techniques) is appropriate within specific data-mining situations depending on the data.

Assess: Assessing a model to determine how well it performs is done by applying it to a portion of the data that was set-aside during the sampling stage. If it is valid, it should work for this reserved sample as well as the sample used to construct the model.

Abbot et al. (1998) highlighted the findings of a study done by DataQuest back in 1997 (more and better Data Mining applications have started to emerge nowadays with the advent of more "powerful" micro-processors) where IBM was the data mining software market leader with a 15% share of license revenue, Information Discovery was second with 10%, Unica was third with 9% and Silicon Graphics was fourth with 6%. They went even further by evaluating data mining products or tools (vendor is given in brackets and will be used henceforth to describe the product) namely Clementine (ISL); Darwin (TMC); Enterprise Miner (SAS); Intelligent Miner for Data (IBM); and Pattern Recognition Workbench (Unica) based on the following factors:

Client Server Processing: Data mining applications often use data sets far too large to be retained in physical RAM, slowing down processing considerably as data loaded to and from virtual memory. Also, algorithm runs far slower when hundreds of candidate inputs are considered in models. Therefore a client server processing model has great appeal by using a single high powered workstation for processing but let multiple analysts access the tools from PCs on their desktops.

Automation and Project Documentation: The experimentation process involves repeatedly adjusting algorithm parameters, candidate inputs and sample sets of the training data. It would be a great help to automate what can be in this process in order to free the analysts from some of the mundane and error prone tasks of linking and documenting exploratory research findings. All five products provided means to

document findings during the research process, including time and date stamps on models, text fields to hold notes about the particular model and the saving of guiding parameters.

Algorithms: Referring to Decision Trees, Neural Networks, Regression, Radical Basic Functions, Nearest Neighbor, Nearest Mean Kohonen Self Organizing Maps, Clustering and Associate Rules.

Ease of Use: Referring to Data Loading and Manipulation, Model Building and Understanding (Specifying Models, Reviewing Trees and Reviewing Classification Results) and Technical Support.

Accuracy: The smaller number of false alarms is better and the larger number of fraudulent activities caught is better. Data used to grade accuracy of the tools contained fraudulent and non-fraudulent financial transactions.

Abbot et al. (1998) found ISL's performance on modem line was acceptably slow. Unica's processor capabilities must be significantly better than is required for the others. IBM's Java runs more slowly than other GUI designs. SAS has the largest disk footprint of any of the tools (i.e. at 300+ MB). Unica doesn't have Decision Trees (this study focused on Decision Trees and Neural Networks). They (i.e. Abbot et al., 1998) concluded that IBM's Intelligent Miner for Data has the advantage of being the current market leader with a strong vendor offering well-regarded consulting support. ISL's Clementine excels in support provided and in ease of use (given Unix familiarity) and might allow the model iterations in a tight deadline. SAS's Enterprise Miner would especially enhance a statistical environment where users are familiar with SAS and

could exploit its macros. Thinking Machine's Darwin is the best when network bandwidth is at a premium (say, on very large databases). Unica's Pattern Recognition Work-bench is a strong choice when it's obvious what algorithm will be most appropriate, or when analysts are more familiar with spreadsheets than Unix.

The detection and analysis of outliers become difficult when the data involved is:

- Time series data because they (i.e. outliers) may be hidden in trend, seasonal or other cyclic changes,
- Multidimensional data where not any particular one but rather a combination of dimension values may be extreme and
- Non-numeric (i.e. categorical data) where the definition of outlier requires special consideration.

Methods for detecting them (i.e. outliers) are (as per listed by Han and Kamber, 2001):

Online Analytic Processing (OLAP) uses data cubes to identify regions of anomalies in large multidimensional data. For example, *discovery driven exploration* is an approach where pre-computed measures indicating data exceptions are used to guide the user in data analysis at all levels of aggregation. A cell value in the cube is considered an exception if it is significantly different from the expected value based on a statistical model. The method uses visual cues such as background color to reflect the degree of exception of each cell. The user can choose to drill down on cells that are flagged as exceptions. The measure value of a cell may reflect exceptions occurring at more detailed or lower levels of the cube where these exceptions are not visible from the current level.

Deviation based Outlier Detection identifies outliers by examining the main characteristics objects in a group. No statistical technique (or distance based measures) is used. For example, Dissimilarity function does not require a metric distance between the objects. It is any function that, if given a set of objects, returns a low value if the objects are similar to one another. The greater the dissimilarity among the objects, the higher the value returned by the function. The dissimilarity of a subset is incrementally computed based on the subset prior to it in the sequence. Given a subset of n numbers $\{x_1,\ldots,x_n\}$ a possible dissimilarity function is the variance of the numbers in the set that is $\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2$ where $\bar{x}$ is the mean of the n numbers in the set. For character strings, the dissimilarity function may be in the form of a pattern string (e.g. containing wildcard characters) that is used to cover all the patterns seen so far. The dissimilarity increases when the pattern covering all of the strings in S_{j-1} doesn't cover any string in *Sj* that is not in S_{j-1}.

Distance based outlier detection: Objects that do not have enough neighbors, where neighbors are defined based on distance from the given object. For example, given a data set, the index-based algorithm uses multi-dimensioning indexing structures, such as R-trees, to search for neighbors of each object *o* within radius d-around that object. Let *M* be the maximum number of objects within the d-neighborhood of an outlier. Therefore, once *M+1* neighbors of object *o* are found, it is clear that *o* is not an outlier.

Statistical approach: Assuming a distribution or probability model for the given data set (e.g. normal distribution) and the identification of outliers with respect to the model using discordance test. Other examples are as follows:

- Hollmen and Tresp (2000) presented a Learning Vector Quantization (LVQ) algorithm for learning a classifier defined by a codebook of probabilistic models. The models implicitly define a discrimination function in the input data space through maximum likelihood search. The prototypical codebook vectors were replaced by generative, probabilistic models and the LVQ learning rules were modified accordingly. The likelihood-based distance was justified by a derivation form the Kullback-Leibler distance. The conceptual difference to conventional training of probabilistic models is the use of supervised, gradient-based learning instead of maximum likelihood estimation. This specially tunes the models for discrimination. The algorithm may also be used in post-processing to enhance the discriminative aspect of generative density models earlier trained by using the EM algorithm.

- Hollmen and Tresp (2000) extended the Hidden Markov Model (or HMM) to modeling time series that exhibit switching between matrix and event based representations. This essentially combines an HMM with continuous emission distribution and one with discrete emission distribution. Additional variable data semantics controls the interpretation of data and is dependent on the hidden variable. Inference and learning rules were developed within a maximum likelihood framework. The approach was illustrated in a user profiling problem where the mechanism leading to the event representation was important from user profiling point of view.

- Taniguchi et al. (1998) presented three approaches to fraud detection in communications networks. They are Neural networks with supervised learning, Probability density estimation methods and Bayesian networks. The

performance of these methods has been validated with data from a real mobile communication networks. The feature vectors used in this application describing the subscriber's behavior were based on toll tickets. For supervised learning approach, the features used were summary statistics over the whole observed time period as no times of fraud were recorded in the data. For the two latter approaches, the features described the daily behavior for every subscriber. To improve the fraud detection system, the combination of the three presented methods could be beneficial. Furthermore, the incorporation of rule-based systems could show an improvement.

- Hollmen and Tresp (1998) presented a call based on line fraud detection system which is based on a hierarchical regime switching generative model. The inference rules are obtained from the junction tree algorithm for the underlying graphical model. The model is trained by using the EM algorithm on an incomplete data set and is further refined with gradient based discriminative training, which considerably improves the results.

- Linoff (2004) has successfully used survival data mining (especially Kaplan-Meier Survival Analysis) to understand customer behavior or churn such as plots produced from using hazards formula (where in this case $\frac{\text{number of customers who stopped with exactly tenure t}}{\text{everyone who had tenures greater than or equal to t}}$) show peaks of non-payment and promotion ends and from using the survival formula (i.e. cumulative probability by multiplying one minus the hazards together for all values less than t) shows the number of customers that will survive beyond the non-payment period. The said data mining would "censor" customers that leave voluntarily and may be able to answer questions such as: "When will a lapsed

customer return?", "When is the next customer's purchase?" and "How long will an upgrade last?".

Xing and Girolami (2007) employ Latent Dirichlet Allocation (LDA) to build user profile signatures. The authors assume that any significant unexplainable deviation from the normal activity of an individual user is strongly correlated with fraudulent activity. A straightforward generalization of LDA to time-invariant Markov chains of arbitrary order is proposed in Girolami and Kaban (2005), where the experimental study refers to modeling the sequential usage of a telephone service by a large group of individuals. Xu et al. (2008) presents a novel rough fuzzy set based approach to detect fraud in 3G mobile telecommunication network. It analyzes the scenarios in 3G network including subscription fraud and superimposed fraud and profile and confirms the parameters to detect the scenarios. Hilas and Mastorocostas (2008) investigates the usefulness of applying different learning approaches to a problem of telecommunications fraud detection that is by applying multilayer perception classifier and the hierarchical agglomerative clustering technique on five models (profiles) of telecommunications users' behaviors. Hilas (2009) constructs an expert system, which incorporates both the network administrator's expert knowledge and knowledge derived from the application of data mining techniques on real-world data. The detection of individual fraud call which are of the time series type become tricky as they may be hidden in trend, seasonal, or other cyclic changes. The problem becomes more complicated when multidimensional data are considered. Such problem may be classified as the problem of detecting outliers.

Gomez-Restrepo and Cogollo-Florez (2012) evaluate the implementation of generalized linear mixed models to detect fraud. They consider the heterogeneity of

customers and generate not only a global model, but also a model for each customer which describes the behavior of each one according to their transactional history and previously detected fraudulent transactions. In particular, a mixed logistic model is used to estimate the probability that a transaction is fraudulent, using information that has been taken by the banking systems in different moments of time.

1.2 Statement of the problem

The number of literatures that discuss about pattern recognition method (namely Gaussian Mixed Model, GMM) used to detect fraud activities in telecommunication industry involving real data other than speech recognition's format is limited and GMM is considered as difficult in reality because we need to find the initial estimates of parameters to start Expectation Maximization (EM) algorithm and the exact number of components. Telekom Malaysia Berhad, a leading telecommunication company in Malaysia, via their current system or application believes the real data collected by them (e.g. duration and charging or billing) from its exchanges are contaminated by fraud activities and, since GMM is not included on the list of methods, there is no knowing if their findings are statistically correct.

1.3 Objectives

Based on the statement of the problem given above, we have outlined the following objectives for this study:

• To improve Gaussian Mixed Model (GMM) from its known (or current) weaknesses (or difficulties) such as finding the initial estimates of parameters to start Expectation Maximization (EM) algorithm and finding exact number of components.

• To introduce a new algorithm that is capable of detecting fraud activities (especially) in telecommunication industry and that incorporates the improvement as mentioned in the first bullet.

• To test the new algorithm (at the same time improving the EM algorithm for GMM) using simulation data and real data (e.g. duration and charging or billing) collected from Telekom Malaysia Berhad's exchanges that are believed to be contaminated by fraud activities.

1.4 Significance of the study

In addition to contributing to the knowledge in statistics, the findings from this study will encourage the use of statistical methods (in this case, Gaussian Mixed Model) in detecting fraud activities in telecommunication industry by incorporating them into the company's decision support system.

1.5 Thesis outline

This research attempts to detect fraud in telecommunication using pattern recognition method and it is outlined as follows:

Chapter two provides a literature review about the use of pattern recognition method in detecting fraud activities in Telecommunication industry where special attention is given to Gaussian Mixed Model (GMM). Formulas involved in GMM and Expectation Maximization, an algorithm typically used in solving the problem of calculating maximum likelihood estimation, are listed and derived.

Chapter three attempts to use Gaussian mixed model which is a probabilistic model normally used in speech recognition to identify fraud calls in the telecommunication industry. We look at several issues encountered when calculating the maximum likelihood estimates of the Gaussian mixed model using an expectation maximization algorithm. Firstly, we look at a mechanism for the determination of the initial number of Gaussian components and the choice of the initial values of the algorithm using the kernel method. We show via simulation that the technique improves the performance of the algorithm via simulation. Secondly, we develop a procedure for determining the order of the Gaussian mixed model using the log-likelihood function and the Akaike information criteria (AIC). Finally, for illustration, we apply the improved algorithm to real telecommunication data. The modified method will pave the way to introducing a comprehensive method for detecting fraud calls in future work.

Chapter four proposes a new fraud detection algorithm that uses Gaussian mixed model, a probabilistic model normally used in recognizing a person's voice in speech recognition field. Using data obtained from one of the leading telecommunication company in Malaysia, we show that the proposed algorithm has not only successfully detected fraud calls as suspected by the company, but also identify suspicious calls which can be candidates of fraud call. The proposed algorithm is easy to implement

with a great potential to be extended to detect billed (or outgoing) fraud calls and hence reduces the loss incurred by the telecommunication companies.

Chapter five shows the successful derivation of hypothesis testing in the determination of the number of components in GMM, which is an important process as highlighted by a number of authors. The performance of the hypothesis testing and the comparison of its results with those of AIC will also be highlighted in this chapter.

Chapter six proposes a new algorithm than can be efficiently used to identify fraud activities. The algorithm is developed by finding the characteristics of historical fraud and non-fraud calls and is consequently used in identifying possible fraud call instantly for immediate call verification process. Using data obtained from one of the leading telecommunication company in Malaysia, we show that the proposed algorithm has successfully detected outgoing fraud calls as suspected by the company.

Chapter seven presents the general conclusion and highlights the significant contributions of this research, moreover, we also suggest several possibilities for extending research work on fraud detection in telecommunication using pattern recognition method.

Appendices A and B present the programming language (and software) used to produce the results in this thesis and a sample of real data supplied by Telekom Malaysia Berhad, respectively.

CHAPTER 2

GAUSSIAN MIXED MODEL

2.1 Introduction

Jain et al. (2000) defines a pattern “as opposite as a chaos; it is an entity, vaguely defined, that could be given a name”. For example, pattern could be a fingerprint image, a handwritten cursive word, a human face, or a speech signal. They added given a pattern, its recognition/classification may consist of one of the following tasks: 1) supervised classification (e.g. discriminate analysis) in which the input pattern is identified as a member of predefined class, 2) unsupervised classification (e.g. clustering) in which the pattern is assigned to hitherto unknown class. They noted that the recognition problem here is being posed as a classification or categorization task, where the classes are either defined by the system designer (in supervised classification) or are learned based on the similarity patterns (in unsupervised classification).

Reynolds (1995) presented an overview of his research efforts in automatic speaker recognition. He based his approach on a statistical speaker-modeling technique that represents the underlying characteristic sounds of a person's voice. Using the said technique, he built speaker recognizers that are computationally inexpensive and capable of recognizing a speaker regardless of what is being said. Performance of the systems is evaluated for a wide range of speech quality; from clean speech to telephone speech, by using several standard speech corpora. Reynolds and Rose (1995)

introduced the use of Gaussian Mixed Model (GMM) for robust text-independent speaker identification. The focus of their work is on applications which require high identification rates using short utterance from unconstrained conversational speech and robustness to degradations produced by transmission over a telephone channel.

The function of GMM is extended to detect fraud activities on the number (as well as length) of domestic and international calls made on a daily basis during office, evening and night hours. Tanigushi et al. (1998) presented three approaches to fraud detection in communication networks: neural networks with supervised learning, probability density estimation methods and Bayesian networks. Information describing a subscriber's behavior kept in toll tickets was used. For example, supervised learning used summary statistics over the whole observed time period (especially the number of times fraud activities were recorded in the data). The two latter approaches used a subscriber's daily behavior. To improve the fraud detection system, they recommended the combination of the three presented methods together with the incorporation of rule-based systems.

The maximum likelihood estimation for a GMM is generally difficult to obtain directly, but it is made easier with the availability of the Expectation Maximization (EM) algorithm which was first introduced by Dempster et al. (1977). Since then, there has been a significant increase of its use especially in finding the maximum likelihood for probabilistic models. For example, Hollmen and Tresp (1998, 2000) developed an online system for detecting fraud calls using a hierarchical switching generative model. The model is trained by using the EM algorithm on an incomplete data set and is further improved by using a gradient-based discriminative method. Redner and Walker (1984) discussed the formulation as well as the theoretical and practical properties of

the EM algorithm for mixture densities, focusing in particular on mixtures of densities from exponential families. Xu and Jordan (1996) built up the mathematical connection between EM algorithm and gradient based approaches for maximum likelihood learning of finite Gaussian mixtures.

2.2 Gaussian Mixed Model

Let $\mathbf{x} \in R^d$ and K be the number of components where each component has its own prior probability a_i and probability density function with mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i, i = 1,\ldots K.$ A Gaussian mixed model is then given by

$$\sum_{i=1}^{K} a_i \phi(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \sum_{i=1}^{K} a_i \frac{1}{\sqrt{(2\pi)^d \mid \boldsymbol{\Sigma}_i \mid}} \exp\left(\frac{-(\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)}{2} \right) \tag{2.1}$$

where $\sum_{i=1}^{K} a_i = 1$. We next define the likelihood function and the log-likelihood function by $L(\mathbf{X} \mid \theta) = \prod_{j=1}^{n} f(\mathbf{x}_j \mid \theta)$ and $l(\mathbf{X} \mid \theta) = \sum_{j=1}^{n} \log\left(\sum_{i=1}^{K} a_i \phi(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right)$ where $\mathbf{X} = \left(\mathbf{x}_1^t, \ldots, \mathbf{x}_n^t\right)^t$ respectively. The maximum likelihood estimation (m.l.e) method aims at finding $\hat{\theta}$ that maximizes $l(\mathbf{X} \mid \theta)$, see Mardia et al. (1979). The expression $\log\left(\sum_{i=1}^{K} a_i \phi(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right)$ in $l(\mathbf{X} \mid \theta)$ is difficult to compute. We use the Expectation Maximization (EM) algorithm to overcome this problem.

2.2.1 Expectation Maximization algorithm

In a general set-up of the EM algorithm given in Dempster et al. (1977), the authors considered an unobservable variable X in sample space $\mathcal{X}$, which is indirectly observed through observed variable Y in sample space $\mathcal{Y}$. Assuming that $f(x \mid \theta)$ is the sampling density depending on the parameter $\theta \in \Omega$, the corresponding family of sampling densities for Y, say $g(y \mid \theta)$, can be derived from

$$g(y \mid \theta) = \int_{\mathcal{X}(y)} f(x \mid \theta) dx \tag{2.2}$$

where $\mathcal{X}(y)$ is a subset of $\mathcal{X}$ under the mapping $x \rightarrow y(x)$ from $\mathcal{X}$ to $\mathcal{Y}$. The main objective of the EM algorithm is to find the value of θ that maximizes equation (2.2). Consider the expected value of $\log f(x \mid \theta')$ given y and θ, denoted by $Q(\theta' \mid \theta)$, where

$$Q(\theta' \mid \theta) = E(\log f(x \mid \theta') \mid y, \theta) \tag{2.3}$$

with the expectation assumed to exist for all pairs (θ', θ) and $f(x \mid \theta) > 0$ for $\theta \in \Omega$. According to Dempster et al. (1977), the EM iteration consists of two steps namely the E-step and the M-step. At the pth iteration with the estimate of θ denoted by $\theta^{(p)}$, the E-step will give the value of $Q(\theta \mid \theta^{(p)})$ and the M-step will find a new estimate of θ, say $\theta^{(p+1)}$, that maximizes $Q(\theta \mid \theta^{(p)})$. The steps are repeated until convergence is achieved.

For the case of a GMM, we define $Q(\theta'|\theta) = E\left[\log \prod_{i=1}^{n} a_{y_i}^{'} \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}_{y_i}^{'}, \boldsymbol{\Sigma}_{y_i}^{'}\right) \middle| \mathbf{X}, \theta \right]$

where $y_i \in \{1,2,\ldots,K\}$ and $y_i = k$ if the i^{th} sample is generated by the k^{th} mixture component. It is simplified by applying, amongst others, the Bayes formula $f(\theta \mid x) \propto f(x \mid \theta) P(\theta)$ where $f(\theta \mid x)$ is the posterior probability, $f(x \mid \theta)$ is the likelihood function and $P(\theta)$ is the prior probability to the following equations (see Bilmes, 1998, and Tsay, 2005):

$$Q(\theta'|\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} p_{i,k} \log a_k^{'} + \sum_{i=1}^{n}\sum_{k=1}^{K} p_{i,k} \log \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{'}, \boldsymbol{\Sigma}_k^{'}\right) \tag{2.4}$$

where

$$p_{i,k} = \frac{a_k^{'} \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{'}, \boldsymbol{\Sigma}_k^{'}\right)}{\sum_l a_l^{'} \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}_l^{'}, \boldsymbol{\Sigma}_l^{'}\right)} \text{ and} \tag{2.5}$$

$$\phi(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{'}, \boldsymbol{\Sigma}_k^{'}) = \frac{1}{\sqrt{(2\pi)^d \mid \boldsymbol{\Sigma}_k^{'} \mid}} \exp\left(\frac{-(\mathbf{x}_i - \boldsymbol{\mu}_k^{'})^t \boldsymbol{\Sigma}_k^{'\,-1} (\mathbf{x}_i - \boldsymbol{\mu}_k^{'})}{2} \right) \tag{2.6}$$

Hence, the EM iteration for a GMM is defined by:

E-Step: Use equation (2.5).

M-Step: Use the formulas

$$a_j^{'} = \frac{1}{n}\sum_i p_{ij}\,,\ \boldsymbol{\mu}_j^{'} = \frac{\sum_i p_{ij}\mathbf{x}_i}{\sum_i p_{ij}}\,,\ \boldsymbol{\Sigma}_j^{'} = \frac{\sum_i p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}_j^{'}\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_j^{'}\right)^t}{\sum_i p_{ij}} \tag{2.7}$$

The above steps (i.e. E-step and M-step) are repeated until convergence is achieved.

The first equation of (2.7) which maximizes equation (2.4) is derived by using Lagrange multipliers (Spiegel, 1974) on the first expression of equation (2.4), i.e. $\max \sum_{i,j} p_{ij} \log\left(a_j^{'}\right)$ subject to $\sum_j a_j^{'} = 1$, followed by $\frac{\partial}{\partial a_j^{'}}\left(\sum_{i,j^*} p_{ij^*} \log\left(a_{j^*}^{'}\right) + \lambda\left(\sum_{j^*} a_{j^*}^{'} - 1\right)\right) = 0$. The results from the derivative are as follows: $\sum_{i=1}^{n}\left(\frac{p_{ij}}{a_j^{'}}\right) + \lambda = 0$ and $\lambda = -n$. Performing a simple mathematical procedure on the derivative's results, we get the first equation of (2.7), i.e. $a_j^{'} = \frac{1}{n}\sum_i p_{ij}$.

We apply the derivative to the extended version of equation (2.4)'s second expression and limited to $\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i - \boldsymbol{\mu}_{j^*}^{'}\right)^t \boldsymbol{\Sigma}_{j^*}^{'\,-1}\left(\mathbf{x}_i - \boldsymbol{\mu}_{j^*}^{'}\right)$. We equate it to zero, i.e.

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^{'}}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i - \boldsymbol{\mu}_{j^*}^{'}\right)^t \boldsymbol{\Sigma}_{j^*}^{'\,-1}\left(\mathbf{x}_i - \boldsymbol{\mu}_{j^*}^{'}\right)\right) = 0 \text{ or}$$

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^{'}}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i^{\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\mathbf{x}_i - \mathbf{x}_i^{\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\boldsymbol{\mu}_{j^*}^{'} - \boldsymbol{\mu}_{j^*}^{'\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\mathbf{x}_i + \boldsymbol{\mu}_{j^*}^{'\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\boldsymbol{\mu}_{j^*}^{'}\right)\right) = 0 \qquad (2.8).$$

The results for second, third and fourth expressions of equation (2.8) are as follows where we use $\frac{\partial \mathbf{x}^t \mathbf{A}\mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}\mathbf{y}$ and $\frac{\partial \mathbf{a}^t \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$ (Mardia et al., 1979):

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^{'}}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i^{\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\boldsymbol{\mu}_{j^*}^{'}\right)\right) = \frac{1}{2}\sum_i p_{ij}\left(\mathbf{x}_i^{\,t}\boldsymbol{\Sigma}_j^{'\,-1}\right),$$

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^{'}}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\boldsymbol{\mu}_{j^*}^{'\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\mathbf{x}_i\right)\right) = \frac{1}{2}\sum_i p_{ij}\left(\boldsymbol{\Sigma}_j^{'\,-1}\mathbf{x}_i\right) \text{ and}$$

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^{'}}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\boldsymbol{\mu}_{j^*}^{'\,t}\boldsymbol{\Sigma}_{j^*}^{'\,-1}\boldsymbol{\mu}_{j^*}^{'}\right)\right) = \frac{1}{2}\sum_i p_{ij}\left(2\boldsymbol{\Sigma}_j^{'\,-1}\boldsymbol{\mu}_j^{'}\right)$$

Applying the results to equation (2.8), we get the second equation of (2.7), i.e. $\boldsymbol{\mu}'_j = \frac{\sum_i p_{ij}\mathbf{x}_i}{\sum_i p_{ij}}$.

We repeat the above steps but this time on the extended version of equation (2.4)'s second expression, i.e. $\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i - \boldsymbol{\mu}'_{j^*}\right)^t \boldsymbol{\Sigma}'^{-1}_{j^*}\left(\mathbf{x}_i - \boldsymbol{\mu}'_{j^*}\right) + \frac{1}{2}\sum_{i,j^*} p_{ij^*}\log\left((2\pi)^d\left|\boldsymbol{\Sigma}'_{j^*}\right|\right)$. We apply the derivative and equate it to zero, i.e.

$$\frac{\partial}{\partial \boldsymbol{\Sigma}'^{-1}_j}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i - \boldsymbol{\mu}'_{j^*}\right)^t \boldsymbol{\Sigma}'^{-1}_{j^*}\left(\mathbf{x}_i - \boldsymbol{\mu}'_{j^*}\right) + \frac{1}{2}\sum_{i,j^*} p_{ij^*}\log\left((2\pi)^d\left|\boldsymbol{\Sigma}'_{j^*}\right|\right)\right) = 0 \qquad (2.9).$$

The results for first and second expressions of equation (2.9) are as follows where we use $\frac{\partial tr(\mathbf{xy})}{\partial \mathbf{x}} = \mathbf{y} + \mathbf{y}^t - Diag(\mathbf{y})$ and $\sum \mathbf{x}_i^t \mathbf{A}\mathbf{x}_i = tr(\mathbf{A}\sum \mathbf{x}_i\mathbf{x}_i^t)$ (Mardia et al., 1979):

$$\frac{\partial}{\partial \boldsymbol{\Sigma}'^{-1}_j}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\left(\mathbf{x}_i - \boldsymbol{\mu}'_{j^*}\right)^t \boldsymbol{\Sigma}'^{-1}_{j^*}\left(\mathbf{x}_i - \boldsymbol{\mu}'_{j^*}\right)\right) = \frac{1}{2}\sum_i 2\,p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)^t - \frac{1}{2}diag\left(\sum_i p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)^t\right)$$

and

$$\frac{\partial}{\partial \boldsymbol{\Sigma}'^{-1}_j}\left(\frac{1}{2}\sum_{i,j^*} p_{ij^*}\log\left((2\pi)^d\left|\boldsymbol{\Sigma}'_{j^*}\right|\right)\right) = \frac{1}{2}\sum_i p_{ij}\left(2\boldsymbol{\Sigma}'_j - diag\left(\boldsymbol{\Sigma}'_j\right)\right)$$

Applying the results to equation (2.9), we get

$$\left(\sum_i p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)^t - \sum_i p_{ij}\boldsymbol{\Sigma}'_j\right) - \frac{1}{2}diag\left(\sum_i p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)^t - \sum_i p_{ij}\boldsymbol{\Sigma}'_j\right) = 0 \qquad (2.10).$$

The condition as displayed by equation (2.10) is satisfied when

$$\sum_i p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)^t - \sum_i p_{ij}\boldsymbol{\Sigma}'_j = 0\,. \qquad (2.11)$$

Performing a simple mathematical procedure on equation (2.11), we get the third equation of (2.7), i.e. $\boldsymbol{\Sigma}'_j = \dfrac{\sum_i p_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}'_j\right)^t}{\sum_i p_{ij}}$.

2.3 Summary

A brief introduction to pattern recognition method namely Gaussian Mixed Model (GMM) is given. The said method is commonly used in voice recognition technique and used EM algorithm to solve the problem related to maximum likelihood estimation. The introduction also listed all of the equations where derivation and weaknesses are provided for some of them.

In the coming chapters, we will focus on the strength and weaknesses of GMM to detect fraud activities in telecommunication industry by using real data different from those that are normally used in voice recognition technique and propose a solution to solve them.

CHAPTER 3

IMPROVED EXPECTATION MAXIMIZATION ALGORITHM FOR GAUSSIAN MIXED MODEL USING THE KERNEL METHOD

3.1 Introduction

A number of authors highlighted the importance of identifying the right number, say *k*, of components in a GMM and subsequently choosing good initial values for the model parameters μ_i and σ_i^2, $i = 1,2,\ldots k$, in the EM algorithm. Schlattmann (2003) noted the difficulty of using log-likelihood ratio statistics to test the number of components and subsequently suggested using a non-parametric bootstrapping approach. Similarly, Wang et al. (2004) pointed out the same concerns and introduced an algorithm called the stepwise-split-and-merge EM algorithm to solve the said problem. In addition, Miloslavsky and Van Der Laan (2003) investigated the possibility of using the minimization of the Kullback-Leiber distance between fitted mixture models and the true density as a method for estimating *k* where the said distance was estimated using cross validation. Zhuang et al. (1996) viewed the mixture distribution as a contaminated Gaussian density and proposed a recursive algorithm called the Gaussian mixture density decomposition Algorithm for identifying each Gaussian component in the mixture. Other works on this topic can also be found, for example, in Lee et al. (2006) and Celeux and Soromenho (1996).

This chapter attempts to use Gaussian mixed model which is a probabilistic model normally used in speech recognition to identify fraud calls in the

telecommunication industry. We look at several issues encountered when calculating the maximum likelihood estimates of the Gaussian mixed model using an expectation maximization algorithm. Firstly, we look at a mechanism for the determination of the initial number of Gaussian components and the choice of the initial values of the algorithm using the kernel method (Section 3.2). We show via simulation that the technique improves the performance of the algorithm (Sections 3.3 and 3.4). Secondly, we develop a procedure for determining the order of the Gaussian mixed model using the log-likelihood function and the Akaike information criteria (Section 3.5). Finally, for illustration, we apply the improved algorithm to real telecommunication data (Section 3.6). The modified method will pave the way to introducing a comprehensive method for detecting fraud calls in future work.

3.2 The Kernel Method

The kernel method can be used to find the probability density estimate for univariate data, see for example Silverman (1986). Let $\alpha < \min(x_i) - 3h$, $\beta > \max(x_i) + 3h$, $M = 2^r$ (for some integer r), h be the bandwidth, $\delta = \frac{\beta - \alpha}{M}$ and $t_k = \alpha + k\delta$ be the k^{th} grid point where $k = 0,1,\ldots,M-1$. The density estimate at grid point t_k is represented by the following equation:

$$\hat{f}(t_k) \approx \sum_{l=-\frac{M}{2}}^{\frac{M}{2}} \left(\frac{1}{M} \sum_{k=0}^{M-1} \xi_k \exp\left(\frac{2\pi kl}{M} i \right) \right) \exp\left(\left(-\frac{2\pi kl}{M} i \right) - \frac{1}{2} h^2 \left(\frac{2\pi l}{\beta - \alpha} \right)^2 \right) \tag{3.1}$$

where $i^2 = -1$.

For $x \in [t_k, t_{k+1}]$, the density estimate $\hat{f}(x)$ is defined by $\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$ where n equals to total number of observations and $K(t) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}t^2\right)$. To compute $\hat{f}(x)$ at a grid of points, a method which makes use of the Fourier transform is employed. Let $\tilde{f}(s)$ be the Fourier transform of the kernel density estimate $\hat{f}(x)$. It can be shown that $\tilde{f}(s) = (2\pi)^{\frac{1}{2}}\tilde{K}(hs)u(s) = \exp\left(-\frac{1}{2}h^2 s^2\right)u(s)$ where $\tilde{K}(s)$ is the Fourier transform of the Gaussian kernel and $u(s) = (2\pi)^{-\frac{1}{2}} n^{-1}\sum_{j=1}^{n}\exp(isx_j)$ is the Fourier transform of the data. Thus, $\hat{f}(x) = (2\pi)^{-\frac{1}{2}}\int e^{-isx}(2\pi)^{\frac{1}{2}}\tilde{K}(hs)u(s)ds$ is the convolution of the data with the kernel.

We will use the following algorithm by Silverman (1986) to discretize the data to very fine grids, and to find $\hat{f}(x)$ by convolving the data with the kernel.

Step A: Discretize the data to find the weight sequence $\{\xi_k\}$ with $M = 2^8$. If $x \in [t_k, t_{k+1}]$, it is split into a weight $\frac{1}{n\delta^2}(t_{k+1} - x)$ at t_k and a weight $\frac{1}{n\delta^2}(x - t_k)$ at t_{k+1}; these weights are accumulated over all the data points x_i to give a sequence of (ξ_k) weights summing up to $\frac{1}{\delta}$.

Step B: Find the sequence $\{Y_l\}$ defined by $Y_l = M^{-1}\sum_{k=0}^{M-1}\xi_k \exp\left(\frac{2\pi kl}{M}i\right)$ where $-\frac{M}{2} \le l \le \frac{M}{2}$. It can be shown that when $\alpha = 0$, $Y_l \approx (2\pi)^{\frac{1}{2}}(\beta-\alpha)^{-1}u(s_l)$ where $s_l = \frac{2\pi l}{\beta-\alpha}$.

Step C: Find the sequence $\{\zeta_l^*\}$ where $\zeta_l^* = \exp\left(-\frac{1}{2}h^2 s_l^2\right)Y_l$. Here, $h = 0.9An^{-\frac{1}{5}}$ where $A = \min\left(sd, \frac{IQR}{1.34}\right)$, *sd* is the standard deviation, and *IQR* is the inter-quartile range. The *IQR* is chosen here by Silverman (1986), who claimed that the bandwidth is useful for a wide range of densities.

Step D: Let ζ_k be the inverse discrete Fourier transform of ζ_l^* i.e. $\zeta_k = \sum_{l=-\frac{M}{2}}^{\frac{M}{2}} \zeta_l^* \exp\left(-\frac{2\pi kl}{M}i\right)$.

It can be shown that when $\alpha = 0$, $\hat{f}(t_k) \approx \zeta_k$. We then identify x_i where its density estimate, denoted by $\hat{f}(x_i)$, is greater than those of its nearest neighbors x_{i-1} and x_{i+1}. In other words, $\hat{f}(x_i) > \hat{f}(x_{i-1})$ and $\hat{f}(x_i) > \hat{f}(x_{i+1})$, refer to Figure 3.1, where the vertical line that touches t_k and $\hat{f}(t_k)$ shows the location of the peak.

Note that we may obtain more than one maximum point which means that the data may consist of more than one Gaussian distribution. These results form a very important component of the improved EM algorithm for GMM to be described next.

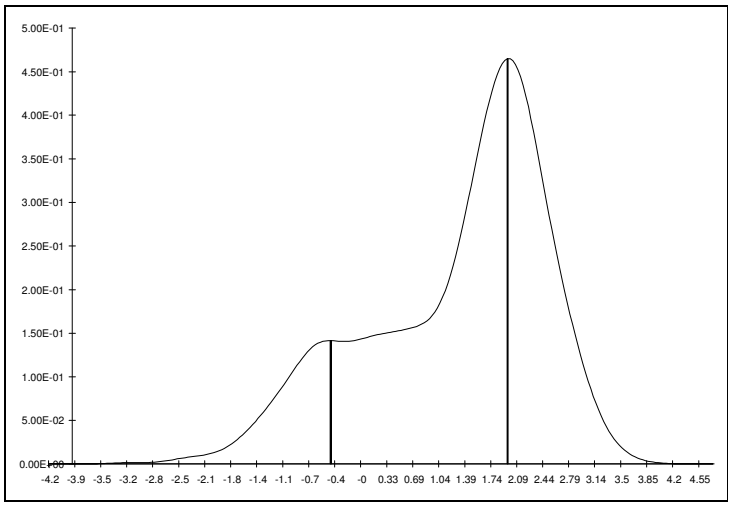


Figure 3.1 Plot of $\hat{f}(t_k)$ against t_k

3.3 Improved EM Algorithm for GMM

In this section, we propose an improved EM algorithm for GMM which can perform both tasks: identifying the initial number of components and providing automatic initial values for the EM algorithm. The full improved EM algorithm for GMM is now presented:

Step 1: The kernel method as described in Section 3.2 is used to determine the number K_0 of components and also the corresponding means μ_i of each component, $i = 1,2,\ldots,K_0$. The initial estimates of the standard deviations σ_{ii} are set to unity while the prior weights a_i are set to be $\frac{1}{K_0}$;

Step 2: The EM algorithm for a GMM as described in Section 2.2 is executed to give the final estimates of parameters μ_i, σ_{ii} and a_i, $i = 1,2,\ldots,K_0$. The log-likelihood function and Akaike information criteria (AIC) are calculated using the said parameters;

Step 3: Step 2 is repeated for other possible number K of components with μ_i=0, $\sigma_{ii} = 1$ for the other $K - K_0$ components and $a_i = \frac{1}{K}$.

Step 4: The log-likelihood function and AIC values for $K = 1,2,\ldots,10$ are plotted. The final number of components K_f is chosen when adding extra components in the model does not significantly increase or decrease the values of the log-likelihood function and the AIC respectively.

3.4 Simulation

We use simulation to investigate the performance of the proposed improved EM algorithm.

3.4.1. Simulation Scheme

Simulation data were generated using the Box and Muller Transformation (1958) as defined by equations (3.2-1) and (3.2-2) below:

$$z_j = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \cos 2\pi u_{j+1} \qquad (3.2\text{-}1)$$

$$z_{j+1} = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \sin 2\pi u_{j+1} \qquad (3.2\text{-}2)$$

where $u_j, u_{j+1} \sim U(0,1)$. For the case of two components, we start by generating a random number $u_1 \sim U(0,1)$. If $0 < u_1 < a_1$, we generate two random numbers

$u_2 \sim U(0,1)$ and $u_3 \sim U(0,1)$ and calculate $z_2 + z_3$ using equations (3.2-1) and (3.2-2) with $\mu^* = \frac{\mu_1}{2}$, and $\sigma^* = \frac{\sigma_1}{\sqrt{2}}$. Otherwise, we use $\mu^* = \frac{\mu_2}{2}$, and $\sigma^* = \frac{\sigma_2}{\sqrt{2}}$. The process continues until the required sample size is obtained. The scheme is easily extended to any number of components. For further details, refer to Fishman (2001).

Table 3.1 List of true values of *a's,* μ*'s*, σ*'s*

Sample name and size (in bracket)	**Prior probability**	**Mean**	**Variance**
Sample 1 Two components	a_1=0.4	μ_1=0.0	σ_1^2=1.0
	a_2=0.6	μ_2=2.0	σ_2^2=0.25
Sample 2 Two components	a_1=0.85	μ_1=0.0	σ_1^2=1.0
	a_2=0.15	μ_2=2.0	σ_2^2=0.25
Sample 3 Three components	a_1=0.33	μ_1=0.0	σ_1^2=1.0
	a_2=0.33, a_3=0.34	μ_2=-1.0, μ_3=4.0	σ_2^2=0.25, σ_3^2=4.0

3.4.2 Study of performance based on log-likelihood function

We first look at the performance of the standard method, called Method 1, followed by that of the improved method, called Method 2. For Method 1, in place of Step 1 of the improved method, we assign values zero and unity respectively to the means and variances of all components. We compare the performances by looking at the log-likelihood function via simulation study.

Following Everitt and Hand (1981), we consider two cases with two components and one case with three components with the true values of the parameters given in Table 3.1. For each case, we generate 100 samples of size 1000 where the chosen sample size reflects the large size of data sets found in the telecommunication industry, the focus of our interest. Figure 3.2 shows histograms for all cases, each with a sample of size 1000, where (a) two peaks are observed representing two components,

(b) two components are observed where the second component is partially hidden and (c) three components but only two are observed where the third component is totally hidden. This scenario is best described by the percentage of overlapping, which will be discussed in the later section.

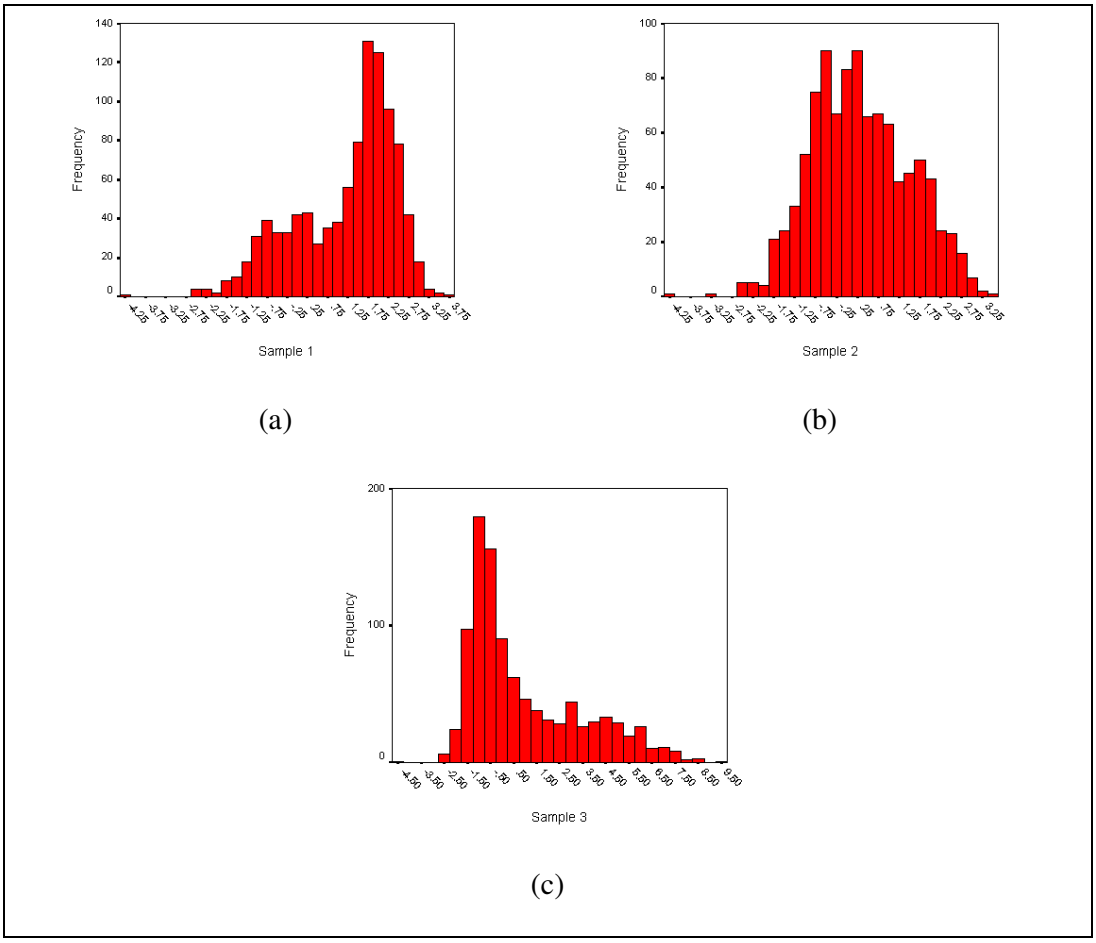
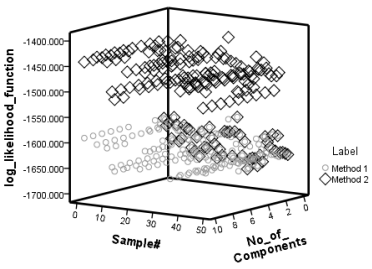


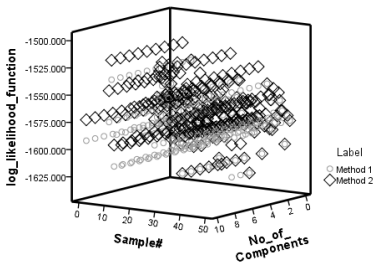
Figure 3.2 The histograms of (a) Sample 1 (with overall mean and standard deviation equal to 1.28 and 1.19, respectively), (b) Sample 2 (with overall mean and standard deviation equal to 0.34 and 1.19, respectively) and (c) Sample 3 (with overall mean and standard deviation equal to 1.18 and 2.62, respectively).

We then apply Method 1 and Method 2 on the simulated data. For each case and better quality viewing, we plot only 50 values selected randomly of the log-likelihood function for both methods on the same plot, as given in Figure 3.3. Figures 3.4, 3.5 and

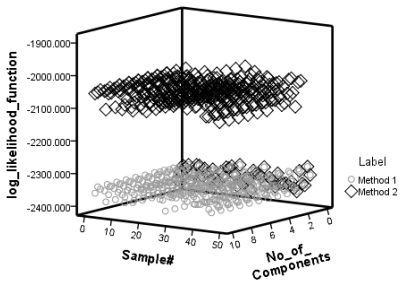
3.6 give the plots of log-likelihood function against number of components for the three samples considered. It can be seen that, for Sample 1 and Sample 3, the proposed Method 2 clearly outperforms the standard Method 1 with the values of the log-likelihood function corresponding to Method 2 always larger than those of Method 1. However, we see that some values overlap for Sample 2, though the proposed Method 2 still generally performs better. In this case, the prior probabilities a_i are distinctly different from the chosen values of a_i in Sample 1 while other true values remain the same which leads to different percentages of overlapping of the Gaussian components in the GMM.



(a) Sample 1

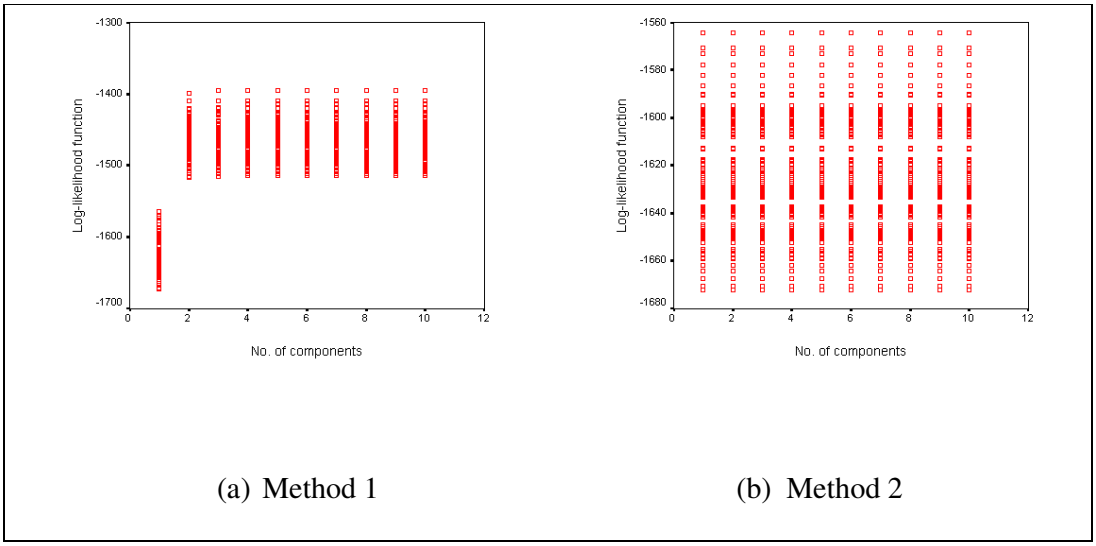


(b) Sample 2



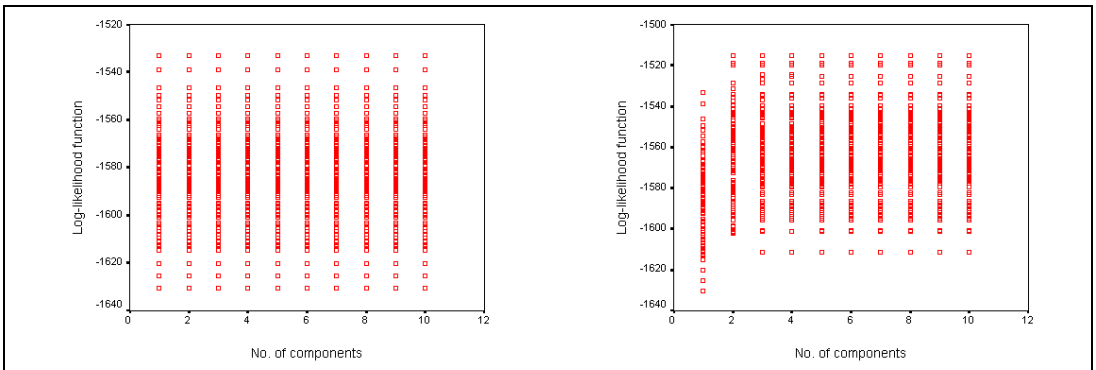
(c) Sample 3

Figure 3.3 Plots of values of log-likelihood function.



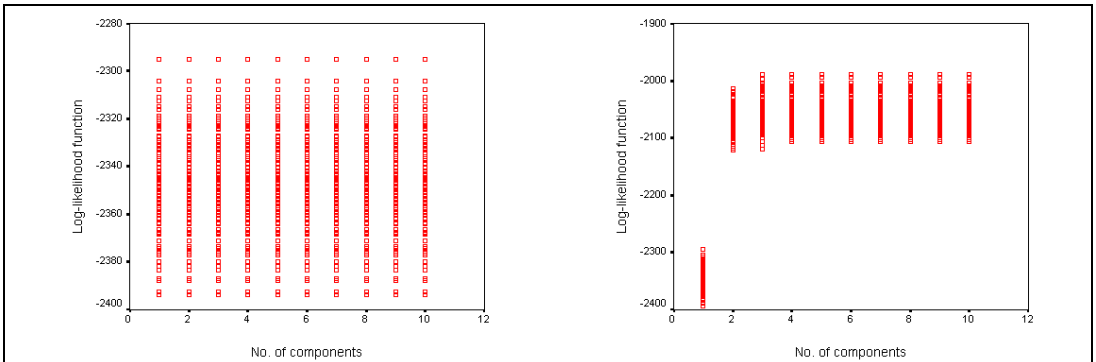
(a) Method 1 (b) Method 2

Figure 3.4 Plot of log-likelihood function against number of components for Sample 1.



(a) Method 1 (b) Method 2

Figure 3.5 Plot of log-likelihood function against number of components for Sample 2.



(a) Method 1 (b) Method 2

Figure 3.6 Plot of log-likelihood function against number of components for Sample 3.

Figure 3.7 gives the plots of log-likelihood function against number of components for simulated data sets number 10, 25, 75 and 90 of Sample 1. It can be seen that the log-likelihood function value improves from 1 till 2 components and becomes constant from 2 components onwards. Figure 3.8 shows similar results for the case simulated data sets number 10, 25, 75 and 90 of Sample 2. Figure 3.9 gives the plots of log-likelihood function against number of components for simulated data sets number 25, 40, 75 and 90 of Sample 3 and they show log-likelihood function value improves from 1 till 3 components and becomes constant from 3 components onwards thus revealing (or exposing) the so-called "hidden component".

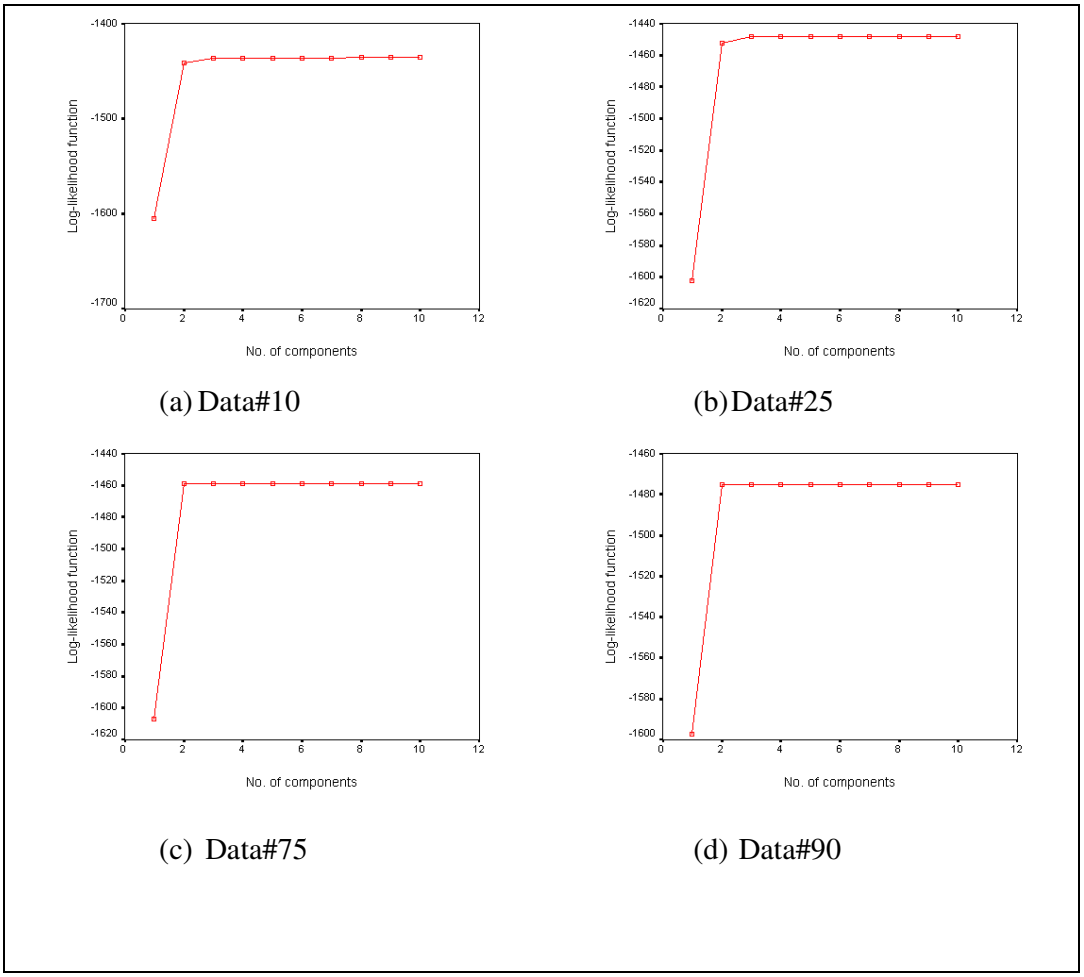


Figure 3.7 Plot of log-likelihood function for selected simulated data set from Sample 1

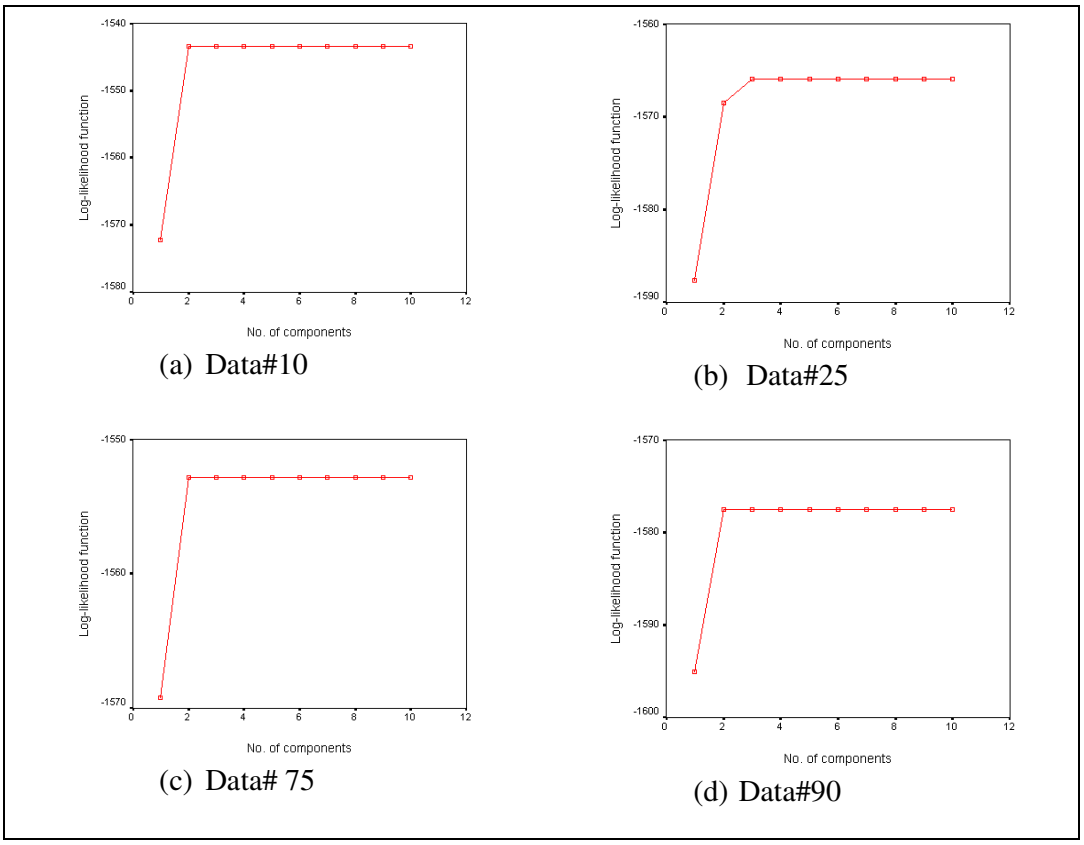


Figure 3.8 Plot of log-likelihood function for selected simulated data set from Sample 2.

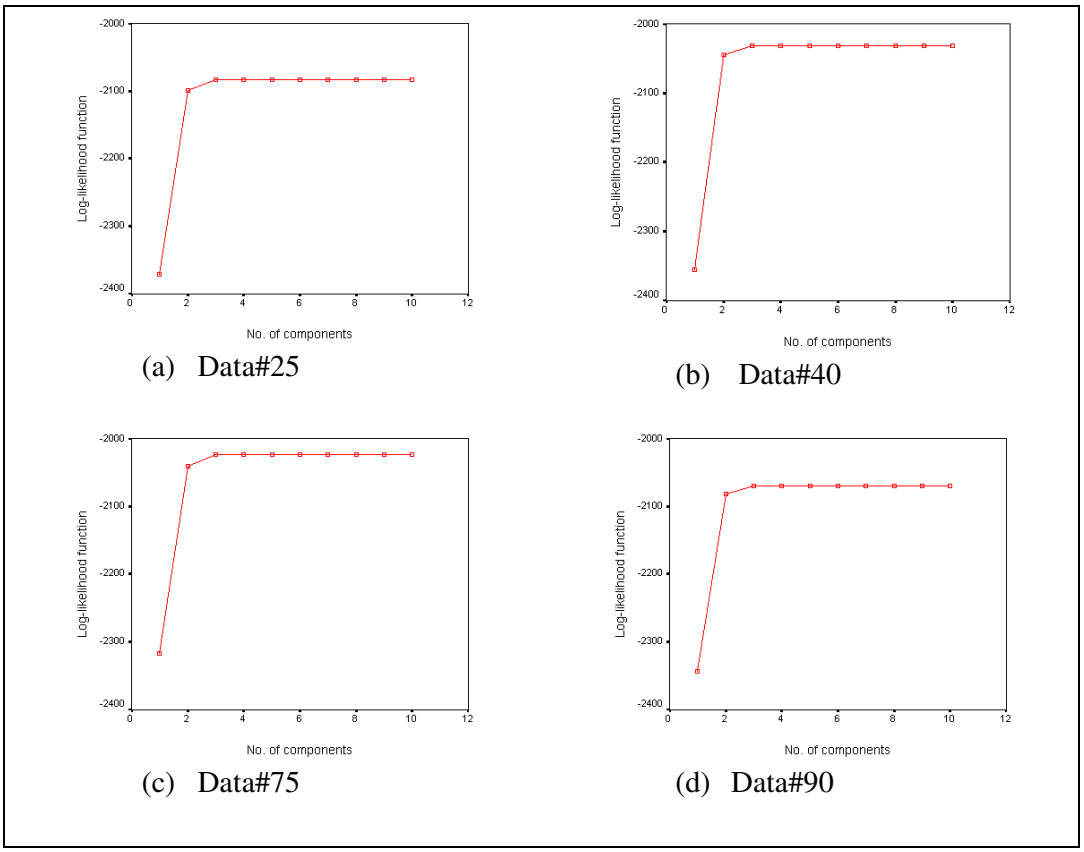


Figure 3.9 Plot of log-likelihood function for selected simulated data set from Sample 3.

We will investigate the performance of the improved EM algorithm in estimating the parameters of the GMM by taking into account the effect of different percentages of overlapping between the components observed in the data.

3.4.3 The value of intersections

The value of intersections (as shown in Figure 3.10(a)) for the case when $\mu_1 \neq \mu_2$, $\sigma_1 \neq \sigma_2$, $f_1(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2}$ and $f_2(y) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu_2}{\sigma_2}\right)^2}$ are obtained from $x_{11} = \frac{-b+\sqrt{b^2-4ac}}{2a}$ and $x_{12} = \frac{-b-\sqrt{b^2-4ac}}{2a}$ where $a = \left(\sigma_2^2 - \sigma_1^2\right)$, $b = 2\left(\sigma_1^2\mu_2 - \sigma_2^2\mu_1\right)$, and $c = \left(\sigma_2^2\mu_1^2 - \sigma_1^2\mu_2^2\right) - 2\sigma_1^2\sigma_2^2 \log\left(\frac{\sigma_2}{\sigma_1}\right)$. Firstly, using the above formula as well as $P(\frac{x-\mu}{\sigma}) = \int_{-\infty}^{\frac{x-\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$, we find the area between x_{11} and x_{12} (and convert it into percentage) for each component; refer to Figure 3.10(a). Secondly, we find the minimum between the areas of the two components. This value represents the percentage of overlapping between two components (which is an approximation). For the case when $\mu_1 \neq \mu_2$, $\sigma_1 = \sigma_2$, $f_1(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2}$ and $f_2(y) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu_2}{\sigma_2}\right)^2}$, let $d = \left(\mu_1 + 2\sigma_1\right) - \left(\mu_2 - 2\sigma_2\right)$. The value of the intersection, say x_1, is approximated from the following formula:

$$x_1 = \begin{cases} 0 & d < 0 \\ \left(\mu_1 + 2\sigma_1\right) & d = 0 \\ \left(\mu_1 + 2\sigma_1\right) - \frac{d}{2} & d > 0 \end{cases}.$$

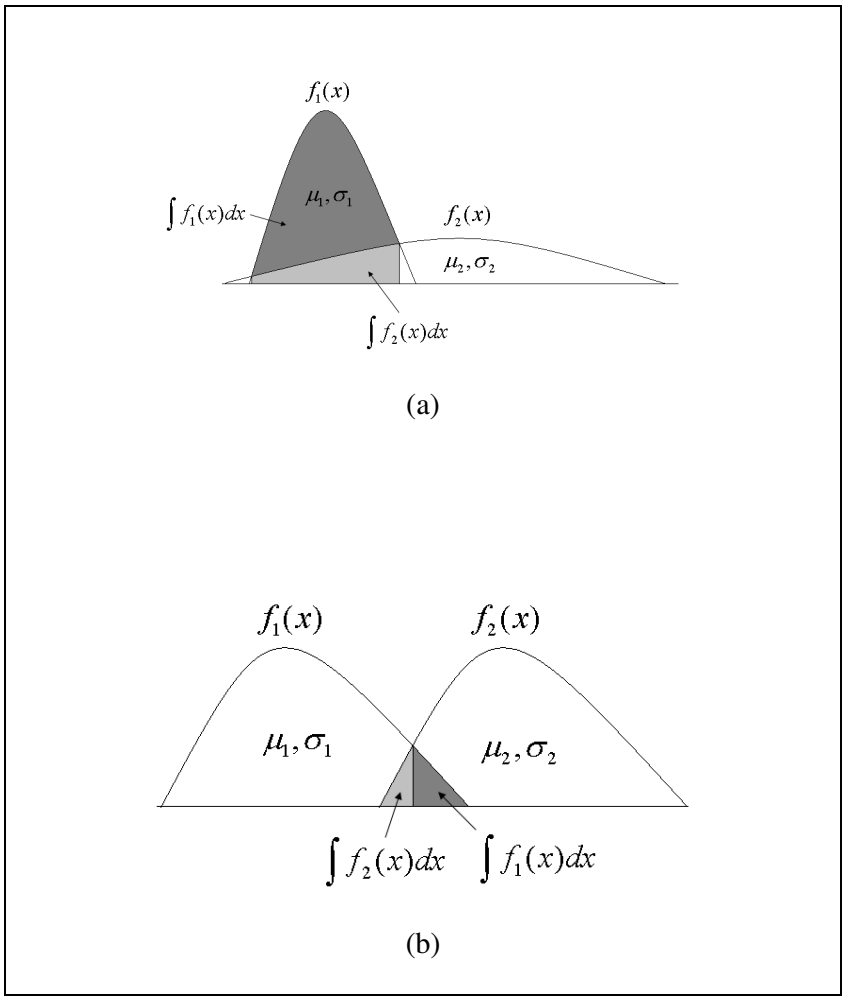


Figure 3.10 $\int f_1(x)dx$ and $\int f_2(x)dx$ are used to find the shaded areas as shown in (a) and (b).

Taking similar steps, the area for the component on the left hand side of Figure 3.10(b) is obtained from $1 - P(\frac{x_1 - \mu_1}{\sigma_1}) = 1 - \int_{-\infty}^{\frac{x_1-\mu_1}{\sigma_1}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$ and that of the component on the right hand side of Figure 3.10(b) from $P(\frac{x_1 - \mu_2}{\sigma_2}) = \int_{-\infty}^{\frac{x_1-\mu_2}{\sigma_2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$. We convert them into percentages before adding them up to represent the percentage of overlapping between two components (which is an approximation).

3.4.4 The effects of different overlapping percentages on performance

The main objective here is to investigate the performance of the improved EM algorithm for different overlapping percentages of the components in the GMM. For simplicity, we restrict our attention to two components so that $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_6) = (a_1, a_2, \mu_1, \mu_2, \sigma_{11}, \sigma_{22})$ are to be estimated. Data is simulated using the simulation scheme described in Section 3.4.1.

After performing Steps 1 and 2, we find $D_i = \theta_i - \hat{\theta}_i$ where θ_i is the true value of the ith parameter and $\hat{\theta}_i$ is the EM estimate of the parameter, *i=1,2,…,n*. The sample mean and standard deviation of D_i are computed using formulas $\overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i$ and $s_D = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2}$. The estimates are considered good if $\overline{D}$ is close to zero, indicating small biases observed in the simulation results, and s_D is also close to zero, indicating that the parameter estimates are concentrated around their respective true values.

We determine the area of overlapping between the two components for each model by using the misclassification concept given in Johnson and Wichern (1998), the details of which are provided in Section 3.4.3. The formula to estimate the overlapping areas depends on the mean and standard deviation of the components. The choices of prior probabilities should not affect the estimates greatly as their sum equals unity.

We consider three cases for different combinations of parameter $\boldsymbol{\theta}$ which give different percentages of overlapping of the GMM components. The results are

tabulated in Tables 3.2-3.4. Table 3.2 deals with case 1, where the true values of μ_1=0, μ_2=3.0 and $\sqrt{\sigma_{11}} = \sqrt{\sigma_{22}} = 0.316$ are fixed but the true values of a_1, a_2 are varied. In all cases, the percentage of overlapping is 0% as the separation of the means is rather large with small values of dispersion. We can see that the values of the mean are close to zero with the small standard errors less than unity for all parameters considered. On the other hand, Table 3.3 gives the results for case 2 where μ_1=0, μ_2=1.0, $\sqrt{\sigma_{22}} = 0.707$ and $\sqrt{\sigma_{11}} = 0.447$ are fixed but a_1, a_2 are varied to give 25% of overlapping. The bias is still considered small but generally larger than that for case 1. In addition, the values are more dispersed here. Finally, Table 3.4 shows the results of case 3 where μ_1=0, μ_2=0.25, $\sqrt{\sigma_{11}} = 0.577$ and $\sqrt{\sigma_{22}} = 1.414$ are fixed with 45% of overlapping. As expected, the results deteriorate when the percentage of overlapping increases.

For each model and final estimates of parameters, we check whether $\left|\overline{D}\right| < 1$ and $S_D^2 < 1$. Out of six (6), we count the number of $\left|\overline{D}\right| < 1$ and find its percentage, which is denoted by A. We repeat the same process for S_D, that is $S_D^2 < 1$, where its percentage is denoted by B. We find the smallest percentage between A and B, which is denoted by $Min\%$. We then plot $Min\%$ against the range for percentage of overlapping (or Range). Range equals to 1 represents percentage of overlapping between 0% and 25%, 2 between 25% and 50%, 3 between 50% and 75% and 4 between 75% and 100%. Note that second component will "hide" behind the first component as Range increases.

Figure 3.11 shows median for Range equals to 1 is located at 100, 54.9% of $Min\%$ equals to 100, and 65.2% of $Min\%$ is greater than (or equal to) 83. Range equals to 2, its median is located at 66.67% where 35.4% of $Min\%$ is greater than (or

equal to) 83. Range equals to 3, its median is similar to the above (that is 66.67%) where 17.4% of *Min%* is greater than (or equal to) 83. Range equals to 4, its median is similar to the above (that is 66.67%) where 24.2% of *Min%* is greater than (or equal to) 83.

Table 3.2 Simulation results for the case $\mu_1=0,\ \mu_2=3.0$ and $\sqrt{\sigma_1^2}=\sqrt{\sigma_2^2}=0.316$

Prior prob.		Bias, D_i											
		a_1		a_2		μ_1		μ_2		σ_1^2		σ_2^2	
a_1	a_2	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D
0.1	0.9	-0.001	0.010	0.001	0.010	0.007	0.033	-0.003	0.013	0.000	0.014	0.001	0.004
0.2	0.8	0.002	0.013	-0.002	0.013	0.002	0.022	-0.003	0.011	0.000	0.011	0.002	0.007
0.3	0.7	-0.002	0.014	0.002	0.014	0.004	0.017	-0.002	0.010	0.002	0.009	0.001	0.005
0.4	0.6	0.003	0.020	-0.003	0.020	-0.004	0.019	-0.006	0.012	0.000	0.009	0.001	0.005

Table 3.3 Simulation results for the case $\mu_1=0,\ \mu_2=1.0,\ \sqrt{\sigma_2^2}=0.707$ and $\sqrt{\sigma_1^2}=0.447$

Prior prob.		Bias, D_i											
		a_1		a_2		μ_1		μ_2		σ_1^2		σ_2^2	
a_1	a_2	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D
0.1	0.9	-0.112	0.148	0.112	0.148	-0.130	0.203	-0.067	0.139	-0.030	0.103	0.034	0.068
0.2	0.8	-0.014	0.073	0.014	0.073	-0.006	0.087	0.006	0.091	0.022	0.056	0.005	0.057
0.3	0.7	0.020	0.087	-0.020	0.087	0.031	0.069	0.023	0.104	0.044	0.048	-0.006	0.075
0.4	0.6	0.067	0.075	-0.067	0.075	-0.112	0.444	0.145	0.232	-0.002	0.085	-0.021	0.099

Table 3.4 Simulation results for the case $\mu_1=0,\ \mu_2=0.25,\ \sqrt{\sigma_1^2}=0.577$ and $\sqrt{\sigma_2^2}=1.414$

Prior prob.		Bias, D_i											
		a_1		a_2		μ_1		μ_2		σ_1^2		σ_2^2	
a_1	a_2	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D	$\overline{D}$	S_D
0.1	0.9	0.089	0.007	-0.089	0.007	-0.817	3.770	0.045	0.048	-0.341	0.823	0.305	0.095
0.2	0.8	0.157	0.108	-0.157	0.108	0.291	3.521	0.075	0.073	-0.169	0.374	0.413	0.256
0.3	0.7	0.237	0.100	-0.237	0.100	-0.205	2.924	0.105	0.121	-0.278	0.389	0.578	0.278
0.4	0.6	0.245	0.187	-0.245	0.187	0.602	2.520	0.092	0.108	-0.008	0.258	0.508	0.435

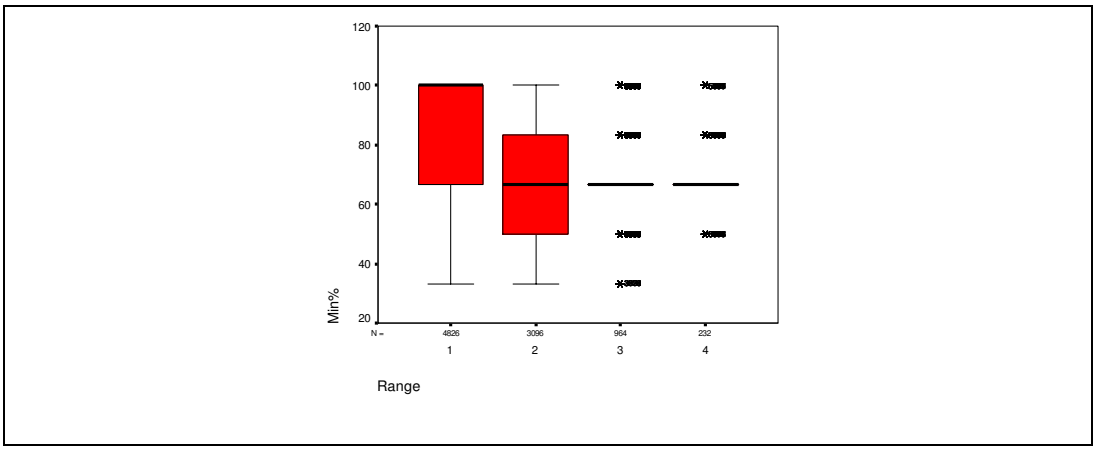


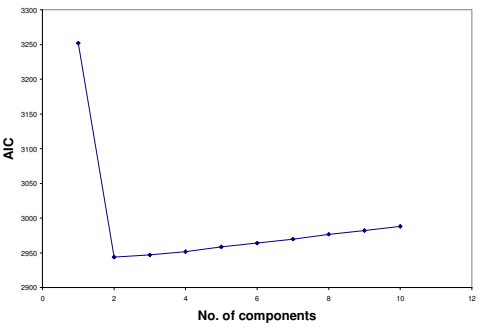
Figure 3.11 *Min%* is plotted against Range for Box and Whisker plot.

We conclude that the improved EM algorithm for GMM performs well when the percentages of overlapping are small, but its performance is affected when the percentages increase.

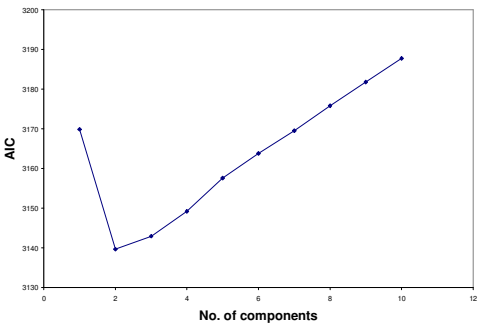
3.5. Determination of the final number of components in the GMM using AIC

In the last two steps of the improved algorithm, we intend to confirm that the choice of the initial number K_0 of components in the GMM using the kernel method is final. This can be done by considering extra components in the model. For that, as stated in Section 3.3.3, we repeat Step 2 for other possible numbers K of components, by setting μ_i=0, σ_{ii}=1 for the other $K - K_0$ components and $a_i = \frac{1}{K}$. The final number of components K_f is chosen when adding extra components neither increases the log-likelihood nor decreases the AIC values significantly. The changes can easily be seen on a line plot of the values. Figure 3.12 shows the plots of AIC against number of components for data set, each from (a) Sample 1, (b) Sample 2 and (c) Sample 3 of Section 3.4.2. All of them show concave like shape where AIC decreases to a minimum value and then increases as the number of components increases. The

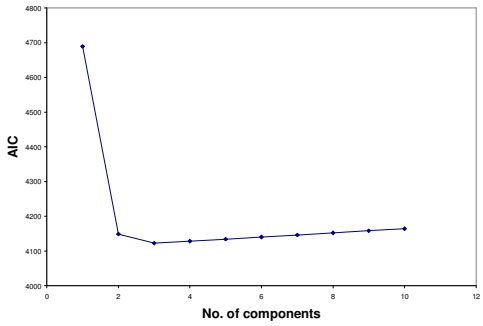
minimum value gives the exact number of components for the plots of AIC against number of components. In the case of Figure 3.12, plots (a) and (b) give 2 components and plot (c) gives 3 components.



(a) Sample 1



(c) Sample 2



(c) Sample 3

Figure 3.12 Plot of AIC of three data sets generated from samples defined in Table 3.1.

3.6 Real example – Phone call data

The call detail record, which was supplied by Telekom Malaysia Berhad (henceforth, TM), consists of calls made by customers that fell victim to fraud activities. Table 3.5 shows the format of the call detail record for each TM customer. We performed several steps on the original data in order to have the data in a desired format i.e. group the real data according to Service No, find the country that matches the Country Code and sort the real data according to Seize Time. The column entitled

"Seize time" gives the time when the call was made; the 4th and 5th columns detail the duration of the call in the following format: day (dd), hour (hh), minute (mm) and second (ss); and the 6th column is the result of converting the information in the 4th and 5th columns into day format.

We consider real data consisting of the duration of each call made by Customer A, whose identity is not revealed to ensure confidentiality, on 31st March 2011 as displayed in Figure 3.13**.** Step 1 of the improved EM algorithm for GMM identifies two initial components. The plot of the log-likelihood function and AIC in Figure 3.14(a) and (b) are the results from performing Steps 2, 3 and 4 of the improved EM algorithm for GMM, which reveal that the EM algorithm fails to achieve convergence when the number of components equals to five or above. It can also be seen that a GMM with 2 components is identified as the 'best' model, since the inclusion of more components not only fails to increase the value of the log-likelihood, but also fails to decrease the values of the AIC. The final EM estimates for the two-component GMM are $\hat{a}_1 = 0.64$, $\hat{a}_2 = 0.36$, $\hat{\mu}_1 = -0.66$, $\hat{\mu}_2 = 1.17$, $\hat{\sigma}_{11} = 0.07$ and $\hat{\sigma}_{22} = 0.35$, and they represent the behavior of calls made by Customer A on 31st March 2011. In the following chapters, we will show how the above information produced from the improved EM algorithm for GMM can be used in the process of detecting fraud activities in the telecommunication industry.

Table 3.5An extract from the TM's customer call detail record.

Service number	Dialed digits	Seize time	Duration (hhmmss)	Duration (dd)	Duration (Convert into day format)
Xxx	yyy	8:41:37	000339	00	0.002534722
Xxx	yyy	9:27:03	000035	00	4.05E-04
Xxx	yyy	9:43:46	000048	00	5.56E-04
Xxx	yyy	9:50:21	000031	00	3.59E-04
Xxx	yyy	10:54:30	000138	00	0.001134259

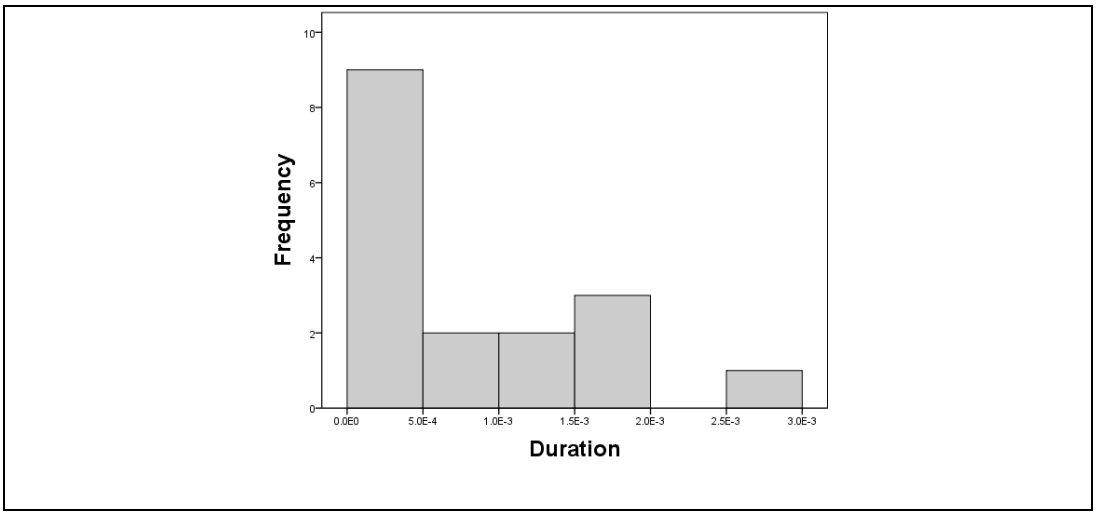


Figure 3.13 Duration (in day format) is displayed in the histogram.

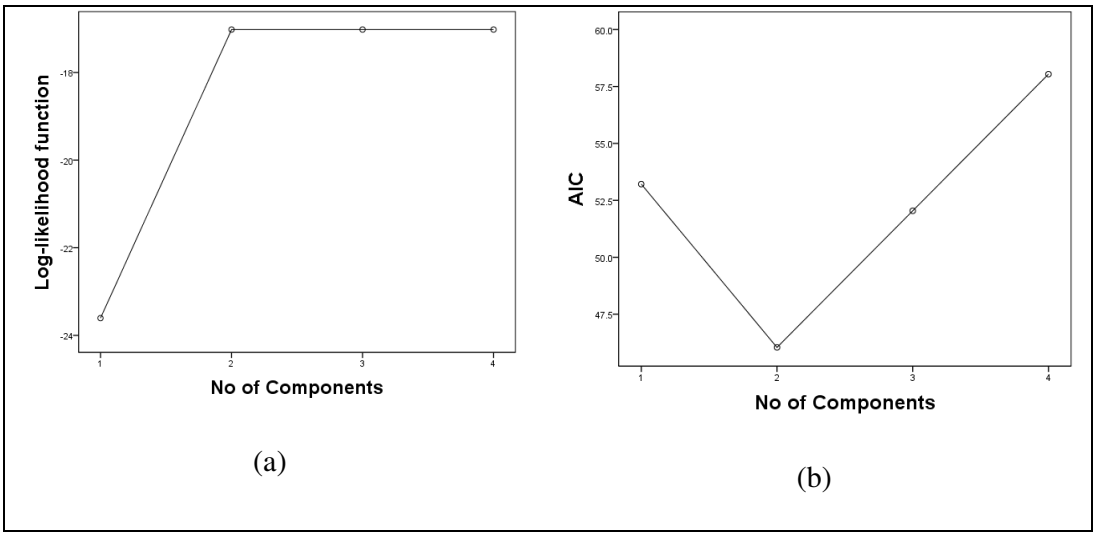


Figure 3.14 Plots of (a) log-likelihood values (b) AIC values.

3.7 Summary

In this chapter, we proposed a modified EM algorithm which can numerically identify the number of components of a GMM and estimate the parameters of the model using the kernel method. We showed via simulation that the performance of the algorithm is generally good but, as expected, is affected by increasing percentages of overlapping of the Gaussian components. We then used the line plots of the log-likelihood and AIC

values to identify the final number of GMM component. They could clearly be determined via the concave-like shape of the AIC plot, which indicates that the AIC decreases to a minimum value and then increases as the number of components increases. Finally, the modified EM algorithm for GMM was tested on real telecommunication data. The results serve as testimony to the effectiveness of the improved EM algorithm for GMM and should be useful when considering the problem of fraud calls faced by the telecommunication companies.

CHAPTER 4

FRAUD DETECTION IN TELECOMMUNICATION INDUSTRY USING GAUSSIAN MIXED MODEL

4.1 Introduction

Gaussian Mixed Model (GMM) has been widely used in voice recognition as exemplified next: suppose we have *Totspeak* speakers where each has *Totsamp* samples of recorded voices to be used as training data as shown in Figure 4.1 surrounded by the dotted line. Next, the GMM is fitted on each sample of recorded voice where it is in vector format after going through the coded process. The GMM parameters namely prior probability, mean and covariances are saved inside the database in training matrix format. They are given special designation as shown on the bottom right of Figure 4.1. For example, $a_{i,j,k}$ is the prior probability for *i*-th speaker with corresponding *j*-th sample and *k*-th component. $K(i, j)$ is the maximum number of components for the said speaker and sample. The identification of a new speaker is done as follows: the speaker's recorded voice is coded into vector $\boldsymbol{x}_{\ell}^{T}$, $\ell = 1, \ldots, n$, based on standard voice recognition criteria as shown on the top left of Figure 4.1. The vector would be known as data matrix from this point onwards. Next, the data matrix is used in $\sum_{L=1}^{n} \log\left(\sum_{k=1}^{K(i,j)} a_{(i,j,k)} \phi(\mathbf{x}_L \mid \boldsymbol{\mu}_{(i,j,k)}, \boldsymbol{\Sigma}_{(i,j,k)}) \right)$ that produce log-likelihood function for each training matrix. All log-likelihood functions produced are compared and the maximum one is chosen. The parameters that give the maximum log-likelihood function, especially its designation in the database, reveal the speaker's true identity.

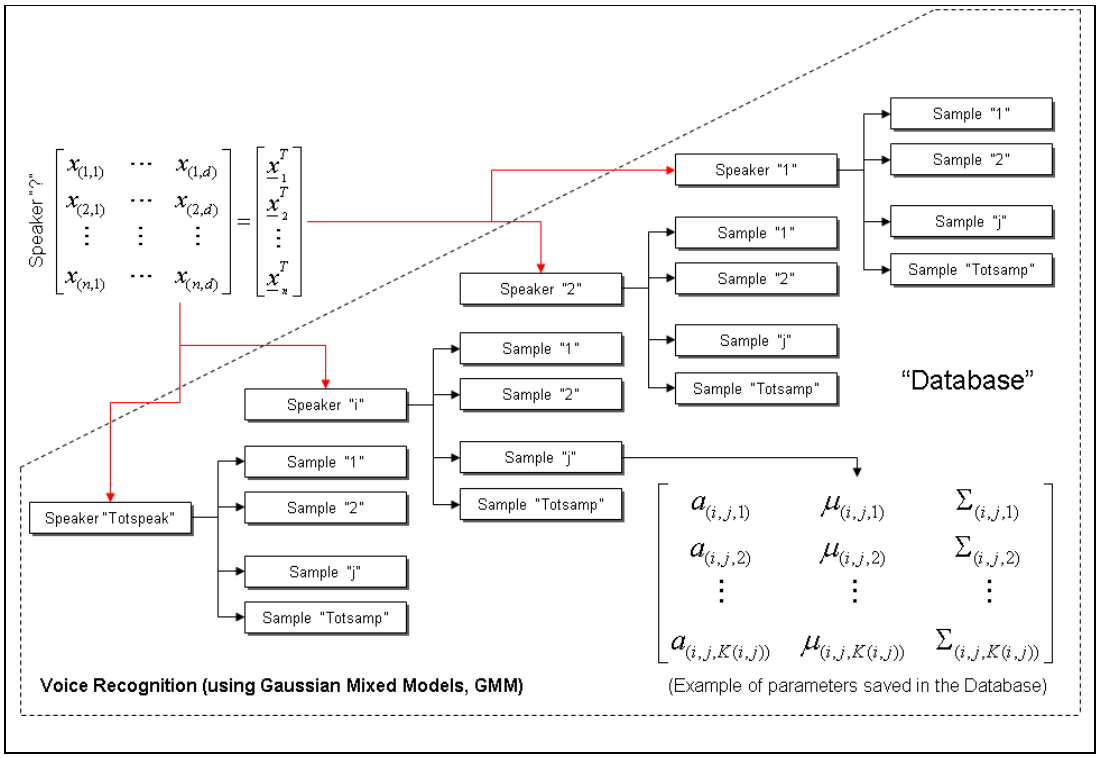


Figure 4.1 The use of GMM in voice recognition technique.

We use the idea given above to detect fraud activities in the telecommunication industry. In our case, the training data is based on customer's call behavior for a period of *Totspeak* days. The training matrix is produced from say duration of each call made for a particular day. The duration of each call made in the subsequent day will be treated as the data matrix. We are interested to identify the behavior saved in the training data that is similar to the one saved in the data matrix. This is the first step of our proposed algorithm for detecting fraud calls, which will be highlighted in Section 4.2.

This chapter, via Section 4.2, proposes a new fraud detection algorithm that uses Gaussian mixed model, a probabilistic model normally used in recognizing a person's voice in speech recognition field. Using data obtained from one of the leading telecommunication company in Malaysia (Section 4.3), we show, via Section 4.4, that

the proposed algorithm has successfully not only detected fraud calls as suspected by the company, but also identify suspicious calls which can be candidates of fraud call. The proposed algorithm is easy to implement with a great potential to be extended to detect billed (or outgoing) fraud calls and hence reduces the loss incurred by the telecommunication companies. Details of Gaussian Mixed Model (GMM) (together with Expectation Maximization, EM, algorithm) and improved EM algorithm for GMM can be found in Chapter 2 and 3, respectively.

4.2 Algorithm for detecting fraud calls

The proposed algorithm for detecting fraud calls in telecommunication involves the following steps as described in Figures 4.1 and 4.2:

Step One: For the selected customer, we perform the steps as per given in Chapter 3 (improved EM algorithm for GMM) on a given data set (which represent say the 1st day, refer to Table 4.1), and save the final estimates of parameters together with the log-likelihood functions (which gives the minimum AIC) in the text file called "database" (refer to Figure 4.3).

We repeat the process for the rest of the data sets, which represent the 2nd day till 7th day. Note that the first seven days are assumed to be "free" from fraud activities and they represent customer's behavior for the first week[1].

Step Two: The saved parameters are used on the data sets for the 8th day onwards, which include choosing the one that gives the maximum log-likelihood function and comparing the maximum log-likelihood function with the one saved in the database (Mardia et al., 1979, mentioned allocate **x** to the population which gives the largest

[1] The number of days is not fixed and can be reduced for newly registered customers.

likelihood to **x**) as shown in Figure 4.4. In the said figure, except for the last row, the first column shows the name of the file where the real data is obtained and the second

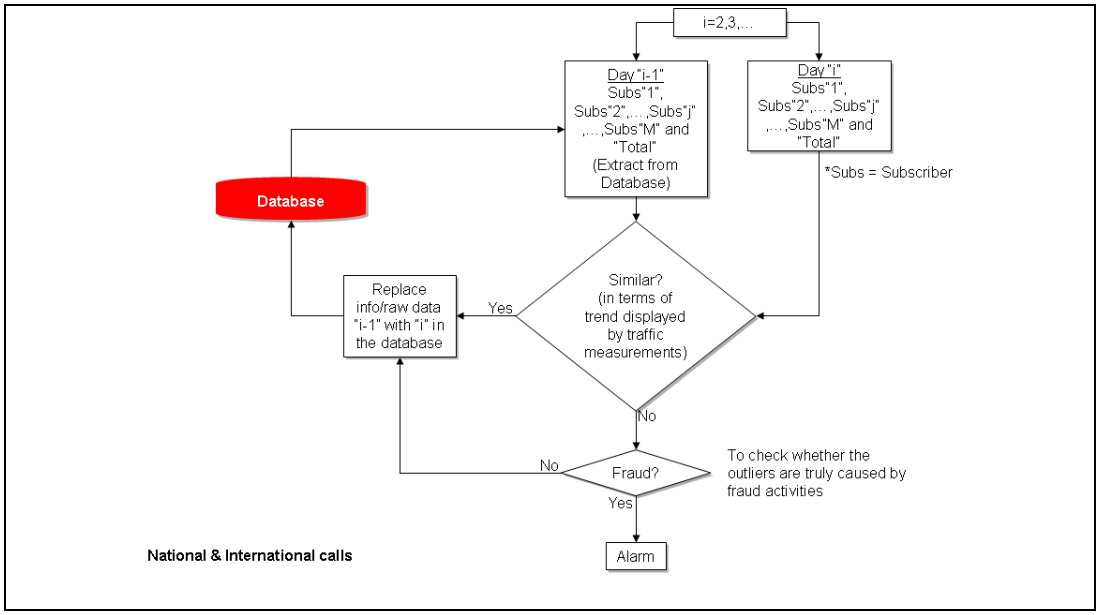


Figure 4.2 Flow-diagram showing the steps that are needed to detect fraud activities.

```
BEGIN
null;8.27205882352941E-4;7.793628360996634E-4;Lambda;Power
2_Normalized_Massaged_TMData_XXX.txt;Filename
2;No of components
2;-17.02083611317418;-1000.0;No of components;LLF;SD
0.6411573990790228;-
0.6564611309420394;0.06975010234300855;Alpha;Mu;Sigma
0.35884260092097725;1.1729234773993837;0.352458162932219;Alpha;Mu;Sigma
END
BEGIN
-7.528699885739343E-16;-
7.688105006114814;1.3737495007245424;Lambda;Power
3_Normalized_Massaged_TMData_XXX.txt;Filename
1;No of components
1;-36.38253259232894;-1000.0;No of components;LLF;SD
1.0;5.102755824719469E-16;0.961538461538469;Alpha;Mu;Sigma
END
```

Figure 4.3 Example of results from Step One.

72_XXX	*-5.10039*	*58_XXX*	*-5.10039*		
72_XXX	*-5.1276881*	*60_XXX*	*-12.240423*		
72_XXX	*-5.104467*	*66_XXX*	*-6.5368338*		
72_XXX	*-5.1316998*	*67_XXX*	*-13.662583*		
72_XXX	*-5.10039*	*69_XXX*	*-5.10039*		
72_XXX	*-5.10039*	*71_XXX*	*-5.10039*		
max llf	*-5.10039*	*(null)*	*71_XXX*	*-5.10039*	*-2.26E-15*

Figure 4.4 Example of results from Step Two (involving log-likelihood function).

column shows the calculated log-likelihood function from using the parameters saved in the database. The details of the parameters are given in the third (i.e. the name of the file) and fourth (i.e. the log-likelihood function) columns. The last row shows the maximum log-likelihood function found from comparing the values in the second column. The information corresponding to the maximum log-likelihood function in the third and fourth column is also captured.

Step Three: Dissimilarity coefficient (or *dc*) is calculated and the said coefficient is defined by

$$dc = \begin{cases} \dfrac{B-A}{B} & if \quad A,B>0 \quad B>A \\ \dfrac{A-B}{A} & if \quad A,B<0 \quad B>A \end{cases}$$

where A and B are log-likelihood functions of training data and observation, respectively. The percentage of similarity coefficient is defined by (1.0-*dc*)100%. *High dissimilarity* will result in low similarity and vice-verse. No similarity is observed when A and B are having different signs that is the percentage of similarity coefficient is zero when $B>0$ and $A<0$[2].

67_XXX	*-11.29525*	*(0.5488501726894228, Moderate_Similarity)*	*15_XXX*	*-25.03658*	*(null)*
68_XXX	*-5.10039*	*(null)*	*38_XXX*	*-5.10039*	*(1.741393151905200 5E-16,High_Similarity)*
69_XXX	*-3.489596*	*(0.757039164822191,Low_Similarity)*	*19_XXX*	*-14.36279*	*(null)*
70_XXX	*-11.10145*	*(0.19362934575160162, High_Similarity)*	*37_XXX*	*-13.76717*	*(null)*
71_XXX	*-2.693121*	*(0.43939063059603395, Moderate_Similarity)*	*33_XXX*	*-4.803917*	*(null)*
72_XXX	*-5.10039*	*(null)*	*71_XXX*	*-5.10039*	*(2.263811097476761 E-15,High_Similarity)*
73_XXX	*-11.23451*	*(0.260756177742852,High_Similarity)*	*26_XXX*	*-15.1973*	*(null)*

Figure 4.5 Example of results from Step Four (involving Similarity coefficient).

[2] In reality $A,B<0$, refer to Chapter 5.

Step Four: Similarity coefficient is assigned to *Low Similarity* group when it (after converting into percentage) is less than (or equal to) 30%, *Moderate Similarity* group when it is greater than 30% and less than (or equal to) 70% and *High Similarity* group when it is greater than 70% as exemplified in Figure 4.5.

In the said figure, the first column shows the list of files that gone through Steps One till Three, the second column shows the maximum log-likelihood function, the information about the parameters used to calculate the maximum log-likelihood function are given in the fourth and fifth columns and similarity coefficient and its group are given in the third and sixth columns (they are placed in a bracket).

The assignment of the similarity group is based on the approach taken by Turkmen (2013) for correlation coefficient, denoted by r, to describe the strength of relationship: None: r from -0.1 to 0.1; Weak: r from 0.1 to 0.3 or from -0.3 to -0.1; Moderate: r from 0.3 to 0.5 or from -0.5 to -0.3; Strong: r from 0.5 to 1.0 or from -1.0 to -0.5.

Step Five: If similarity coefficient between maximum log-likelihood function and the one saved in the database is 0.3 and below (or in terms of percentage, 70 and above) then no updating is performed on the database.

Step Six: Updating is performed on the database if, in terms of percentage, less than 70. Updating involves performing the steps as per given in Section 4.2.2 on the data set and saving final estimates of parameters together with log-likelihood function produced in the database.

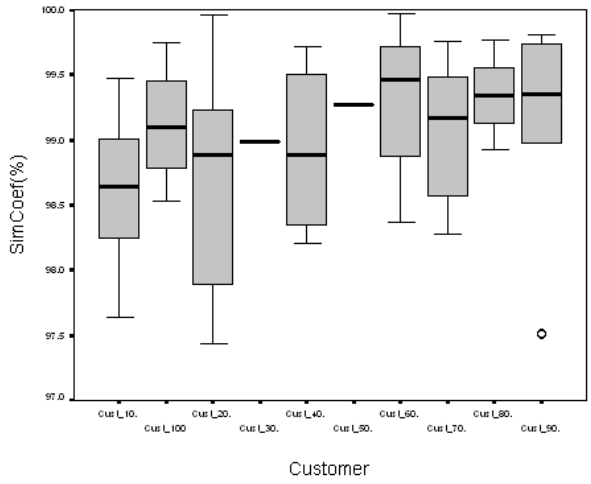
In this chapter we will compare the results produced from the proposed algorithm using one variable with two variables.

4.2.1 The performance of the algorithm

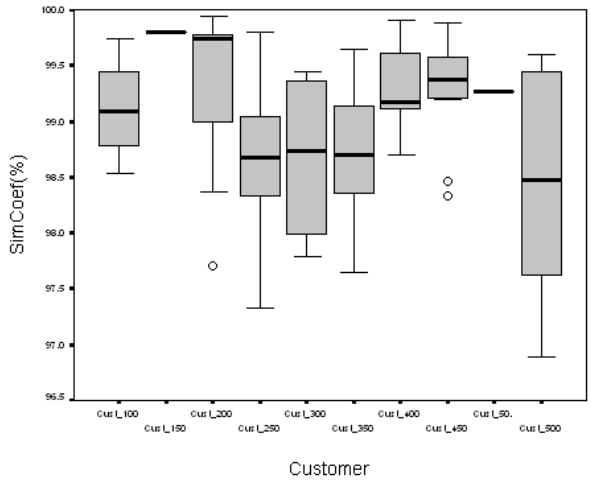
For each of the five hundred customers and for each of the seven days, the number of components are chosen at random where the maximum number of components is fixed at three (3). If three components are chosen, the values of a_1, a_2 and a_3 are chosen at random from 0.1,…,0.9 where $a_1 + a_2 + a_3 = 1$; the values of μ_1, μ_2 and μ_3 are chosen at random from 0.25,0.50,0.75,1.0,2.0,3.0 where the means of two or more components are chosen such that no two or more means are the same; and the values of σ_1, σ_2 and σ_3 are chosen at random from $\frac{1}{2},\frac{1}{3},\cdots,\frac{1}{10},1,2,3,\cdots,10$. Similar steps are taken if less than three components are chosen. The chosen values (that represent a model) are used in $z_j = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \cos 2\pi u_{j+1}$ and $z_{j+1} = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \sin 2\pi u_{j+1}$ where $u_j, u_{j+1} \sim U(0,1)$ (Box and Muller, 1958) to generate (one thousand) simulation data. For each day starting the 8th till 28th, simulation data is generated using either one of the seven models (which represents the first condition where the performance of the algorithm is evaluated by this condition) or other model derived from repeating the steps as mentioned above (which represents the second condition where the performance of this condition is partly explained in the upcoming/following section).

For each customer, we perform all of the steps in the algorithm (altogether they are six) on models that represent 1st till 28th day. The similarity coefficients, derived from performing the algorithm's third step on models produced from the first condition and represent 8th day onwards, are displayed using Box plot as shown in Figure 4.6 (due to its size, only a few of the customers are displayed here). For ease of graphical

presentation, the similarity coefficient is converted into percentage. All of them (i.e. the percentage of similarity coefficient) exceed 90.



(a)



(b)

Figure 4.6 Box plot for customers (a) 10, 20, 30,…,100 and (b) 50, 100, 150,…,500 on the x-axis and SimCoef(%) on the y-axis is a short-form for similarity coefficient in percentage.

Our observation is supported by the histogram in Figure 4.7, which is derived from using all of the (five hundred customers) similarity coefficients, where the mean

and standard deviation are 99.07 and 0.76, respectively. We find $D_i = \theta_i - \hat{\theta}_i$ where θ_i equals to 100.0 and $\hat{\theta}_i$ is the similarity coefficient used to produce Figure ii, $i = 1,2,...,m$. The sample mean and standard deviation of D_i are computed using formulas $\overline{D} = \frac{1}{m}\sum_{i=1}^{m} D_i$ and $S_D = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}\left(D_i - \overline{D}\right)^2}$, respectively. The similarity coefficients are considered good if $\overline{D}$ is close to zero, indicating small biases observed in the similarity coefficient results, and S_D is also close to zero, indicating that the similarity coefficients are concentrated around their respective true values. The sample mean and standard deviation (of D_i, $i = 1,2,...,m$) are 0.93 and 0.76, respectively.

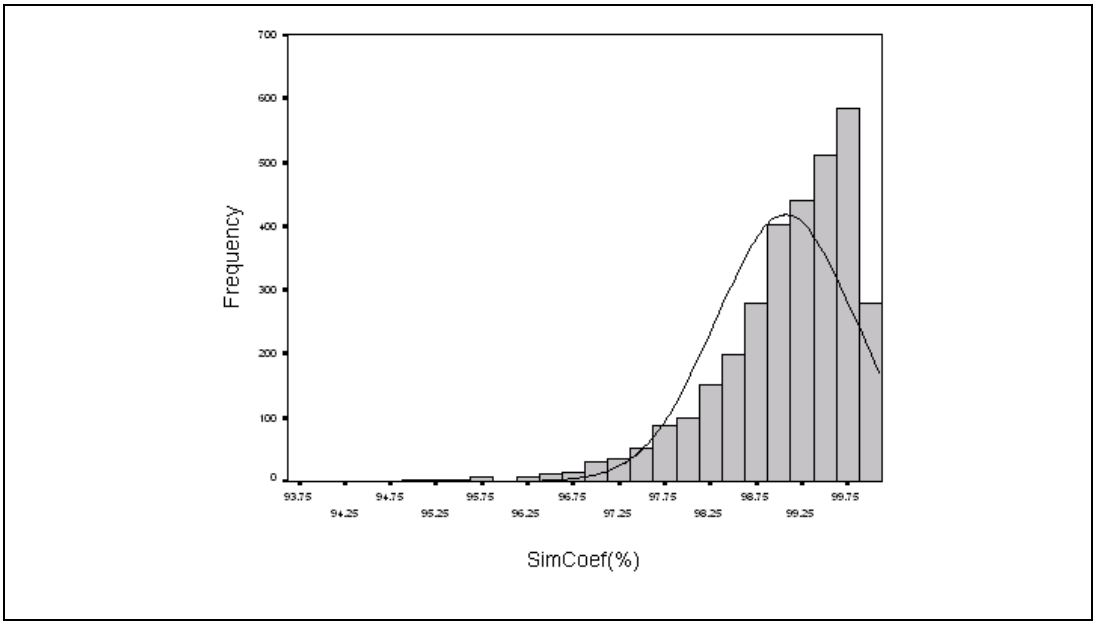


Figure 4.7 The histogram of Frequency versus SimCoef(%). SimCoef(%) on the y-axis is a short-form for similarity coefficient in percentage.

The steps in the algorithm (i.e. second till sixth) are performed on models that represent 8th days onwards. We are interested on models that are produced from the first condition. "A" denotes the total number of models produced from the first condition that is correctly classified. "B" denotes the total number of models that is

produced from the first condition. Count(%) equals to (A/B)%. Figure 4.8 shows the histogram of Frequency against Count(%) where the mean and standard deviation are 61.2 and 22.5, respectively.

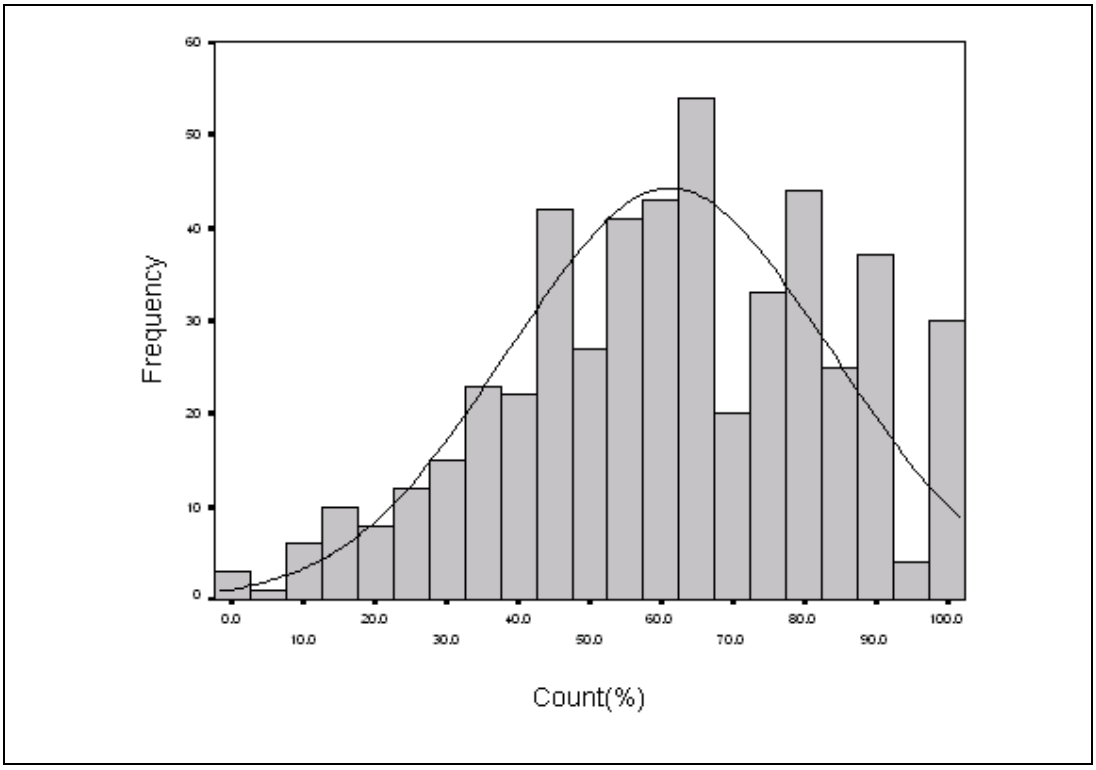


Figure 4.8 The histogram of Frequency against Count(%).

4.2.2 The characteristics of the similarity coefficient

The characteristics of similarity coefficient are best described using Figure 4.9. Let A represents call behavior for a customer collected on the first day and B represents call behavior collected on the second day. The probability density function $f_1\left(y_i \mid \mu_1, \sigma_1^2\right)$ for some of y_i is greater than zero and the rest is close to zero. We calculate $\frac{llf_A}{llf_B} = h$ where $0 \le h \le 1$, $llf_B = \sum_{i=1}^{n_2} \log\left(f_1\left(y_i \mid \mu_1, \sigma_1^2\right)\right)$,

$llf_A = \sum_{i=1}^{n_1} \log\left(f_1\left(x_i \mid \mu_1, \sigma_1^2\right)\right)$, $llf_A, llf_B < 0$, $n_1 \Psi n_2$ where $\Psi \in \{<,>,=\}$ and $llf_A > llf_B$.

Note that the parameters μ_1, σ_1^2 and llf_A of A are kept inside the database. Using these information on B, the percentage of similarity coefficient, denoted by $h\%$, is close to zero due to $f_1\left(y_i \mid \mu_1, \sigma_1^2\right) \approx 0.0$ for the majority of y_i. In other words, call behavior on the first day is dissimilar to the second day.

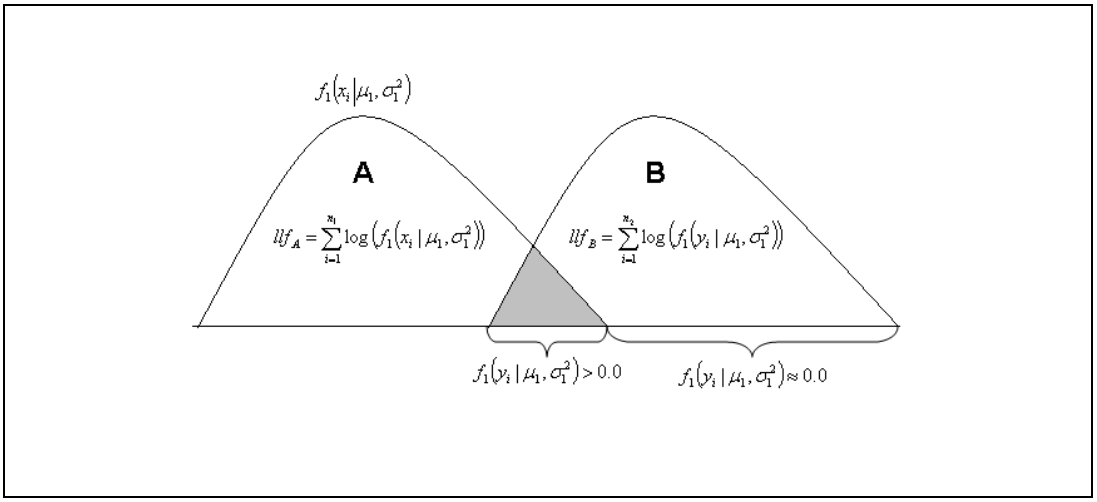


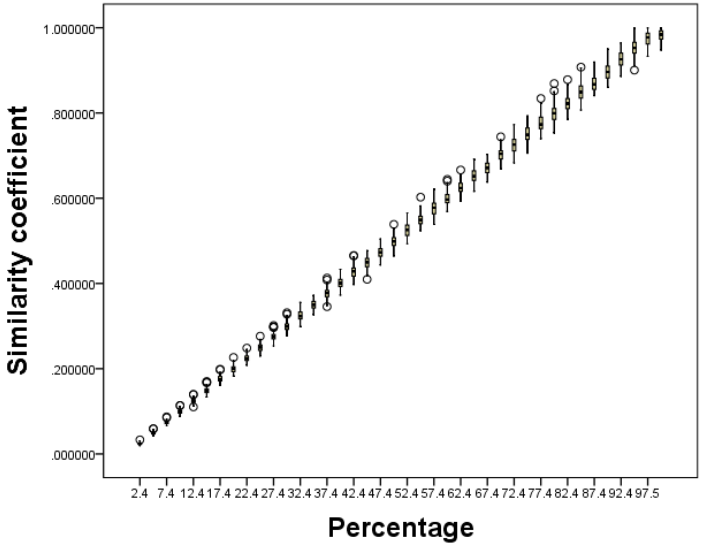
Figure 4.9 A represents the probability density function derived from the histogram of call behavior for a customer collected on the first day ($x_i, i = 1,2,...,n_1$) and B represents the histogram of call behavior collected on the second day ($y_i, i = 1,2,...,n_2$).

Other characteristics are described by performing the following processes and repeat them 100 times for each $n_2 = \Omega n_1$ where $\Omega = 1.0,0.975,0.95,0.925,...,0.025$ (these values are converted into percentage for ease of graphical presentation) and $n_1 = 1000$.

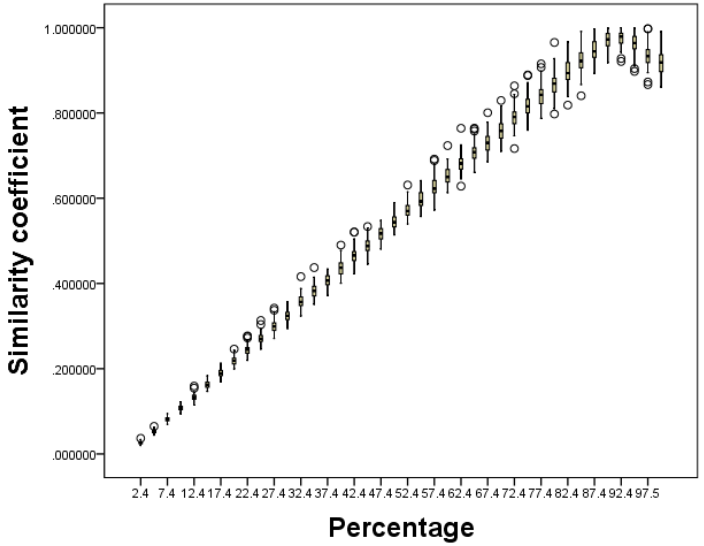
We generate n_1 simulation data from $N(0,1)$ using Box and Muller Transformation (Box and Muller, 1958). We treat the simulation data as training data (or td). We calculate the mean (denoted by μ_{td}), variance (denoted by σ_{td}^2) and log-likelihood function (denoted by llf_{td}).

We generate n_2 simulation data from $N(\mu_O,1)$ using Box and Muller Transformation (Box and Muller, 1958) for each $\mu_O = 0.0,0.1,...,10.0$. We treat the simulation data as observation (or O). Using μ_{td} and σ_{td}^2 from the above, we calculate the log-likelihood function (denoted by llf_O) and similarity coefficient $\frac{llf_O}{llf_{td}}$.

Figures 4.10(a)-(h) give the plot of similarity coefficient against the percentage of sample ratio (i.e. $\Omega \times 100\%$) for all values of μ_O considered. Figure 4.10(a) shows for the case $\mu_O = 0.0$, $n_1 = 1000$, percentage (or $\Omega\%$) equals to 2.4 (i.e. $n_2 = 24$), x_i is a training data where $i = 1,2,...,n_1$, y_i is an observation where $i = 1,2,...,n_2$, both training data and observation are randomly generated, the mean and standard deviation for $llf_O = \sum_{i=1}^{n_2} \log\left(f_1\left(y_i \mid \mu_{td}, \sigma_{td}^2\right)\right)$ are -34.27 and 3.63, respectively, the mean and standard deviation for $llf_{td} = \sum_{i=1}^{n_1} \log\left(f_1\left(x_i \mid \mu_{td}, \sigma_{td}^2\right)\right)$ are -1415.99 and 23.34, respectively and the similarity coefficient is close to zero. The similarity coefficient is close to one when percentage (or $\Omega\%$) equals to 100 (i.e. $n_2 = 1000$) where the mean and standard deviation for $llf_O = \sum_{i=1}^{n_2} \log\left(f_1\left(y_i \mid \mu_{td}, \sigma_{td}^2\right)\right)$ are -1424.43 and 22.36, respectively and the

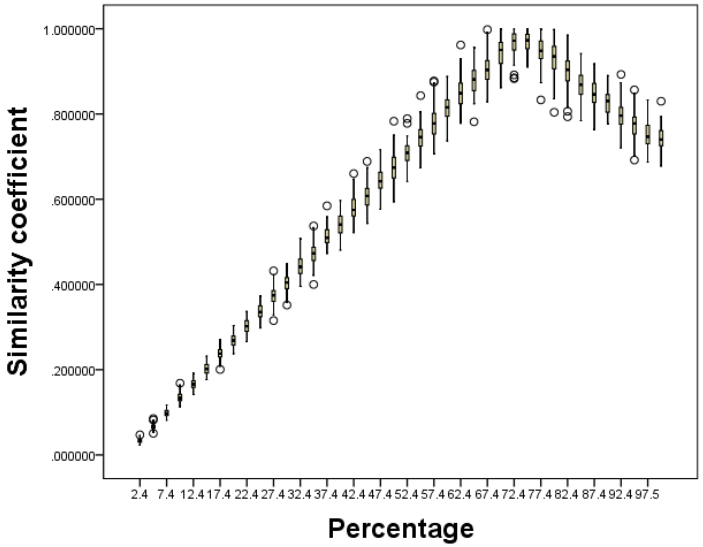


(a) $\mu_o = 0.0$

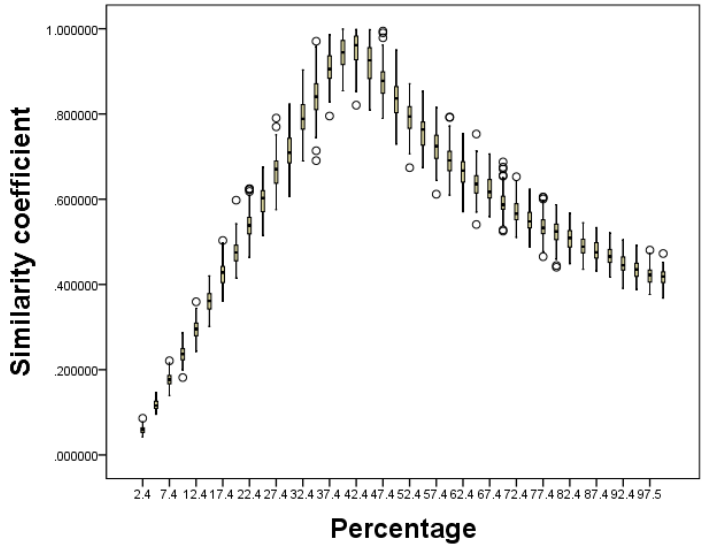


(b) $\mu_o = 0.5$

Figure 4.10 Similarity coefficient is plotted against percentage using Box plot for (a) μ_o=0.0, (b) μ_o=0.5, (c) μ_o=1.0, (d) μ_o=2.0, (e) μ_o=4.0, (f) μ_o=6.0, (g) μ_o=8.0 and (h) μ_o=10.0.

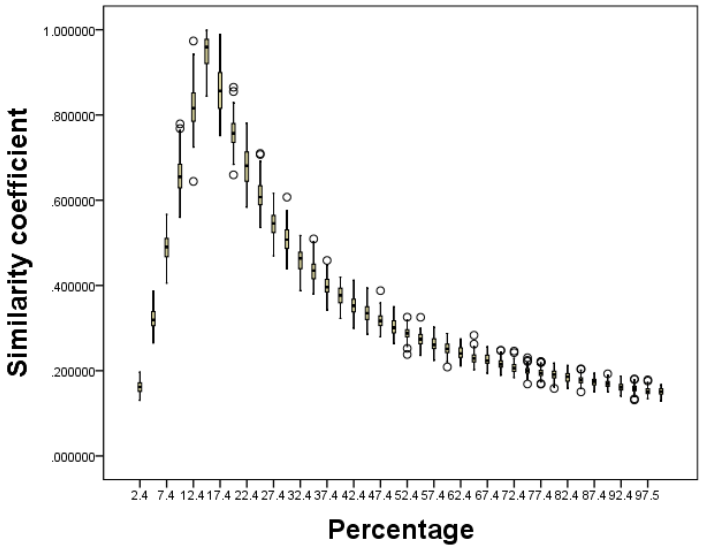


(c) μ_O=1.0

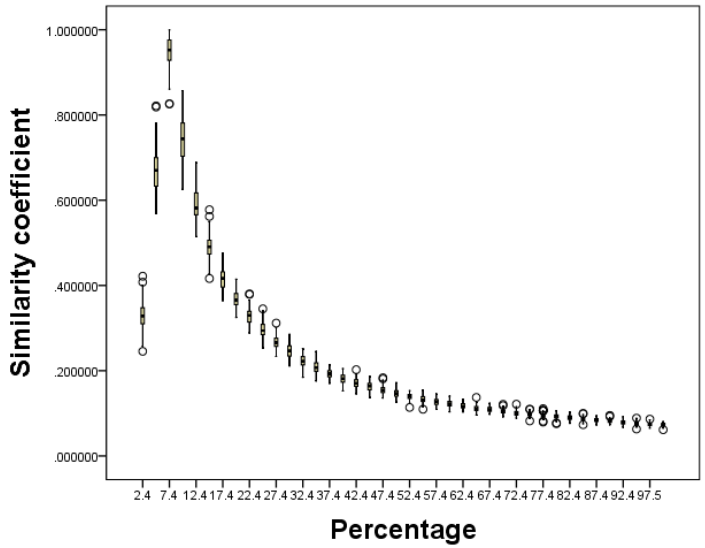


(d) μ_O=2.0

Figure 4.10 Continued.

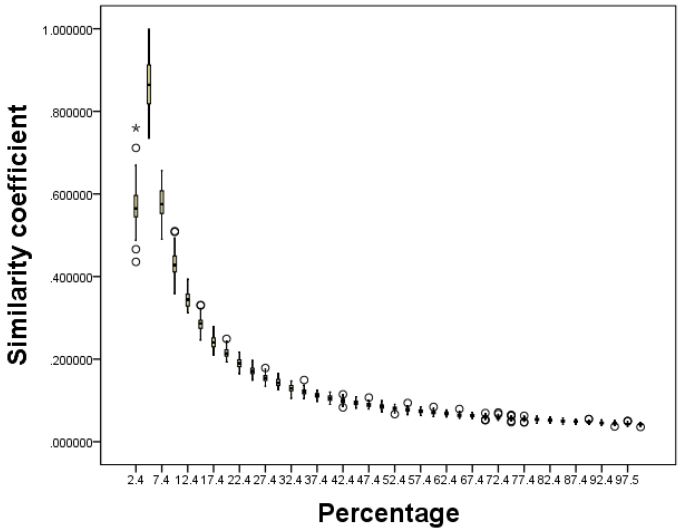


(e) μ_o =4.0

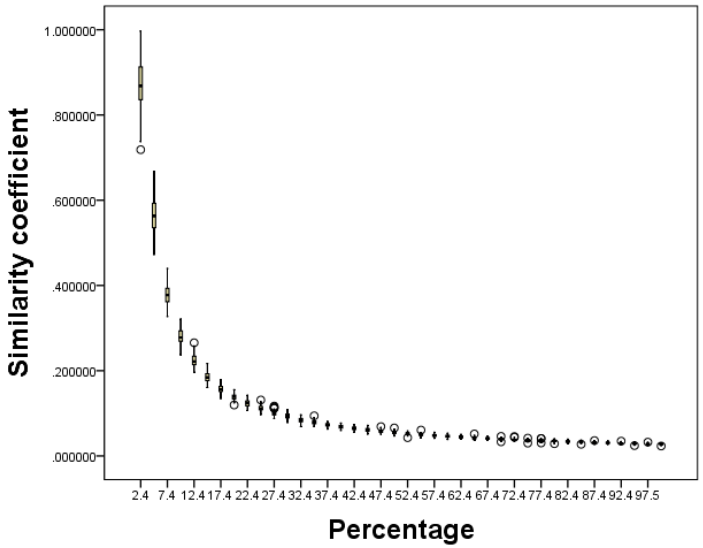


(f) μ_o =6.0

Figure 4.10 Continued.



(g) μ_O=8.0



(h) μ_O=10.0

Figure 4.10 Continued.

mean and standard deviation for $llf_{td} = \sum_{i=1}^{n_1} \log\left(f_1\left(x_i \mid \mu_{td}, \sigma_{td}^2\right)\right)$ are -1419.90 and 19.90, respectively.

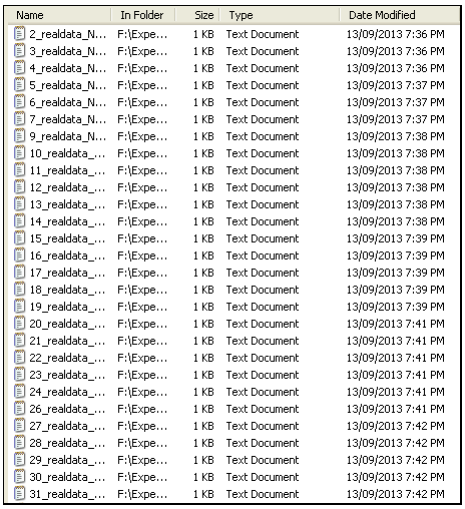
4.3 Data

The call detail record, which was supplied by Telekom Malaysia Berhad (henceforth, TM), consists of calls made by the customers and they were victims of fraud activities. Altogether there are 18 customers and they are labeled as A till R to ensure confidentiality. We use the same format of call detail record for each TM's customer as described before in Table 4.1. We performed several steps to get the desired format e.g. group the real data according to service no, find the country that matches with the country code and sort the real data according to seize time.

To make our job of handling the real data for the TM customers easier, we divided them into several parts and saved in the following format: (for each customer) 1, 2, 3, 4,… represent fn(1), fn(2), fn(3), fn(4),… and date(1), date(2), date(3),… where fn is a short-form for filename and date(1) < date(2) < date(3) < …. as exemplified in Figure 4.11.

Table 4.1 An example of TM's customer call detail record.

No	Service No	Dialed Digit	…	Seize Time1	Duration1	Duration2
281	XXX	yyy	…	31/03/2011 10:07	000255 00	0.0020255
282	XXX	yyy	…	31/03/2011 15:24	000054 00	6.25E-04
283	XXX	yyy	…	31/03/2011 16:16	000045 00	5.21E-04
284	XXX	yyy	…	31/03/2011 16:37	000556 00	0.0041204



Name	In Folder	Size	Type	Date Modified
2_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:36 PM
3_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:36 PM
4_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:36 PM
5_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:37 PM
6_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:37 PM
7_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:37 PM
9_realdata_N...	F:\Expe...	1 KB	Text Document	13/09/2013 7:38 PM
10_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:38 PM
11_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:38 PM
12_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:38 PM
13_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:38 PM
14_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:38 PM
15_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:39 PM
16_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:39 PM
17_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:39 PM
18_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:39 PM
19_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:39 PM
20_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:41 PM
21_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:41 PM
22_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:41 PM
23_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:41 PM
24_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:41 PM
26_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:41 PM
27_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:42 PM
28_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:42 PM
29_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:42 PM
30_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:42 PM
31_realdata_...	F:\Expe...	1 KB	Text Document	13/09/2013 7:42 PM

Figure 4.11 Files created for Customer A.

4.4 Results

TM's (current) system, which uses non-GMM method and customer's call detail record (details of the system will not be revealed to ensure confidentiality), detected fraud activity on the 15th of November 2011 for the customers mentioned in Section 4.3. From our analysis of customers D and Q, by studying the duration and real data on the 15th of November 2011 that are saved in filename 63, similarity coefficient is assigned to High Similarity group. Similar results are obtained when using two variables namely duration and call charging (or billing). An example is given in Figure 4.12.

For each customer mentioned above (using one variable, i.e. duration, and two variables, i.e. duration and call charging or billing), we find the number of similarity coefficient assigned to Low Similarity group, which we believe they have close connection to fraud activity, and convert them into percentage. For example, $x_1,...,x_k$

represent the percentage of similarity coefficient assigned to Low Similarity group for k customers when using one variable and $y_1,...,y_k$ when using two variables where $k = 18$. Statistical methods such as summary statistics are applied to $x_1,...,x_k$ and $y_1,...,y_k$. The k customers are grouped together based on which group the percentage of similarity coefficient is assigned to when using one and two variables on the 15th of November 2011. For example, Step Four results for customer D especially on the 15th of November 2011 show, when using one variable, log-likelihood function and similarity coefficient (after converting into percentage) equal to -5.503 and 86.69%, respectively. High Similarity group (denoted by H) is assigned to customer D. Similar results are obtained, in terms of assigning High Similarity group (denoted by H) to customer D, when using two variables where log-likelihood function and similarity coefficient (after converting into percentage) equal to -15.996 and 85.27%, respectively.

HH is the group assigned to customers D and Q. By studying the duration, the minimum and maximum percentages of similarity coefficients assigned to Low Similarity group for all customers mentioned above are 5 and 22, respectively. The minimum and maximum percentages are reduced to 3 and 12, respectively, when using two variables namely duration and call charging (or billing).

From our (second) analysis of customers H, K, O and P, by studying the duration and real data on the 15th of November 2011 that are saved in filename 57, 73, 51 and 55, respectively, similarity coefficient is assigned to Moderate Similarity group. Similarity coefficient is assigned to High Similarity group when using two variables namely duration and call charging (or billing) thus upgrading the previous group. MH is the group assigned to customers H, K, O and P. By studying the duration, the average

percentage of similarity coefficient assigned to Low Similarity group is 8 bounded by 0 and 14. They spread around the average (or standard deviation) at 5.9. When using two variables namely duration and call charging (or billing), the average is lowered to 5.5 bounded by 0 and 9. The standard deviation equals to 3.9. Note that no similarity coefficient is assigned to Low Similarity group for customers O for one variable (namely duration) and P for two variables (namely duration and call charging or billing).

From our (third) analysis of customers R, E, F and I, by studying the duration and real data on the 15th of November 2011 that are saved in filename 68, 75, 67 and 68 respectively, similarity coefficient is assigned to Low Similarity group. Similar results are obtained (i.e. similarity coefficient is assigned to Low Similarity group) when using two variables namely duration and call charging (or billing). An example is given in Figure 4.13. LL is the group assigned to customers R, E, F and I. The average percentage of similarity coefficient assigned to Low Similarity group for duration is 17.5 where it's left and right wings are 10 and 21, respectively. They spread around the average (or standard deviation) at 5.1. When using two variables namely duration and call charging (or billing), most of the values are lowered. For example, the average is 16.8, minimum and maximum values are 5 and 41, respectively, and standard deviation equals to 16.5. The results of the fourth till the seventh analysis can be found in Table 4.2. For MM, an example is given in Figure 4.14.

4.5 Discussion

In the previous chapters, we introduced the GMM, EM algorithm and algorithm for determining the number of components that incorporates kernel method. We also

introduced in the previous section an algorithm for detecting fraud calls. We used them on two variables namely duration and call charging (or billing) of the real data (TM customers), which revealed interesting results.

Table 4.2 Summary statistics for groups HM till MM. Note that customers are given in the bracket; SD and Var are short forms for standard deviation and variable, respectively.

Group	HM (A,L,N,J)		LM (B)	
	1 Var	2 Vars	1 Var	2 Vars
Min	2	3	14	3
Average	13	14.5	14	3
SD	8.1	9.0	-	-
Max	20	25	14	3

Group	LH (G)		MM (C,M)	
	1 Var	2 Vars	1 Var	2 Vars
Min	14	7	1	7
Average	14	7	2.5	9.5
SD	-	-	2.1	3.5
Max	14	7	4	12

TM's (current) system detected fraud activity on the 15th of November 2011. If one variable is used in the proposed algorithm, 33% of 18 TM's customers used in this study support the findings made by TM's system. The rest i.e. 33% of the customers are assigned to Moderate Similarity group and 33% to High Similarity group. If two variables are used in the proposed algorithm, 22% of 18 TM's customers used in this study support the findings made by TM's system. The rest i.e. 39% of the customers are assigned to Moderate Similarity group and 39% to High Similarity group.

Furthermore, 22% of 18 TM's customers are downgraded (i.e. from High Similarity group to Moderate Similarity group), 33% of 18 TM's customers are upgraded (i.e. 6% from Low Similarity group to Moderate Similarity group, 6% from Low Similarity group to High Similarity group and 22% from Moderate Similarity group to High

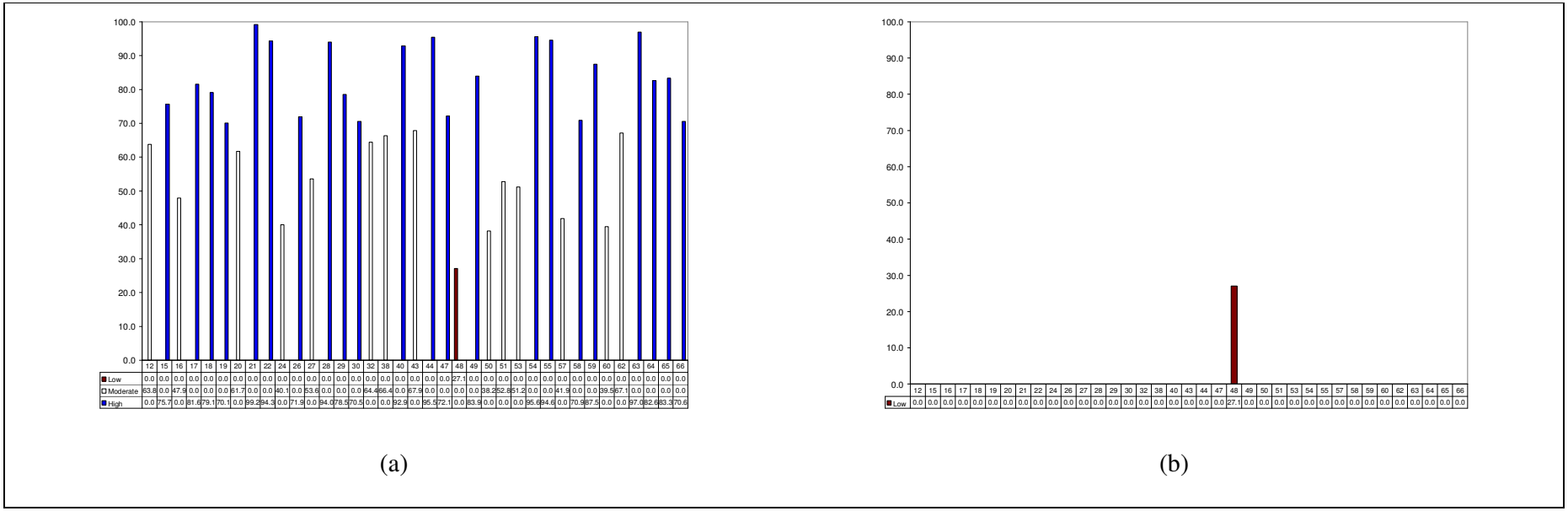


Figure 4.12 Similarity coefficient (after converting into percentage) together with its classification for customer Q (using two variables namely duration and call charging or billing): (a) Overall and (b) Low Similarity only.

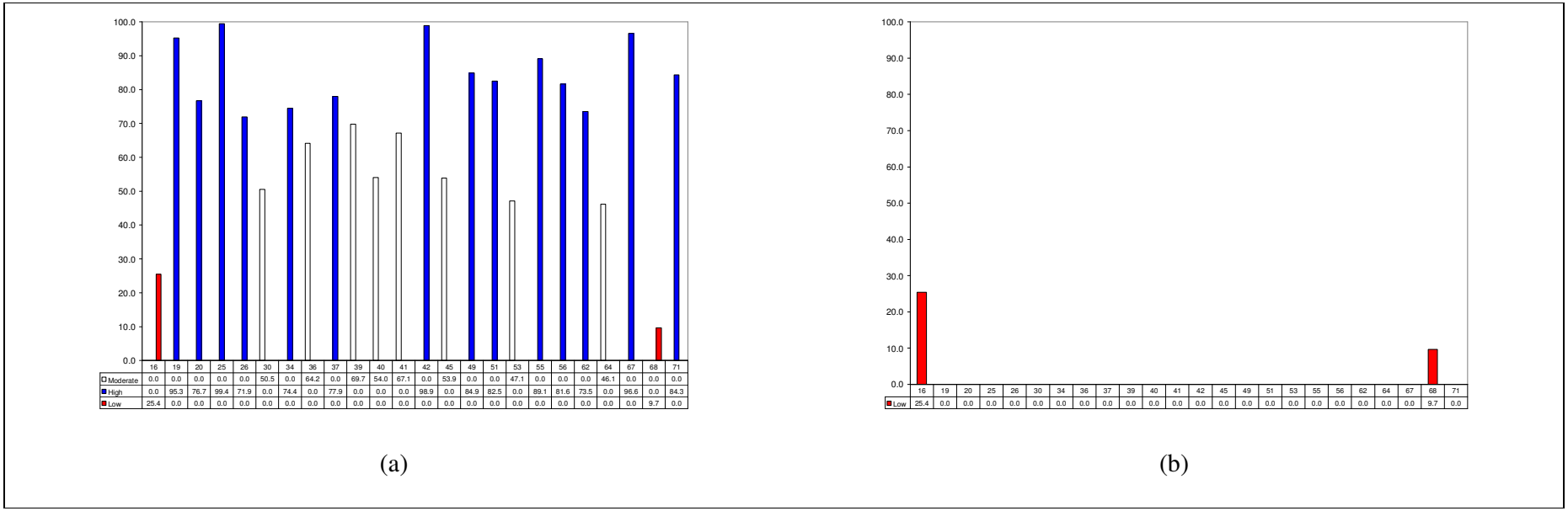


Figure 4.13 Similarity coefficient (after converting into percentage) together with its classification for customer I (using two variables namely duration and call charging or billing): (a) Overall and (b) Low Similarity only.

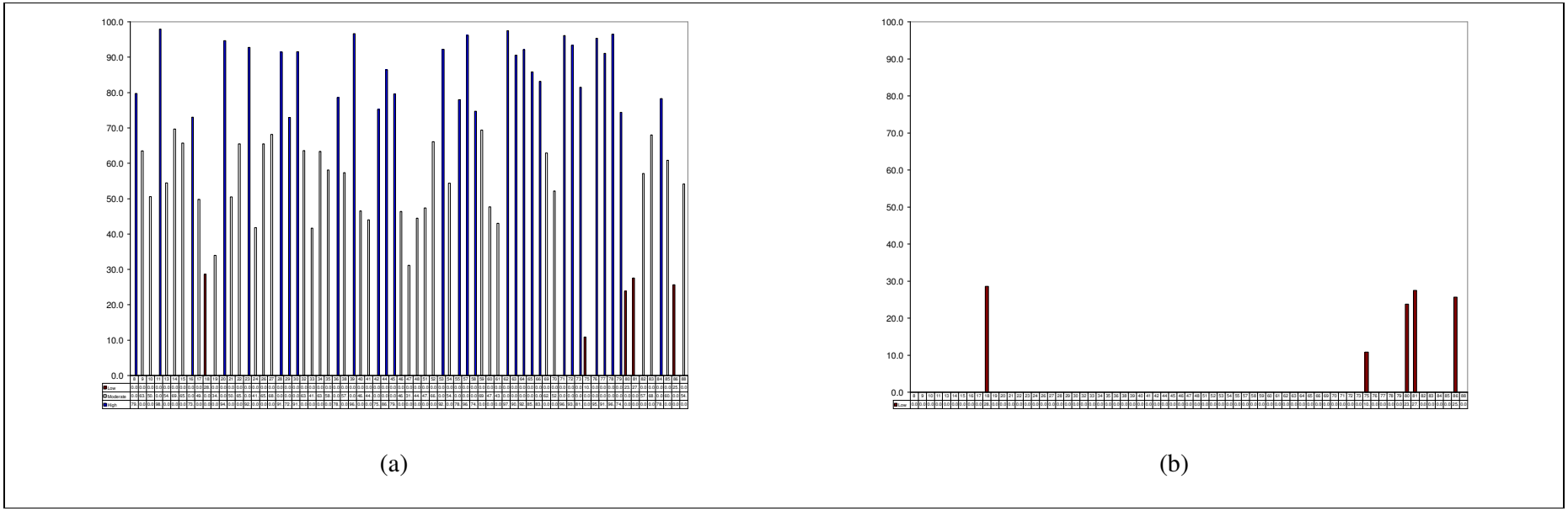


Figure 4.14 Similarity coefficient (after converting into percentage) together with its classification for customer C (using two variables namely duration and call charging or billing): (a) Overall and (b) Low Similarity only.

Similarity group) and 44% of 18 TM's customers are unchanged (i.e. 22% at Low Similarity group, 11% at Moderate Similarity group and 11% at High Similarity group). Similarity group for 11% of 18 TM's customers depends on whether using one or two variables (e.g. Low Similarity group when using one variable).

No similarity coefficient is assigned to Low Similarity group for customer P (hence no fraud activity is observed or expected) when using two variables (i.e. duration and call charging or billing) and no similarity coefficient is assigned to Low Similarity group for customer O (hence no fraud activity is observed or expected) when using one variable (i.e. duration).

The average number of similarity coefficient assigned to Low Similarity group when using two variables is 11.2% (meaning, fraud activities might happened earlier than 15th of November 2011).

The results show the effectiveness of the proposed algorithm in detecting Low Similarity group (before and on the 15th of November 2011, which we believe they have close connection to fraud activity). Future research work will involve the following: the proposed algorithm will be tested on a bigger number of customers, the handling of files with small data points, twenty four (24) hours period for collecting customer's call detail record will be divided into 3 parts (sub-periods), similarity coefficient that falls under Moderate Similarity group will be further investigated (for possible fraud activities) and the use of other variables such as type of call (domestic and international) as shown in Figure 4.2.

It was mentioned by Schonlau et al. (2001) (computer) intrusion can be detected by the statistical methods in any circumstances even in difficult ones and this field of study (i.e. computer intrusion detection) offers many challenges and opportunities to statistics and statisticians. Bolton and Hand (2002), page 246, generalized by saying "Fraud detection is an important area, one in many ways ideal for the application of statistical and data analytic tools and one where statisticians can make a very substantial and important contribution".

CHAPTER 5

EXPLORING THE USE OF HYPOTHESIS TESTING IN DETERMINING THE NUMBER OF COMPONENTS IN GAUSSIAN MIXED MODEL

5.1 Introduction

The improvement of EM algorithm for GMM involves the use of Kernel method (Silverman, 1986) to determine the number of components and to find means as initial values to start EM algorithm for GMM has been described in Chapter 3. It also involves the calculation of the log-likelihood function and Akaike Information Criteria (AIC) (Akaike, 1974) and the comparison of all AICs where the minimum value gives the true (or correct) number of components. Details of Gaussian Mixed Model (GMM) and Expectation Maximization (EM) algorithm can be found in Chapter 2.

This chapter, via Section 5.2, shows the successful derivation of hypothesis testing in the determination of the number of components in GMM, which is an important process as highlighted by a number of authors (for example Schlattmann, 2003, and Wang et al., 2004), and the performance of the hypothesis testing. The comparison of its results with those of AIC will be highlighted in Section 5.3.

The development of this method enables one to determine the number of components in the GMM in an objective way.

5.2 Hypothesis testing

5.2.1 Introduction to property 1 and 2

Property 1 (or Prop 1) is defined by $0<\left(\prod_{i=1}^{n}\phi_i\right)\left(\sum_{j=1}^{n}\phi_j\right)<n$ where it is obtained by using $0<\phi_i<1$ where $i=1,2,...,n$, $0<\left(\prod_{i=1}^{n}\phi_i\right)<1$, $0<\left(\sum_{j=1}^{n}\phi_j\right)<n$ and $0<\left(\prod_{i=1}^{n}\phi_i\right)\left(\sum_{j=1}^{n}\phi_j\right)<\left(\sum_{j=1}^{n}\phi_j\right)<n$. Apply logarithm to $0<\left(\prod_{i=1}^{n}\phi_i\right)\left(\sum_{j=1}^{n}\phi_j\right)<n$, we get $\log\left(\left(\prod_{i=1}^{n}\phi_i\right)\left(\sum_{j=1}^{n}\phi_j\right)\right)<\log(n)$ or $\log\left(\left(\sum_{j=1}^{n}\phi_j\right)\right)<\log(n)-\log\left(\left(\prod_{i=1}^{n}\phi_i\right)\right)$ (Spiegel, 1974).

An example is given in Figure 5.1.

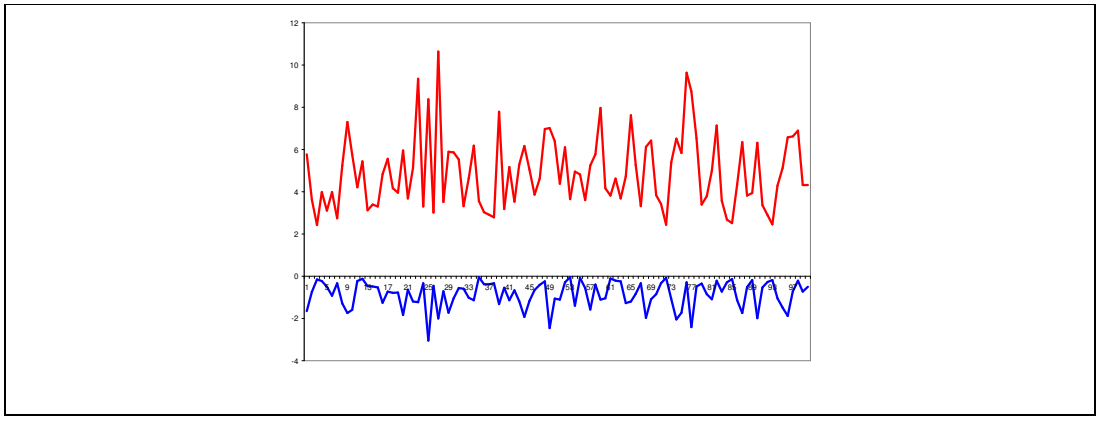


Figure 5.1 The first (i.e. upper) and second (i.e. lower) lines represent $\log(2)-\log\left(\left(\prod_{i=1}^{2}\phi_i\right)\right)$ and $\log\left(\left(\sum_{j=1}^{2}\phi_j\right)\right)$, respectively. X-axis represents 100 samples (generated by using random numbers).

Property 2 (or Prop 2) is defined by $\left(\frac{n}{n}\right)\left(\prod_{i=1}^{n}\phi_i\right)<\left(\sum_{j=1}^{n}\phi_j\right)$ or $\left(\prod_{i=1}^{n}\phi_i\right)<\left(\sum_{j=1}^{n}\phi_j\right)$ where it is obtained by using $0<\phi_i<1$ where $i=1,2,...,n$, $0<\left(\prod_{i=1}^{n}\phi_i\right)<1$, $0<\left(\sum_{j=1}^{n}\phi_j\right)<n$ and $\left(\frac{1}{n}\right)\left(\prod_{i=1}^{n}\phi_i\right)<\left(\prod_{i=1}^{n}\phi_i\right)<\phi_j$ or $\left(\frac{1}{n}\right)\left(\prod_{i=1}^{n}\phi_i\right)<\phi_j$ where $j=1,2,...,n$ (Spiegel, 1974). An example is given in Figure 5.2.

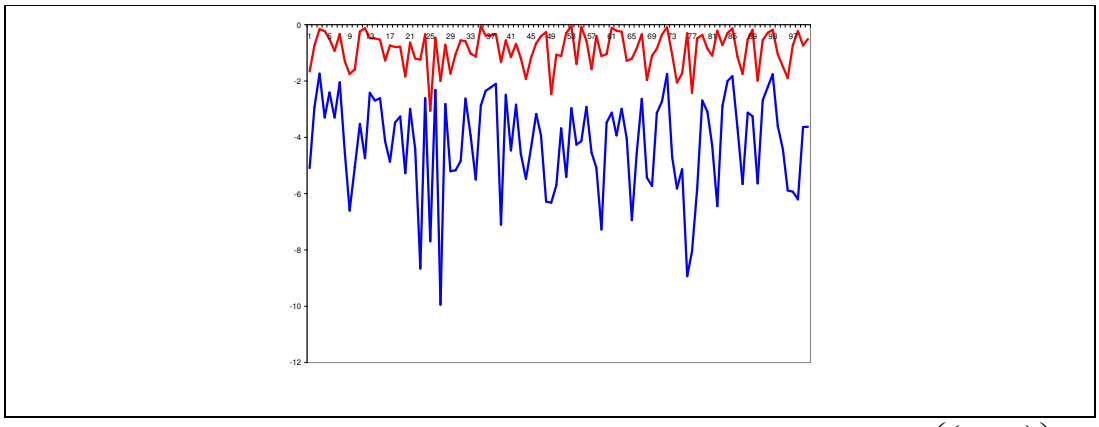


Figure 5.2 The first (i.e. upper) and second (i.e. lower) lines represent $\log\left(\left(\sum_{j=1}^{2}\phi_j\right)\right)$ and $\log\left(\left(\prod_{i=1}^{2}\phi_i\right)\right)$, respectively. X-axis represents 100 samples (generated by using random numbers).

5.2.2 The derivation of the hypothesis testing

AIC used in the improvement of EM algorithm for GMM can be replaced by hypothesis testing $H_0:\boldsymbol{\theta}=\left(\begin{bmatrix}a_1\\ \vdots\\ a_k\end{bmatrix},\begin{bmatrix}\boldsymbol{\mu}_1\\ \vdots\\ \boldsymbol{\mu}_k\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma}_1\\ \vdots\\ \boldsymbol{\Sigma}_k\end{bmatrix}\right)$ versus $H_1:\boldsymbol{\theta}^*=\left(\begin{bmatrix}a_1^*\\ \vdots\\ a_{k^*}^*\end{bmatrix},\begin{bmatrix}\boldsymbol{\mu}_1^*\\ \vdots\\ \boldsymbol{\mu}_{k^*}^*\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma}_1^*\\ \vdots\\ \boldsymbol{\Sigma}_{k^*}^*\end{bmatrix}\right)$ where $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^*$ are final estimates of parameters corresponding to k and k^*,

respectively. k and k^* are number of parameters where $k,k^*=1,2,\ldots,K$ (and preferably $k^*>k$). The likelihood ratio statistics for testing the above hypothesis is defined by $-2\log\lambda = 2\sum_{j=1}^{n}\log\left(\frac{\sum_{i=1}^{k^*} a_i^*\phi(\mathbf{x}_j \mid \boldsymbol{\mu}_i^*, \boldsymbol{\Sigma}_i^*)}{\sum_{i=1}^{k} a_i\phi(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}\right)$ (Mardia et al., 1979). It can be written as

$$-2\log\lambda = 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{(\mathbf{x}_j-\boldsymbol{\mu}_i^*)^t \boldsymbol{\Sigma}_i^{*-1}(\mathbf{x}_j-\boldsymbol{\mu}_i^*)}{2}\right)\right) - 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\exp\left(-\frac{(\mathbf{x}_j-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_j-\boldsymbol{\mu}_i)}{2}\right)\right). \quad (5.1)$$

We apply $\log\left(\sum_{i=1}^{h}\phi_i\right) < \log\left(h\left(\prod_{i=1}^{h}(\phi_i)\right)^{-1}\right)$ where $0<\phi_i<1, i=1,2,\ldots,h$ (see Section 5.2.1) and $\sum_{i=1}^{k^*} a_i^* f_i^* < 1$ [3] to the first term of equation (5.1) yielding

$$\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{(\mathbf{x}_j-\boldsymbol{\mu}_i^*)^t \boldsymbol{\Sigma}_i^{*-1}(\mathbf{x}_j-\boldsymbol{\mu}_i^*)}{2}\right)\right) < \sum_{j=1}^{n}\log\left(\left(\prod_{i=1}^{k^*}\left(a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{(\mathbf{x}_j-\boldsymbol{\mu}_i^*)^t \boldsymbol{\Sigma}_i^{*-1}(\mathbf{x}_j-\boldsymbol{\mu}_i^*)}{2}\right)\right)\right)^{-1}\right). \quad (5.2)$$

The right hand side of equation (5.2) can be written as

[3] Using $0<a_i^*<1,\ 0<f_i^*<1,\ 0<a_i^* f_i^* < a_i^*$ where $i=1,2,\ldots,k^*$ and $\sum_{i=1}^{k^*} a_i^* = 1$, we get $\sum_{i=1}^{k^*} a_i^* f_i^* < 1$

$$\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*}a_i^*\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i^*\mid}}\exp\left(-\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i^*\right)^t\boldsymbol{\Sigma}_i^{*-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i^*\right)}{2}\right)\right)$$
$$<-\sum_{j=1}^{n}\sum_{i=1}^{k^*}\left(\log\left(a_i^*\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i^*\mid}}\right)\right)+\sum_{j=1}^{n}\sum_{i=1}^{k^*}\left(\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i^*\right)^t\boldsymbol{\Sigma}_i^{*-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i^*\right)}{2}\right). \tag{5.3}$$

We apply $\log\left[\sum_{i=1}^{h}\phi_i\right]>\log\left[\prod_{i=1}^{h}\phi_i\right]$ where $0<\phi_i<1, i=1,2,...,h$ (see Section 5.2.1) to the second term of equation (5.1) yielding

$$\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k}a_i\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i\mid}}\exp\left(-\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)^t\boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)}{2}\right)\right)$$
$$>\sum_{j=1}^{n}\log\left(\prod_{i=1}^{k}a_i\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i\mid}}\exp\left(-\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)^t\boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)}{2}\right)\right). \tag{5.4}$$

The right-hand-side of equation (5.4) can be written as

$$\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k}a_i\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i\mid}}\exp\left(-\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)^t\boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)}{2}\right)\right)$$
$$>\sum_{j=1}^{n}\sum_{i=1}^{k}\left(\log\left(a_i\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i\mid}}\right)\right)-\sum_{j=1}^{n}\sum_{i=1}^{k}\left(\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)^t\boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)}{2}\right). \tag{5.5}$$

Applying minus sign to both sides of equation (5.5), we get

$$-\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k}a_i\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i\mid}}\exp\left(-\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)^t\boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)}{2}\right)\right)$$
$$<-\sum_{j=1}^{n}\sum_{i=1}^{k}\left(\log\left(a_i\frac{1}{\sqrt{(2\pi)^p\mid\boldsymbol{\Sigma}_i\mid}}\right)\right)+\sum_{j=1}^{n}\sum_{i=1}^{k}\left(\frac{\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)^t\boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_i\right)}{2}\right). \tag{5.6}$$

Note that $\sum_{j=1}^{n}\sum_{L=1}^{k^*}\left(\mathbf{x}_j-\boldsymbol{\mu}_L^*\right)^t\boldsymbol{\Sigma}_L^{*-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_L^*\right)$ of equation (5.3) and $\sum_{j=1}^{n}\sum_{L=1}^{k}\left(\mathbf{x}_j-\boldsymbol{\mu}_L\right)^t\boldsymbol{\Sigma}_L^{-1}\left(\mathbf{x}_j-\boldsymbol{\mu}_L\right)$ of equation (5.6) are greater than $\sum_{L=1}^{k^*}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L^*\right)^t\boldsymbol{\Sigma}_L^{*-1}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L^*\right)$ and $\sum_{L=1}^{k}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)^t\boldsymbol{\Sigma}_L^{-1}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)$, respectively. $\sum_{L=1}^{k^*}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L^*\right)^t\boldsymbol{\Sigma}_L^{*-1}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L^*\right)$ and $\sum_{L=1}^{k}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)^t\boldsymbol{\Sigma}_L^{-1}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)$ follow χ^2 distribution with *k*p* and *kp* degrees of freedom, respectively. They are derived from the following property (Mardia et al., 1979):

$$\sum_{i=1}^{n}\left(\mathbf{x}_i-\boldsymbol{\mu}_L\right)^t\boldsymbol{\Sigma}_L^{-1}\left(\mathbf{x}_i-\boldsymbol{\mu}_L\right)=ntr\left(\boldsymbol{\Sigma}_L^{-1}\mathbf{S}_L\right)+n\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)^t\boldsymbol{\Sigma}_L^{-1}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)$$

where $n\mathbf{S}_L=\sum_{i=1}^{n}\left(\mathbf{x}_i-\bar{\mathbf{x}}_L\right)\left(\mathbf{x}_i-\bar{\mathbf{x}}_L\right)^t$ and $\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)^t\boldsymbol{\Sigma}_L^{-1}\left(\bar{\mathbf{x}}_L-\boldsymbol{\mu}_L\right)\sim\chi_p^2$. The lower limit (it is found by Prop1 swapping places with Prop2 that is Prop2 and Prop1 are applied to first and second term of equation (5.1), respectively) has the same distribution as the upper limit hence $-2\log\lambda=2\sum_{j=1}^{n}\log\left(\frac{\sum_{i=1}^{k^*}a_i^*\phi\left(\mathbf{x}_j\mid\boldsymbol{\mu}_i^*,\boldsymbol{\Sigma}_i^*\right)}{\sum_{i=1}^{k}a_i\phi\left(\mathbf{x}_j\mid\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right)}\right)$ satisfy χ^2 distribution with *(k*+k)p* degrees of freedom. H_0 is accepted when $-2\log\lambda=2\sum_{j=1}^{n}\log\left(\frac{\sum_{i=1}^{k^*}a_i^*\phi\left(\mathbf{x}_j\mid\boldsymbol{\mu}_i^*,\boldsymbol{\Sigma}_i^*\right)}{\sum_{i=1}^{k}a_i\phi\left(\mathbf{x}_j\mid\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right)}\right)$ is less than $\chi^2_{\alpha,(k+k^*)p}$.

If $-2\log\lambda = 2\sum_{j=1}^{n}\log\left(\frac{\sum_{i=1}^{k^*} a_i^* \phi\left(\mathbf{x}_j \mid \boldsymbol{\mu}_i^*, \boldsymbol{\Sigma}_i^*\right)}{\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}\right) < 0$ due to the following prop erties

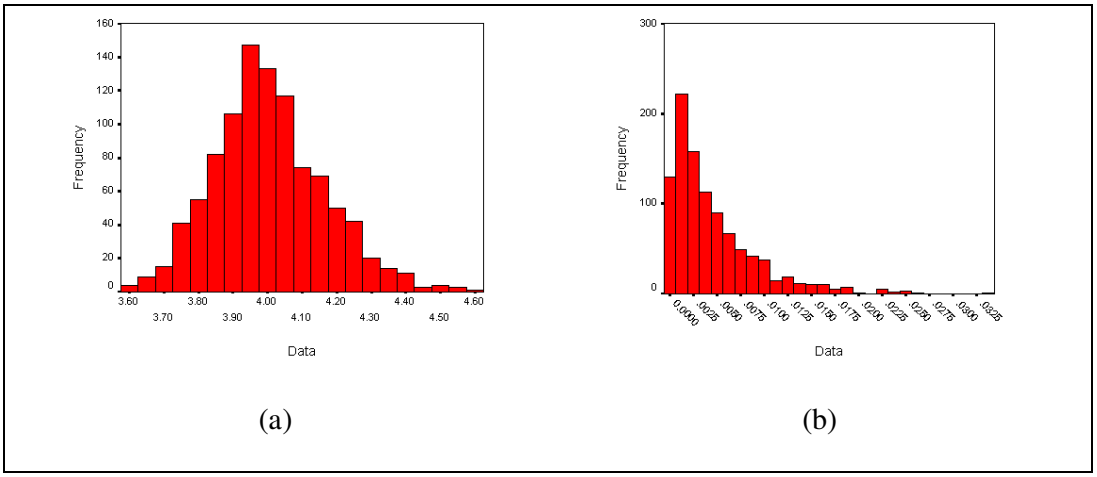
$\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\right) < 0$ and $\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \phi\left(\mathbf{x}_j \mid \boldsymbol{\mu}_i^*, \boldsymbol{\Sigma}_i^*\right)\right) < 0$, we swap places between $\boldsymbol{\theta} = \left(\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_1 \\ \vdots \\ \boldsymbol{\Sigma}_k \end{bmatrix}\right)$ and $\boldsymbol{\theta}^* = \left(\begin{bmatrix} a_1^* \\ \vdots \\ a_{k^*}^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\mu}_1^* \\ \vdots \\ \boldsymbol{\mu}_{k^*}^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_1^* \\ \vdots \\ \boldsymbol{\Sigma}_{k^*}^* \end{bmatrix}\right)$ and test $H_0 : \boldsymbol{\theta}^* = \left(\begin{bmatrix} a_1^* \\ \vdots \\ a_{k^*}^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\mu}_1^* \\ \vdots \\ \boldsymbol{\mu}_{k^*}^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_1^* \\ \vdots \\ \boldsymbol{\Sigma}_{k^*}^* \end{bmatrix}\right)$ versus $H_1 : \boldsymbol{\theta} = \left(\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_1 \\ \vdots \\ \boldsymbol{\Sigma}_k \end{bmatrix}\right)$ where H_0 is accepted when $-2\log\lambda$ (which is now greater than 0) is less than $\chi^2_{\alpha,(k+k^*)p}$.

5.2.3 The performance of the hypothesis testing

The characteristics of the hypothesis testing as mentioned in the previous section are described by performing the following processes for $\sum_{i=1}^{2} a_i \phi\left(x, \mu_i, \sigma_i^2\right)$ where ($a_1 = 0.4, \mu_1 = 0.0, \sigma_1^2 = 1.0$) and ($a_2 = 0.6, \mu_2 = 2.0, \sigma_2^2 = 0.25$) (refer to Table 3.1 under Sample 1) and repeat them 1000 times:

We generate 1000 simulation data using Box and Muller Transformation (Box and Muller, 1958) and calculate $\sum_{L=1}^{2}\left(\frac{\bar{x} - \mu_L}{\sigma_L}\right)^2$ and $\sum_{L=1}^{2}\left(\frac{\bar{x}_L - \mu_L}{\sigma_L}\right)^2$. The one thousand (1000) simulation data is then plotted as shown in Figure 5.3. Figure 5.4 shows several

χ^2 distributions for comparison purposes. Note that the histogram displayed in Figure 5.3 (b) is similar in terms of shape to Figure 5.4 (a).



(a) (b)

Figure 5.3 Simulation data is displayed in the histogram for (a) $\sum_{L=1}^{2}\left(\frac{\bar{x}-\mu_L}{\sigma_L}\right)^2$ and (b) $\sum_{L=1}^{2}\left(\frac{\bar{x}_L-\mu_L}{\sigma_L}\right)^2$.

The performance of the hypothesis testing is described by using Box and Muller Transformation (Box and Muller, 1958) to generate simulation data for $\sum_{i=1}^{K} a_i \phi(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \sum_{i=1}^{K} a_i \frac{1}{\sqrt{(2\pi)^d \mid \boldsymbol{\Sigma}_i \mid}} \exp\left(\frac{-(\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}{2}\right)$ focusing on two components with the following properties: a_1 is chosen from 0.1,…,0.9, a_2 is derived from the following formula $a_2 = 1 - a_1$, μ_1 is fixed at 0.0; μ_2 is chosen from 0.25,0.50,0.75,…,3.0, σ_{11} and σ_{22} are chosen from $\frac{1}{2}, \frac{1}{3}, \cdots, \frac{1}{10}, 1, 2, 3, \cdots, 10$. A total of 25 samples, each with 1000 observations, are generated for each model.

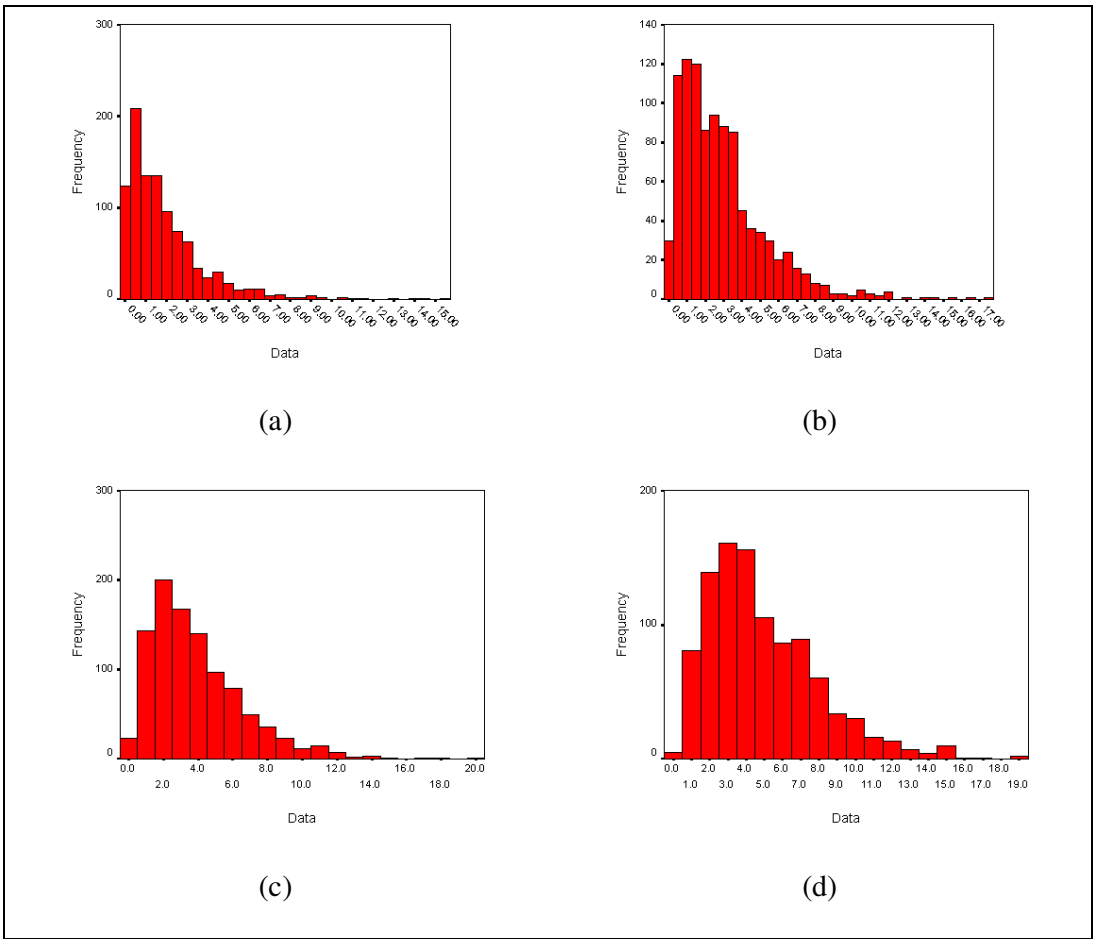


Figure 5.4 χ^2 -distribution with (a) 2, (b) 3, (c) 4 and (d) 5 degrees of freedom.

Note that

$$Percentage = \left(\frac{\text{Number of hypothesis testings that accepts 2 number of components}}{\text{Total number of hypothesis testings}} \right)\%$$

is calculated and *Range* is assigned to each model where *Range* equals to 1 represents percentage of overlapping between 0% and 25%, 2 represents percentage of overlapping between 25% and 50%, 3 represents percentage of overlapping between 50% and 75% and 4 represents percentage of overlapping between 75% and 100%.

Example of an output is given in Figure 5.5, which represents ($a_1 = 0.2, \mu_1 = 0.0, \sigma_1^2 = (0.707)^2$) and ($a_2 = 0.8, \mu_2 = 0.25, \sigma_2^2 = 1.0$).The Range for the given example equals to 3 and H_0 is accepted when $k^* = 3$ where p is greater

than 0.05. Note that $-2\log\lambda$ is given in the first bracket and probability value p is given in the second.

```
Akaike Information Criteria

no_of_components      AIC     Min
1       2759.241221463931     2759.241221463931
2       2742.695779311172     2742.695779311172
3       2745.7438644121394    2742.695779311172

Hypothesis Testing

H_0:theta,k=1
H_0:theta,k=1 versus H_1:theta*,k*=2  (22.54544215275928)(p=5.028078782631841E-5)
H_0:theta,k=1 versus H_1:theta*,k*=3  (25.497357051791823)(p=3.9955836008321434E-5)

H_0:theta,k=2
H_0:theta,k=2 versus H_1:theta*,k*=3  (2.951914899032545)(p=0.7073991861415829)
```

Figure 5.5 Results of AIC and hypothesis testing for $\sum_{i=1}^{2} a_i \phi\left(x, \mu_i, \sigma_i^2\right)$ where

($a_1 = 0.2$, $\mu_1 = 0.0$, $\sigma_1^2 = (0.707)^2$) and ($a_2 = 0.8$, $\mu_2 = 0.25$, $\sigma_2^2 = 1.0$).

The results are displayed in Figure 5.6 where the values used can be found in Table 5.1. The value under the column titled "(Freq/Tot)%" of Table 5.1 that corresponds to Percentage equals to 100 decreases not lower than 50 as the Range increases.

Table 5.1 Frequency table for Range equals to (a) 1, (b) 2, (c) 3 and (d) 4.

		Frequency	(Freq/Tot)%
Valid	0	21	1.2
	25	1	0.1
	33.3	24	1.4
	50	197	11.6
	66.7	131	7.7
	75	15	0.9
	100	1313	77.1
	Total	1702	100

(a)

		Frequency	(Freq/Tot)%
Valid	0	29	4.9
	33.3	12	2
	50	121	20.3
	66.7	37	6.2
	75	7	1.2
	100	391	65.5
	Total	597	100

(b)

Table 5.1 Continued.

		Frequency	(Freq/Tot)%
Valid	0	13	9.5
	50	36	26.3
	66.7	10	7.3
	75	1	0.7
	100	77	56.2
	Total	137	100

(c)

		Frequency	(Freq/Tot)%
Valid	0	6	20.7
	50	6	20.7
	66.7	2	6.9
	100	15	51.7
	Total	29	100

(d)

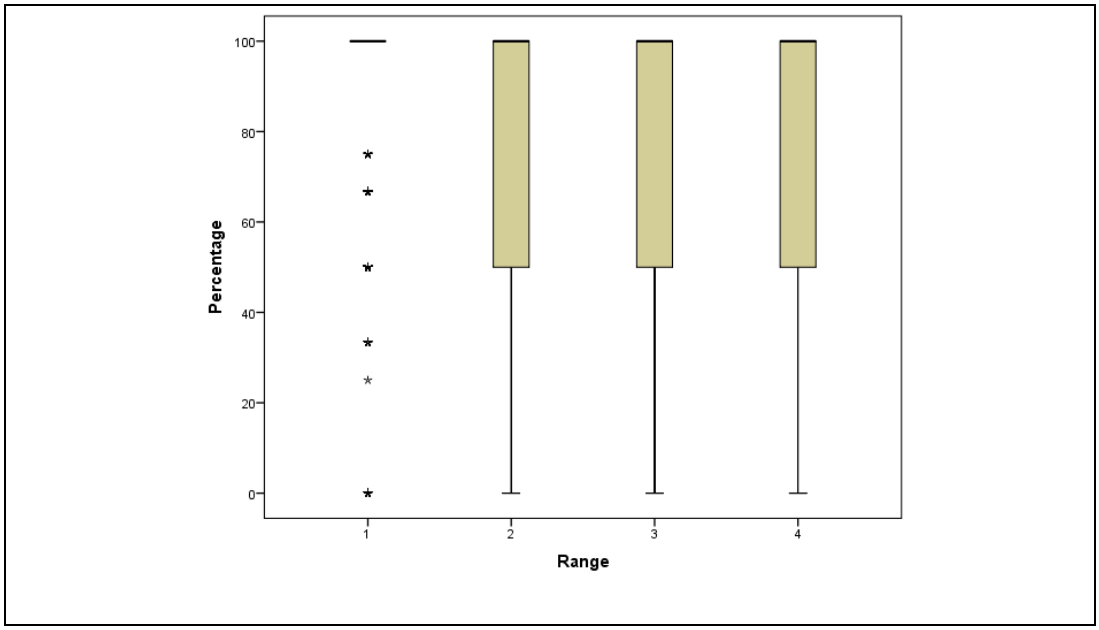


Figure 5.6 Percentage is plotted against Range in the Box plot.

5.3 Comparison between using the AIC and hypothesis testing in determining the number of components in GMM

Akaike Information Criteria (AIC) used in the improvement of EM algorithm for GMM is defined by $AIC = 2pmtr - 2Log(L)$ where $pmtr$ is the number of parameters and $Log(L)$ is the maximized log-likelihood function.

Let

$$AIC_k = 2\Omega k - 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)^t \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)}{2}\right)\right) \tag{5.7}$$

and

$$AIC_{k^*} = 2\Omega k^* - 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)^t \boldsymbol{\Sigma}_i^{*-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)}{2}\right)\right) \tag{5.8}$$

Where the second term of AIC_k and AIC_{k^*} is taken from equation (5.1), and $2\Omega = 6$ if $p = 1$.

In this section, we present two cases. They are:

Case 1: Let $AIC_{k^*} > AIC_k$ (according to Step 5, AIC_k is minimum therefore it is chosen) where $k^*>k$. Using equations (5.7) and (5.8), we get

$$\begin{aligned} &2\Omega k^* - 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)^t \boldsymbol{\Sigma}_i^{*-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)}{2}\right)\right) > \\ &2\Omega k - 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)^t \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)}{2}\right)\right) \end{aligned} \tag{5.9}$$

Equation (5.9) can be written as

$$2\Omega\left(k^* - k\right) > \beta \tag{5.10}$$

where

$$\beta = 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)^t \boldsymbol{\Sigma}_i^{*-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)}{2}\right)\right)$$
$$-2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)^t \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)}{2}\right)\right)$$

Let $\gamma \sim \chi^2_{\alpha,(k+k^*)p}$. If $2\Omega(k^*-k) > \beta > \gamma$, we reject H_0, which contradicts the AIC results. If $2\Omega(k^*-k) > \beta$ and $\gamma > \beta$, we accept H_0.

Case 2: Let $AIC_{k^*} < AIC_k$ (according to Step 5, AIC_{k^*} is minimum therefore it is chosen) where $k^*>k$. By repeating the process in Case 1, that is using equations (5.7) and (5.8), we get

$$2\Omega(k^*-k) < \beta \tag{5.11}$$

where

$$\beta = 2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k^*} a_i^* \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i^* \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)^t \boldsymbol{\Sigma}_i^{*-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i^*\right)}{2}\right)\right)$$
$$-2\sum_{j=1}^{n}\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\exp\left(-\frac{\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)^t \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)}{2}\right)\right)$$

Let $\gamma \sim \chi^2_{\alpha,(k+k^*)p}$. If $2\Omega(k^*-k) < \beta$ and $\beta > \gamma$, we reject H_0. If $2\Omega(k^*-k) < \beta < \gamma$, we accept H_0, which contradicts the AIC results.

Figure 5.7 (the results shown here are taken from Chapter 4) shows hypothesis testing results support those of AIC (note that $-2\log\lambda$ is given in the first bracket whereas probability value is given in the second). Case 1 and 2 can be found in the hypothesis testing when $(k=2, k^*=3)$ and $(k=1, k^*=2,3)$, respectively. The probability value α is fixed at 0.05. Figure 5.8 shows, especially for the case when k and k^* equal to 2 and 3, respectively, hypothesis testing results do not support those of AIC. Case 2 can be found in all of the hypothesis testing.

```
Akaike Information Criteria

no_of_components      AIC     Min
1       92.957704551177       92.957704551177
2       78.93876969062063     78.93876969062063
3       80.78840789298813     78.93876969062063


Hypothesis Testing

H_0:theta,k=1
H_0:theta,k=1 versus H_1:theta*,k*=2  (20.01893486055637)(Prob=1.6832767836781848E-4)
H_0:theta,k=1 versus H_1:theta*,k*=3  (24.16929665818887)(Prob=7.386963782137005E-5)

H_0:theta,k=2
H_0:theta,k=2 versus H_1:theta*,k*=3  (4.150361797632499)(Prob=0.5279773066298088)
```

Figure 5.7 Results of AIC and hypothesis testing for customer C.

```
Akaike Information Criteria

no_of_components      AIC     Min
1       47.53326292383591     47.53326292383591
2       35.70610331847866     35.70610331847866
3       33.64022457875683     33.64022457875683


Hypothesis Testing

H_0:theta,k=1
H_0:theta,k=1 versus H_1:theta*,k*=2  (17.827159605357252)(Prob=4.7764602611496796E-4)
H_0:theta,k=1 versus H_1:theta*,k*=3  (25.89303834507908)(Prob=3.325553903926139E-5)

H_0:theta,k=2
H_0:theta,k=2 versus H_1:theta*,k*=3  (8.065878739721828)(Prob=0.15264184961471652)
```

Figure 5.8 Results of AIC and hypothesis testing for customer D.

In addition to the above, the contradiction between AIC and hypothesis testing can also be seen in Figure 5.1 and Table 5.1 especially the values under the column

titled "(Freq/Tot)%" that correspond to Percentage not equal to 100. The total number of Percentage not equal to 100 increases as the Range increases.

Hypothesis testing results depend on log-likelihood function and the choice of the probability value α that gives $\chi^2_{\alpha,(k+k^*)p}$. AIC results on the other hand depend on log-likelihood function only as shown in Figures 5.9 and 5.10. Hypothesis testing results are similar to those of AIC if α is set at different value (i.e. other than 0.05).

Further research on the behavior of the hypothesis testing especially when it conflicts with AIC is required that will involve the use of the power of a test (Guenther, 1977).

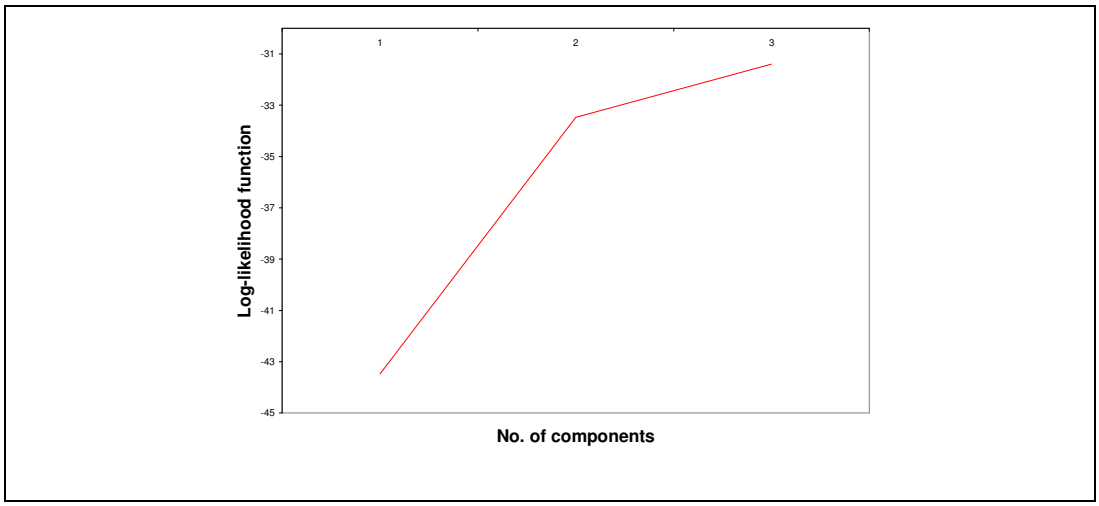


Figure 5.9 Log-likelihood function against number of components for Customer C.

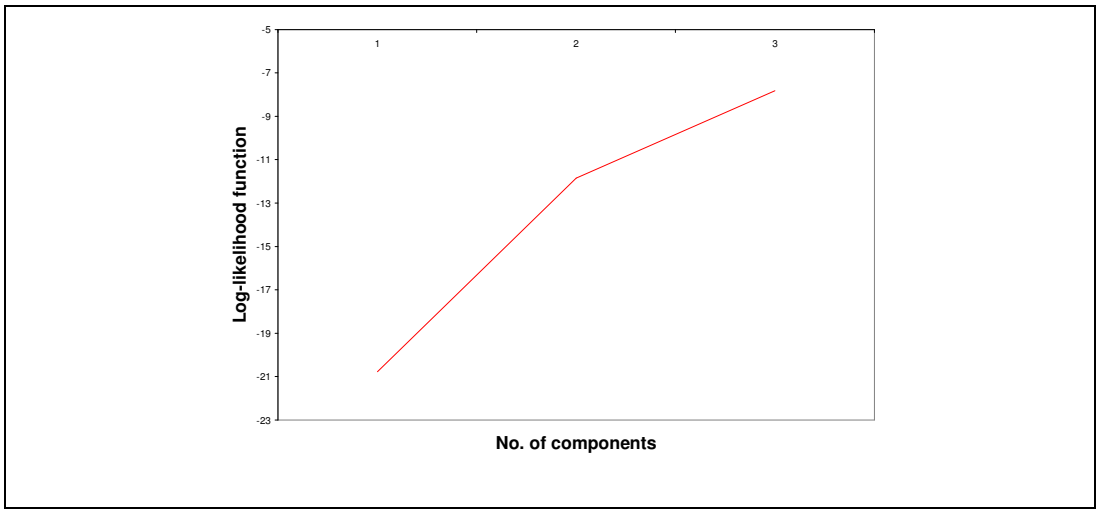


Figure 5.10 Log-likelihood function against number of components for Customer D.

5.4 Summary

In the previous chapters, we showed the effects of fraud activities to telecommunication industry and gave a brief introduction to GMM and EM algorithm. We also mentioned when would we determine the number of components in GMM and gave several examples that are normally used in the determination of the number of components in GMM, including the use of AIC in the determination process.

We successfully derived hypothesis testing in the previous sections, which we believe can be used as an alternative method to AIC in the determination of the number of components in GMM.

CHAPTER 6

"REAL TIME" FRAUD DETECTION ALGORITHM FOR TELECOMMUNICATION INDUSTRY USING GAUSSIAN MIXED MODEL

6.1 Introduction

Sain et al. (1999) consider the difficult task of using seismic signals (or any other discriminates) for detecting nuclear explosions from the large number of background signals such as earth quakes and mining blasts. They used the following nonparametric bootstrapping by Efron and Tibshirani (1993) to test $H_0 : \mathbf{x}_{n+1} \in \Pi$ versus $H_1 : \mathbf{x}_{n+1} \notin \Pi$ for the case in which no events in the training sample are labeled and the number of event types represented in the training sample is unknown.

Step A: Given the training sample $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in \Pi$ and potential outlier $\mathbf{x}_{n+1}$, calculate W based on $W = \dfrac{\sup_{\boldsymbol{\theta} \in \Theta} L_0(\boldsymbol{\theta})}{\sup_{\boldsymbol{\theta} \in \Theta} L_1(\boldsymbol{\theta})}$ where $L_0(\boldsymbol{\theta}) = \left(\prod_{s=1}^{n} f(\mathbf{x}_s; \boldsymbol{\theta}) \right) f(\mathbf{x}_{n+1}; \boldsymbol{\theta})$ and $L_1(\boldsymbol{\theta}) = \left(\prod_{s=1}^{n} f(\mathbf{x}_s; \boldsymbol{\theta}) \right)$.

Step B: For each integer *b*, *b*=1,…,*B* draw a sample of size *n* with replacement from the training data. Additionally, an (*n*+1)st observation is also drawn from the training data (because we are approximating the distribution of *W* when $H_0 : \mathbf{x}_{n+1} \in \Pi$ is true). For

each b, re-sampled data is used to compute the statistics in $W = \frac{\sup_{\boldsymbol{\theta} \in \Theta} L_0(\boldsymbol{\theta})}{\sup_{\boldsymbol{\theta} \in \Theta} L_1(\boldsymbol{\theta})}$. The test statistics is denoted by W_b^*.

Step C: Define W_a to be the (100α)th percentile of all W_b^*. Specifically if α =j/(B+1), then W_a is the jth smallest value of $\{W_b^*\}_{b=1}^{B}$ (see McLachlan, 1987)

Step D: $H_0 : \mathbf{x}_{n+1} \in \Pi$ is rejected and concluded that the (n+1)st point is an outlier if $W \leq W_a$.

This chapter proposes a new algorithm that can be efficiently used to identify fraud activities (Section 6.2). The algorithm is developed by using the above concept but instead of using nonparametric bootstrapping, we use likelihood ratio test. It also finds the characteristics of historical fraud and non-fraud calls and is consequently used in identifying possible fraud call instantly for immediate call verification process. Using simulation and data obtained from one of the leading telecommunication company in Malaysia, we show that the proposed algorithm has successfully detected outgoing fraud calls as suspected by the company (Sections 6.3 and 6.4).

6.2 "Real time" fraud detection algorithm using GMM

The algorithm for detecting fraud activities as mentioned in Chapter 4, which include the improved EM algorithm for GMM, involves two steps. The first step finds and saves the final estimates of parameters in the database for each of the seven days. The second step finds the maximum of log-likelihood function, the similarity coefficient and performs the updating of the database process for the eight day

onwards. The observations used in the said steps that represent i^{th} day where i=1,2,3,… are collected over a period of 24 hours. The second step is improved to include the testing of each observation (as soon as it is available or in "real time") whether it is an outlier or not that is $H_0 : \mathbf{x}_{n+1} \in \Pi$ versus $H_1 : \mathbf{x}_{n+1} \notin \Pi$ (refer to Figure 6.1). Note that the rest of the second step remains unchanged.

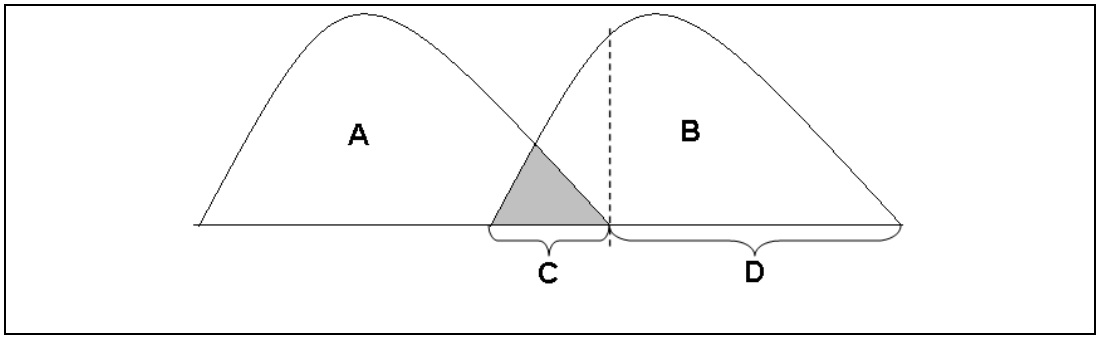


Figure 6.1 A represents the probability density function of customer X call detail record for the 1st day collected over a period of 24 hours and saved in the database. B represents the probability density function of customer X call detail record for the i^{th} day where i=8,9,… collected over a period of 24 hours. C represents customer X call detail record for the i^{th} day that is classified as an observation belonging to A and D represents customer X call detail record for the i^{th} day that is classified as an outlier (i.e. by using the parameters belonging to A).

The likelihood ratio statistics for testing the above hypothesis is defined by

$$-2\log\lambda = 2\sum_{j=1}^{n}\log\left(\frac{\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}{\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}\right) - 2\log\left(\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_{n+1} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\right)$$

or

$$-2\log\lambda = -2\log\left(\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_{n+1} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\right)$$

(Mardia et al., 1979). It can be written as

$$-2\log\lambda = -2\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}} \exp\left(-\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)' \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)}{2}\right)\right) \quad (6.1)$$

Using equations (5.3) and (5.6), we get

$$\begin{aligned} &\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}} \exp\left(-\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)' \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)}{2}\right)\right) \\ &< -\sum_{i=1}^{k}\left(\log\left(a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\right)\right) + \sum_{i=1}^{k}\left(\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)' \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)}{2}\right) \end{aligned} \quad (6.2)$$

and

$$\begin{aligned} &\log\left(\sum_{i=1}^{k} a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}} \exp\left(-\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)' \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)}{2}\right)\right) \\ &> \sum_{i=1}^{k}\left(\log\left(a_i \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_i \mid}}\right)\right) - \sum_{i=1}^{k}\left(\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)' \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_i\right)}{2}\right) \end{aligned} \quad (6.3)$$

For example, replacing *k* with unity in equations (6.2) and (6.3), we get

$$\begin{aligned} &-\log\left(a_1 \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_1 \mid}} \exp\left(-\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)' \boldsymbol{\Sigma}_1^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)}{2}\right)\right) \\ &< -\left(\left(\log\left(a_1 \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_1 \mid}}\right)\right) - \left(\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)' \boldsymbol{\Sigma}_1^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)}{2}\right)\right) \end{aligned}$$

and

$$-\log\left(a_1 \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_1 \mid}} \exp\left(-\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)^t \boldsymbol{\Sigma}_1^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)}{2}\right)\right)$$
$$> \left(\log\left(a_1 \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_1 \mid}}\right)\right) - \left(\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)^t \boldsymbol{\Sigma}_1^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)}{2}\right).$$

It can be shown that

$$-\log\left(a_1 \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_1 \mid}} \exp\left(-\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)^t \boldsymbol{\Sigma}_1^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)}{2}\right)\right)$$
$$= -\left(\log\left(a_1 \frac{1}{\sqrt{(2\pi)^p \mid \boldsymbol{\Sigma}_1 \mid}}\right)\right) + \left(\frac{\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)^t \boldsymbol{\Sigma}_1^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_1\right)}{2}\right).$$

Note that $\sum_{L=1}^{k}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_L\right)^t \boldsymbol{\Sigma}_L^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_L\right)$ of equations (6.2) and (6.3) is greater than $\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_L\right)^t \boldsymbol{\Sigma}_L^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_L\right)$ where $\mathbf{x}_{n+1} \in N\left(\boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L\right)$ and $L = 1,2,...,k$. $\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_L\right)^t \boldsymbol{\Sigma}_L^{-1}\left(\mathbf{x}_{n+1}-\boldsymbol{\mu}_L\right)$ follows χ^2 distribution with one (1) degree of freedom (Mardia et al., 1979). Hence, $-2\log\lambda = -2\log\left(\sum_{i=1}^{k} a_i \phi\left(\mathbf{x}_{n+1} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\right)$ satisfy χ^2 distribution with one (1) degree of freedom.

The characteristics of the hypothesis testing as mentioned above are described by performing the following processes for $\sum_{i=1}^{2} a_i \phi\left(x, \mu_i, \sigma_i^2\right)$ where ($a_1 = 0.4, \mu_1 = 0.0, \sigma_1^2 = 1.0$) and ($a_2 = 0.6, \mu_2 = 2.0, \sigma_2^2 = 0.25$) (refer to Table 3.1 under Sample 1). We generate 1000 simulation data using Box and Muller Transformation (Box and Muller, 1958) and calculate $\sum_{L=1}^{2}\left(\frac{x_i-\mu_L}{\sigma_L}\right)^2$ and $\left(\frac{x_i-\mu_L}{\sigma_L}\right)^2$

where $x_i \in N(\mu_L, \sigma_L^2)$ and $L = 1,2$. The one thousand (1000) simulation data is then plotted as shown in Figure 6.2.

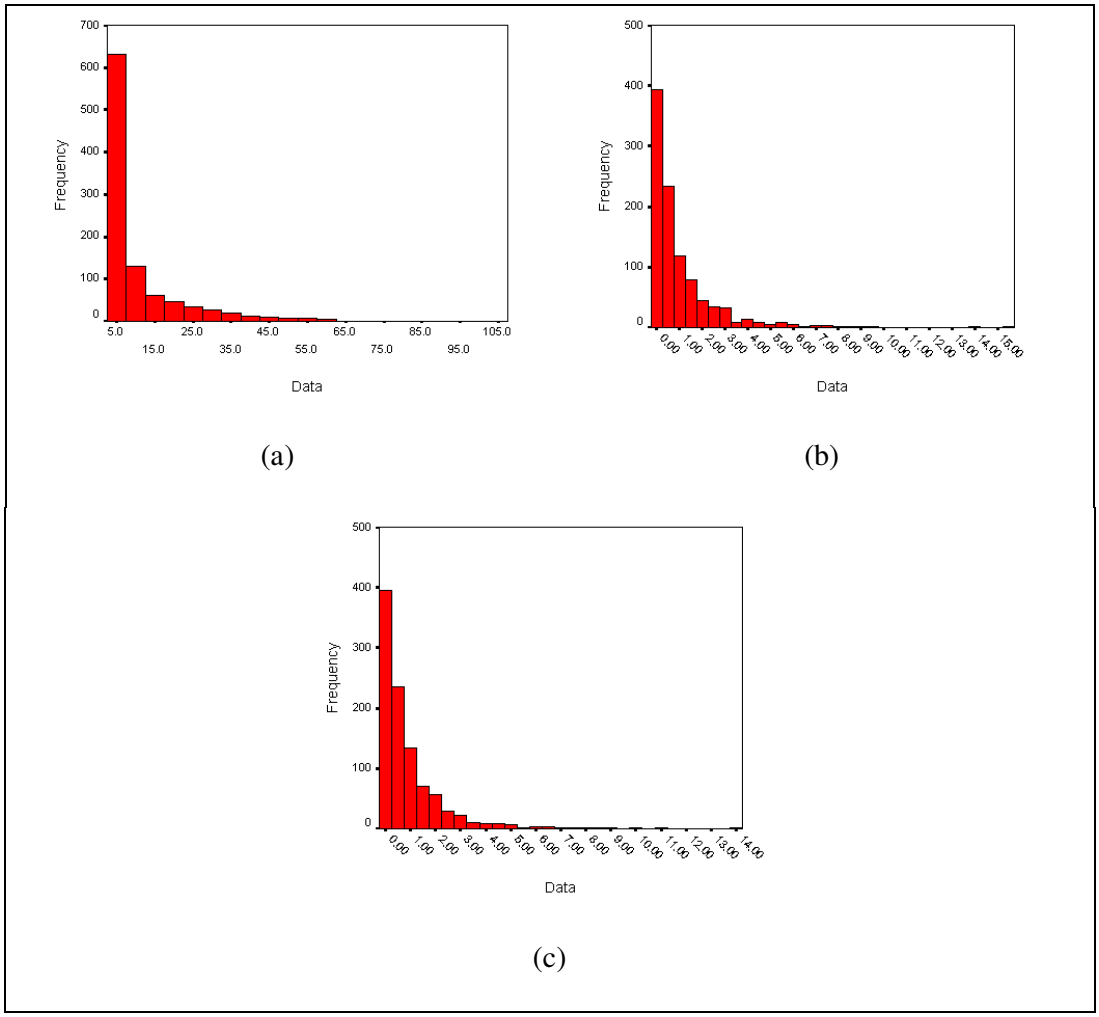


Figure 6.2 Simulation data is displayed in the histogram for (a) $\sum_{L=1}^{2}\left(\frac{x_i - \mu_L}{\sigma_L}\right)^2$, (b) $\left(\frac{x_i - \mu_L}{\sigma_L}\right)^2$ where $x_i \in N(\mu_L, \sigma_L^2)$ and $L = 1,2$ and (c) χ^2 distribution with one (1) degree of freedom.

Note that the histogram displayed in Figure 6.2 (b) is similar in terms of shape to (c) χ^2 distribution with one (1) degree of freedom.

H_0 is accepted when $-2\log\lambda = -2\log\left(\sum_{i=1}^{k} a_i \phi(\mathbf{x}_{n+1} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right)$ is less than $\chi^2_{\alpha,p}$

where α=0.0001. The value of α is chosen such that it follows Sachs (1984) where a new observation that falls outside the boundaries derived from the mean and (four times) standard deviation of the current observations is an outlier.

6.2.1 The performance of the "real time" fraud detection algorithm using GMM

We generate n_1=1000 simulation data from $N(0,1)$ using Box and Muller Transformation (Box and Muller, 1958). We treat the simulation data as training data (or td). We calculate the mean (denoted by μ_{td}), variance (denoted by σ^2_{td}) and log-likelihood function (denoted by llf_{td}). The following is repeated 100 times: For each $\mu_O = 0.0,0.1,...,10.0$, we generate n_2=1000 simulation data from $N(\mu_O,1)$ using Box and Muller Transformation (Box and Muller, 1958). We treat the simulation data as observation (or O). Using μ_{td} and σ^2_{td} from the above, we calculate the log-likelihood function (denoted by llf_O) and similarity coefficient $\frac{llf_O}{llf_{td}}$. Each observation of the simulation data is tested whether it is an outlier or not by using the likelihood ratio statistics as mentioned in Section 6.2. We count the total number of outliers for the given simulation data and convert it into percentage (denoted by the percentage of outliers or outliers(%)).

Figure 6.3 (a) shows (negative) s-curve where similarity coefficient decreases as μ_O increases and (b) shows, as we expected, (positive) s-curve where the percentage of outliers increases as μ_O increases.

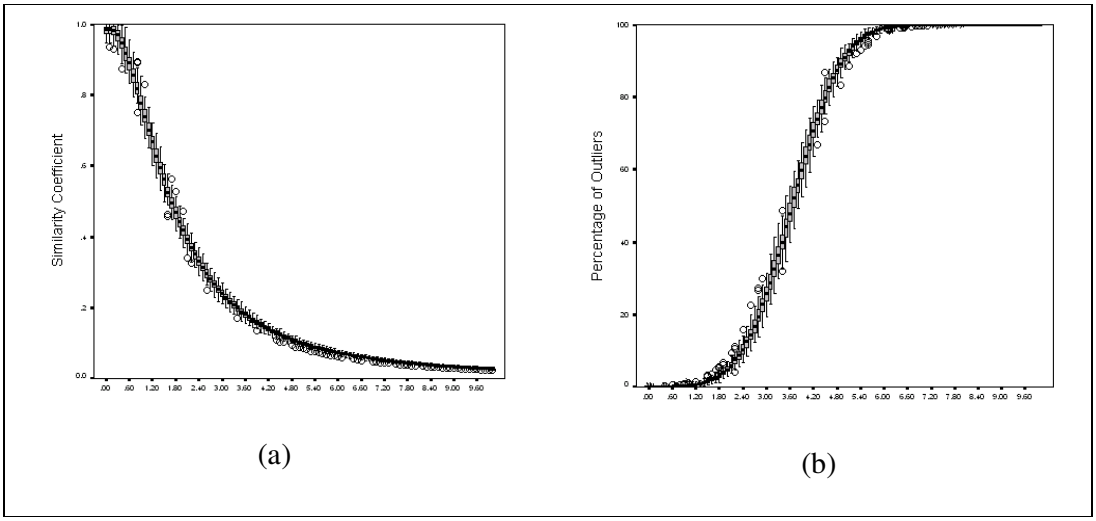


Figure 6.3 Box plot for (a) similarity coefficient and (b) percentage of outliers where the values on the x-axis represent μ_O.

For each of the five hundred customers (they are labeled as 1st, 2nd, 3rd,…,500th customer) and for each of the seven days, the number of components are chosen at random where the maximum number of components is fixed at three (3). If three components are chosen, the values of a_1, a_2 and a_3 are chosen at random from 0.1,…,0.9 where $a_1 + a_2 + a_3 = 1$; the values of μ_1, μ_2 and μ_3 are chosen at random from 0.25,0.50,0.75,1.0,2.0,3.0 where the means of two or more components are chosen such that no two or more means are the same; and the values of σ_1, σ_2 and σ_3 are chosen at random from $\frac{1}{2},\frac{1}{3},\cdots,\frac{1}{10},1,2,3,\cdots,10$. Similar steps are taken if less than three components are chosen. The chosen values (that represent a model) are used in $z_j = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \cos 2\pi u_{j+1}$ and $z_{j+1} = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \sin 2\pi u_{j+1}$ where $u_j, u_{j+1} \sim U(0,1)$ (Box and Muller, 1958) to generate (one thousand) simulation data. For each day starting the 8th till 28th, simulation data is generated using either one of the seven models (which represents the first condition where the performance of the

algorithm is evaluated by this condition) or other model derived from repeating the steps as mentioned above.

For each customer, Chapter 4's algorithm is employed on the models that represent 1st till 7th day. Each observation on 8th day onwards that satisfies the first condition is tested whether it is an outlier or not by using the likelihood ratio statistics as mentioned in Section 6.2. We repeat the steps as mentioned earlier that is we count the total number of outliers for the given data set (that represents 8th day onwards and satisfies first condition) and convert it into percentage. We find the frequency for each Outliers(%)and convert it into percentage (denoted by Freq(%)). Figure 6.4, which is derived from Table 6.1, shows the results from the steps taken on the 10th customer and Outliers(%) with the highest Freq(%) is zero. There is a huge gap (or difference) between Outliers(%) equals to 0.0 with the rest of the Outliers(%). The same pattern can also be found in Figure 6.5 (due to its size, only a few of the customers are displayed here).

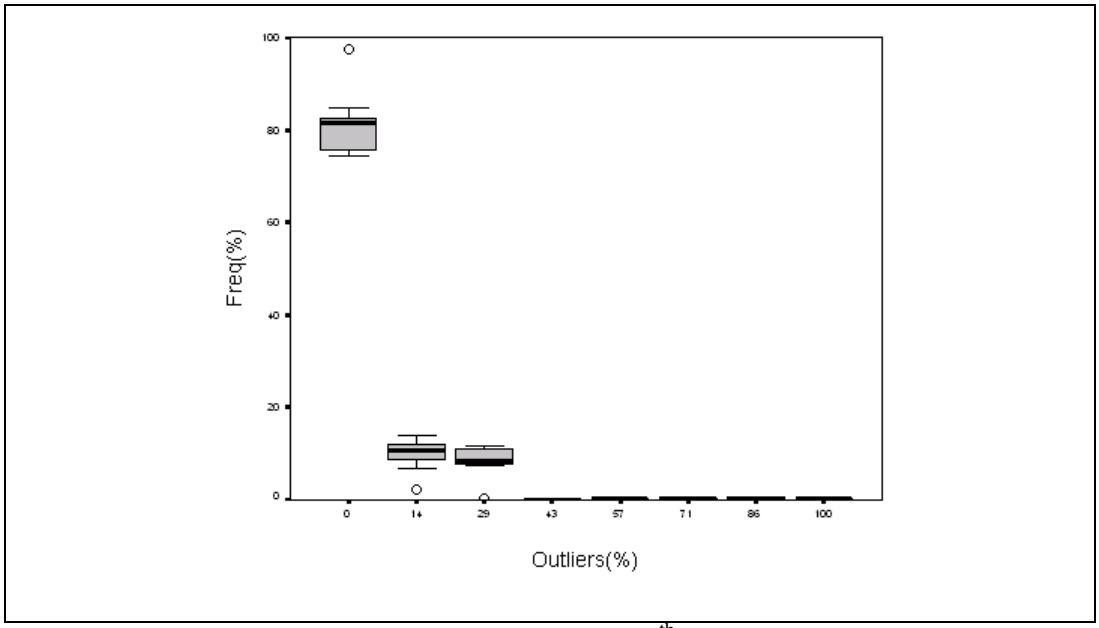
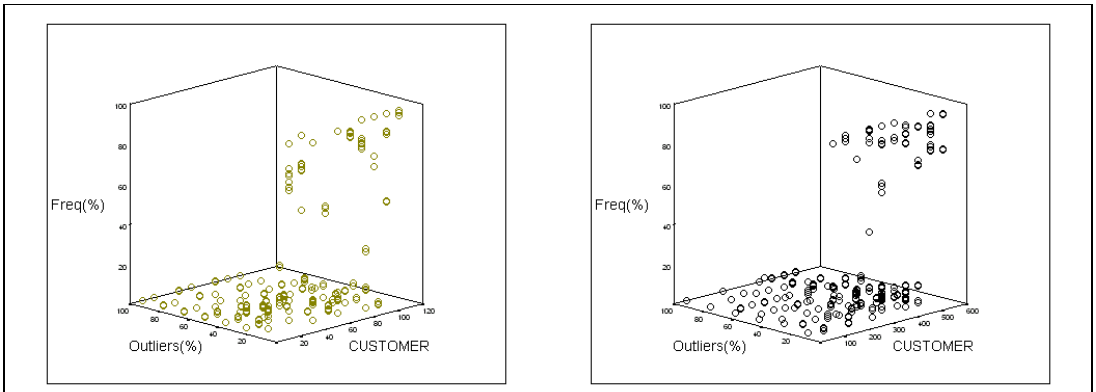


Figure 6.4 Box plot for the 10th customer.



(a) (b)

Figure 6.5 Scatter plot for (a) 10^{th} , 20^{th} ,…,100^{th} customer and (b) 50^{th} ,100^{th} ,150^{th} ,…,500^{th} customer.

6.3 Data

Two types of data will be used. They are:

(i) Simulation data. For each of the one thousand customers and for each of the seven days, the number of components are chosen at random where the maximum number of components is fixed at three (3). If three components are chosen, the values of a_1, a_2 and a_3 are chosen at random from 0.1,…,0.9 where $a_1 + a_2 + a_3 = 1$; the values of μ_1, μ_2 and μ_3 are chosen at random from 0.25,0.50,0.75,1.0,2.0,3.0 where the means of two or more components are chosen such that no two or more means are the same; and the values of σ_1, σ_2 and σ_3 are chosen at random from $\frac{1}{2},\frac{1}{3},\cdots,\frac{1}{10},1,2,3,\cdots,10$. Similar steps will be taken if less than three components are chosen. The chosen values (that represent a model) will be used in $z_j = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \cos 2\pi u_{j+1}$ and

Table 6.1 Outliers(%) * Freq(%) * 10th customer cross-tabulation.

Customer			Freq(%)														
			0.1	0.2	0.3	0.4	0.5	0.6	2.4	6.7	7.4	7.8	8.2	8.4	8.7	8.8	10	10.6
Cust_10.txt	Outliers(%)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		14.29	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1
		28.57	0	1	0	0	0	0	0	0	1	1	1	1	0	0	1	0
		42.86	3	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0
		57.14	0	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0
		71.43	1	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0
		85.71	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
		100	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	Total		6	5	6	3	1	1	1	1	1	1	1	1	1	1	1	1

11.1	11.4	11.6	11.9	12.4	14.1	74.2	75.7	75.8	78	81.4	81.5	82.4	84.8	97.4	Total
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	9
0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	9
1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	9
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	48

$z_{j+1} = \mu + (-2\sigma^2 \log u_j)^{\frac{1}{2}} \sin 2\pi u_{j+1}$ where $u_j, u_{j+1} \sim U(0,1)$ (Box and Muller, 1958) to generate (one thousand) simulation data. For each day starting the 8th till 28th, simulation data is generated using either one of the seven models or other model derived from repeating the steps as mentioned above.

(ii) Phone call data. Call detail record, which was supplied (or provided) by Telekom Malaysia Berhad (henceforth, TM), consists of calls made by customers and they were victims of fraud activities (altogether there are 18 customers and they are labeled as A till R to ensure confidentiality). As mentioned in Chapter 4, we performed several steps for each of TM's customer call detail record to get the desired format; e.g. group the real data according to service no, find the country that matches with the country code and sort the real data according to seize time. We divided them (i.e. real data for TM customers) into several parts and saved in the following format: (for each customer) 1, 2, 3, 4,… represent fn(1), fn(2), fn(3), fn(4),… and date(1), date(2), date(3),… where fn is a short-form for filename and date(1) < date(2) < date(3) < ….

6.4 Results

The results using simulation and real data are presented in the visual format as exemplified in Figure 6.6 where, for ease of visual presentation, the times when the calls were made are arranged in ascending order (and labeled 1,2,3,…). For each call, we calculate equation (6.1) and $\left(\frac{\text{The number of times } H_0 \text{ is rejected}}{\text{The total number of hypothesis testings}}\right) X\,(100\%)$ where the latter is denoted by Percentage. Percentage that is greater than zero is plotted in Figure 6.6.

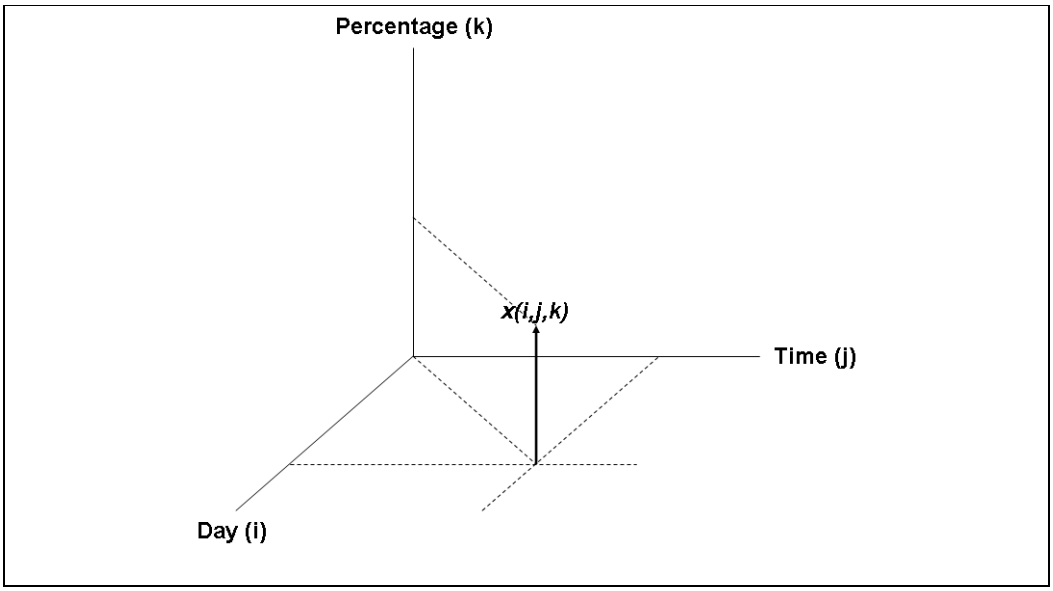


Figure 6.6 Percentage is plotted against Day and Time in the Scatter plot.

Due to its size, only a portion of the results are presented here. For the simulation data where we choose Customer 1, Figure 6.7 shows there are lots of calls that are classified as outliers.

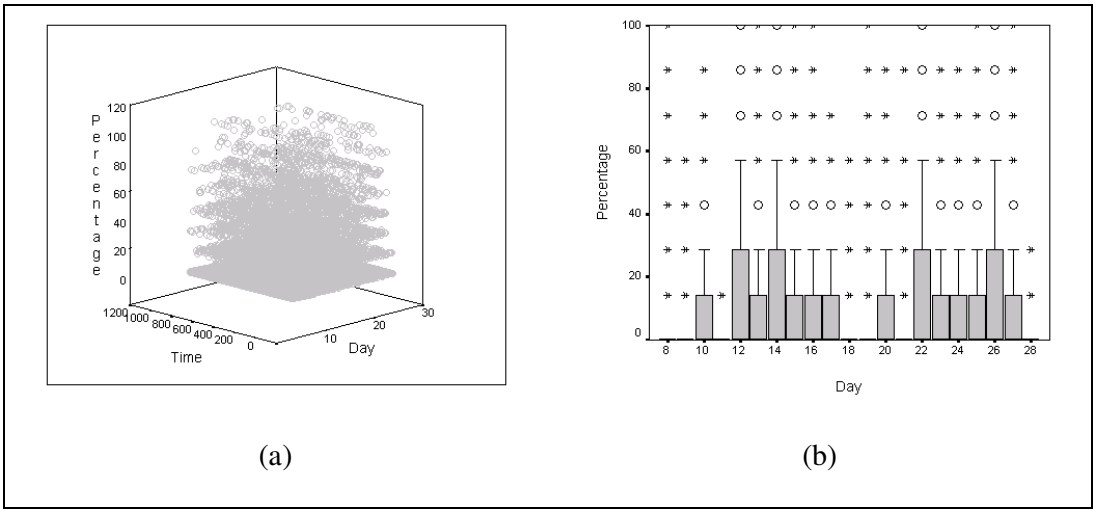


Figure 6.7 For Customer 1 of the simulation data, (a) Percentage is plotted against Day and Time in the Scatter plot and (b) Percentage is plotted against Day in the Box plot.

This is supported by Figure 6.8 and Table 6.2 for Day equals to 22. For the said Day, out of 1000 calls, each of 135 calls has the Percentage equals to 14.29, each of 105 calls has the Percentage equals to 42.86 and each of 72 calls has the Percentage greater than 70. Note that nine calls have the Percentage equals to 100. In other words, the call is considered as an outlier by all models saved inside the database.

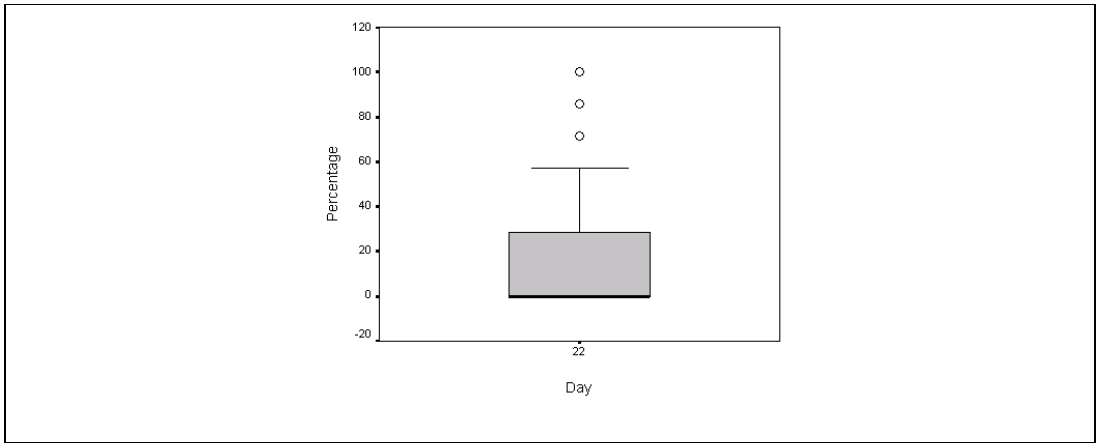


Figure 6.8 Percentage is plotted against Day, which is equals to 22, in the Box plot.

Table 6.2 Frequency table for Day equals to 22.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.00	538	53.8	53.8	53.8
	14.29	135	13.5	13.5	67.3
	28.57	95	9.5	9.5	76.8
	42.86	105	10.5	10.5	87.3
	57.14	55	5.5	5.5	92.8
	71.43	47	4.7	4.7	97.5
	85.71	16	1.6	1.6	99.1
	100.00	9	.9	.9	100.0
	Total	1000	100.0	100.0	

By taking similar steps, we get the following figures and table for Customer A of the real telecommunication data. Figure 6.9 shows there are quite a considerable number of calls that are classified as outliers and none of them has the Percentage equals to 100. Figure 6.10 and Table 6.3 are derived from Figure 6.9 where we are

focusing on Day equals to 72 (i.e. 15th of November 2011), the same day TM's system claimed to detect fraud activity. For the said Day, out of 4 calls, one call has the Percentage equals to 3, one call has the Percentage equals to 8 and one call has the

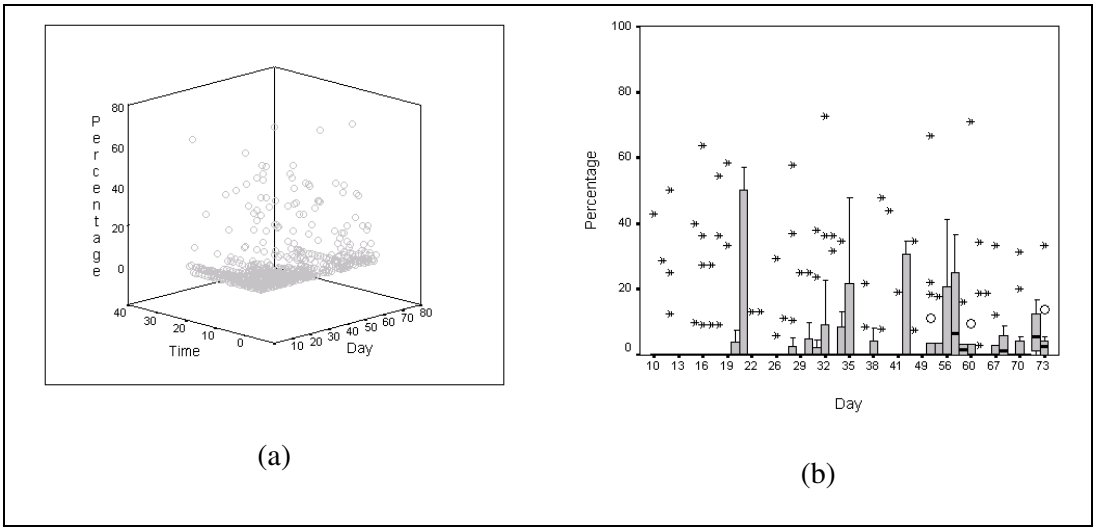


Figure 6.9 For Customer A of the real telecommunication data, (a) Percentage is plotted against Day and Time in the Scatter plot and (b) Percentage is plotted against Day in the Box plot.

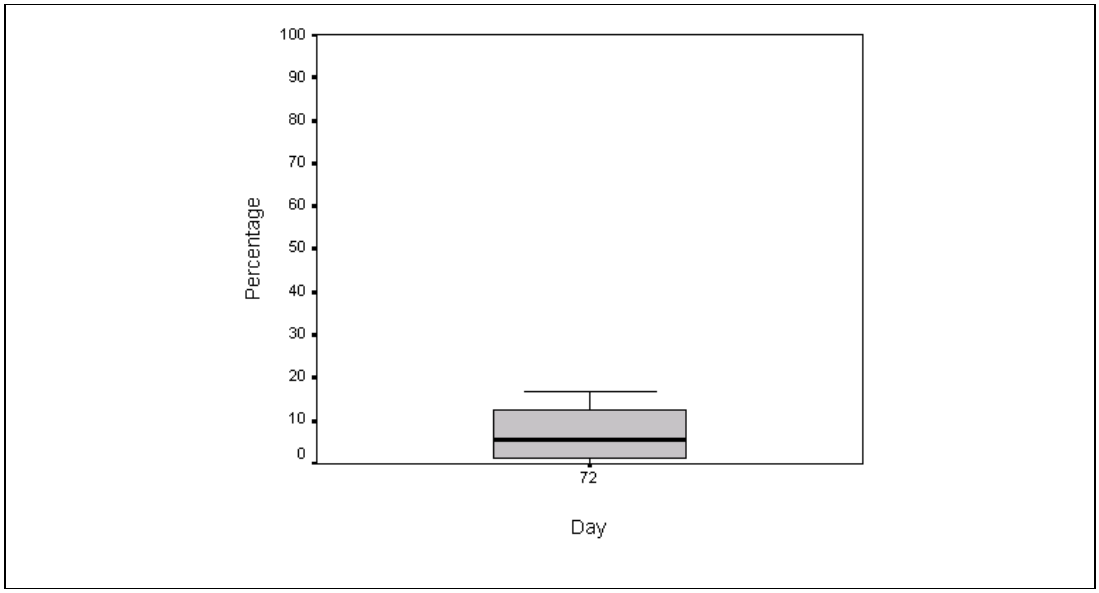


Figure 6.10 Percentage is plotted against Day, which is equals to 72, in the Box plot.

Table 6.3 Frequency table for Day equals to 72.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.00	1	25.0	25.0	25.0
	2.78	1	25.0	25.0	50.0
	8.33	1	25.0	25.0	75.0
	16.67	1	25.0	25.0	100.0
	Total	4	100.0	100.0	

Percentage equals to 17. Note that none of the calls has the Percentage equals to 100. Fraud call is the one that gives the Percentage equals to 100.

6.5 Summary

In the previous section, we highlighted the approach taken by Sain et al. (1999) in detecting nuclear explosions from the large number of background signals. The same approach is included in the algorithm for detecting fraud activities as mentioned in Chapter 4, which involves the testing of each observation whether it is an outlier or not i.e. $H_0 : \mathbf{x}_{n+1} \in \Pi$ versus $H_1 : \mathbf{x}_{n+1} \notin \Pi$. We showed its performance by using real telecommunication data and simulation data. The introduction of the "real time" fraud detection algorithm using GMM would help telecommunication companies to act upon fraud calls instantaneously instead of waiting until the 24 hours period is complete.

CHAPTER 7

CONCLUSION AND FUTURE RESEARCH

7.1 Conclusion

The damages caused by fraud activities to telecommunication companies are valued at millions (or billions) of dollars (Telecom and Network Security Review, 1997, Cahill et al., 2002, and Moreau et al., 1996) and the said activities could come in many forms. Superimposed fraud, which is the one of our interest, refers to the use of a service without permission and it would appear as phantom calls on a bill.

The number of literatures that discuss about pattern recognition method (namely Gaussian Mixed Model, GMM) used to detect fraud activities in telecommunication industry involving real data other than speech recognition's format is limited and GMM is difficult to apply (or implement) in real practice because we need to find the initial estimates of parameters to start Expectation Maximization (EM) algorithm and find the exact number of Gaussian components. Telekom Malaysia Berhad, a leading telecommunication company in Malaysia, via their current system or application believes the real data collected by them (e.g. duration and charging or billing) from its exchanges are contaminated by fraud activities and, since GMM is not included on the list of methods, there is no knowing if their findings are statistically correct. The following objectives for this study were derived based on the above problems. The first objective is to improve Gaussian Mixed Model (GMM) from its known (or current) weaknesses (or difficulties) such as finding the initial estimates of parameters to start

Expectation Maximization (EM) algorithm and finding the exact number of Gaussian components. The second objective is to introduce a new algorithm that is capable of detecting fraud activities (especially) in telecommunication industry and that incorporates the improvement as mentioned in the first objective. The third objective is to test the new algorithm (as well improved the EM algorithm for GMM) using simulation data and real data (e.g. duration and charging or billing) collected from Telekom Malaysia Berhad's exchanges that are believed to be contaminated by fraud activities.

Schlattmann (2003) suggested using a non-parametric bootstrapping approach to identify the right number, say k, of components in a GMM and subsequently choosing good initial values for the model parameters μ_i and σ_i^2, $i = 1,2,\ldots k$, in the EM algorithm. Wang et al. (2004) introduced an algorithm called the stepwise-split-and-merge EM algorithm to solve the said problem and Miloslavsky and Van Der Laan (2003) suggested using the minimization of the Kullback-Leiber distance between fitted mixture models and the true density as a method for estimating k where the said distance was estimated using cross validation. Other works on this topic can also be found, for example, in Zhuang et al. (1996), Lee et al. (2006) and Celeux and Soromenho (1996). We proposed an improved EM algorithm for GMM to identify the number of components of a GMM and estimate the parameters of the model using the kernel method. The first step uses kernel method, Silverman (1986), to determine the number of components, say K components, and to find Means as initial values to start EM algorithm for GMM. The second step executes EM algorithm for GMM to find the final estimates of parameters using $k=1$ number of components, Means obtained from the first step, and Variances fixed at 1 as initial values. The third step calculates log-likelihood function and Akaike Information Criteria (AIC) Akaike (1974) using final

estimates of parameters from the second step. The fourth step repeats the second and third steps for $k=2,\ldots,K$ number of components. The final step compares all (K) AICs obtained from the fourth step and the one that gives the minimum value is chosen (which gives the true or correct number of components). The performance of the algorithm via simulation is generally good but, as we expected, is affected by increasing percentages of overlapping of the Gaussian components. The final number of GMM component could clearly be determined via the concave-like shape of the AIC plot, which indicates that the AIC decreases to a minimum value and then increases as the number of components increases.

The idea used to give "birth" to the algorithm for detecting fraud calls is related to speaker identification, which involve the coding of the new speaker's recorded voice into vector, the calculation of the log-likelihood function for each training matrix, the comparison of all log-likelihood functions and the selection of the maximum one thus revealing the speaker's true identity. In our case, we use customer's call behavior in place of speaker's recorded voice. The algorithm for detecting fraud calls involves two steps. The first step performs, for each of the first seven days, the improved EM algorithm for GMM and save the final estimates of parameters in the database. For the 8^{th} day onwards, the second step uses the parameters saved in the database on the data set to find the maximum log-likelihood function, calculates the percentage of similarity coefficient and performs the updating process, which depends on the percentage of similarity coefficient. We used them on two variables namely duration and call charging (or billing) collected from 18 TM's customers (that fell victim to fraud activities on the 15^{th} of November 2011), which revealed interesting results. The percentage of TM's customers that support the findings made by TM's system decreases as the number of variables used increases. The downgrading, upgrading and

unchanging of the similarity group depend on whether using one or two variables. No downgrading to Low Similarity group is observed. The unchanging of Low Similarity group is observed for more than 10% of TM's customers. No Low Similarity group is observed for several of TM's customers depending on whether using one or two variables. Fraud activities might happen earlier than 15th of November 2011 due to the average number of similarity coefficient assigned to Low Similarity group when using two variables, which is greater than 10%. The results show the effectiveness of the proposed algorithm in detecting Low Similarity group (before and on the 15th of November 2011, which we believe they have close connection to fraud activity). Bolton and Hand (2002) (and Schonlau et al., 2001) said fraud (and computer intrusion) detection offers many challenges and opportunities to statisticians where they could make a very substantial and important contribution.

The proposed algorithm uses AIC to determine the number of components in GMM and we showed this task could also be performed by the hypothesis testing. The comparison between hypothesis testing and AIC using mathematical derivation and real telecommunication data revealed conflicting results under certain conditions due to the dependence of the former on log-likelihood function and the choice of the probability value α that gives $\chi^2_{\alpha,(k+k^*)p}$ and the latter on log-likelihood function. Hypothesis testing results are similar to those of AIC if α is set at a different value (i.e. other than 0.05).

The approach taken by Sain et al. (1999) in detecting nuclear explosions from the large number of background signals is included in the algorithm for detecting fraud activities that involves the testing of each observation whether it is an outlier or not i.e. $H_0 : \mathbf{x}_{n+1} \in \Pi$ versus $H_1 : \mathbf{x}_{n+1} \notin \Pi$ (as shown by several examples using real

telecommunication data and simulation data). Thus helping the telecommunication companies to act upon fraud calls instantaneously instead of waiting until the day is over.

7.2 Future research

Future research work will involve the following: the proposed algorithm will be tested on a bigger number of customers, the handling of files with small data points, twenty four (24) hours period for collecting customer's call detail record will be divided into 3 parts (sub-periods), similarity coefficient that falls under Moderate Similarity group will be further investigated (for possible fraud activities) and the use of other variables such as type of call (domestic and international). The suitability of the Gaussian Mixed Model Hidden Markov Model (refer to Bilmers, 1998, Rabiner, 1989, Juang and Rabiner, 1985, Box and Jenkins, 1976, Bidgoli, 2007) for type of call will be explored where S_1 and S_2 are the states assigned to the domestic and international call, respectively. The power of a test (Guenther, 1977) will be used to explain the behavior of the hypothesis testing especially when it conflicts with AIC (in terms of the number of components in GMM).

REFERENCES

Abbott, D.W., Matkovsky, I.P., & Elder, J.F. (1998). An evaluation of high-end data mining tools for fraud detection. *IEEE International Conference on Systems, Man, and Cybernetics*, 3, 2836-2841. doi:10.1109/ICSMC.1998.725092

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatics Control*, 19(6), 716-723. doi:10.1109/TAC.1974.1100705

Becker, R.A., Volinsky, C., & Wilks, A.R. (2010). Fraud detection in Telecommunications: History and Lessons Learned. *Technometrics*, 52(1), 20-33. doi:10.1198/TECH.2009.08136

Bidgoli, H. (2007). *The Handbook of Computer Networks, Volume 3, Distributed Networks, Network Planning, Control, Management, and New Trends and Applications*. New Jersey: John Wiley & Sons, Inc. doi:10.1002/9781118256107

Bihina Bella, M.A., Eloff, J.H.P., Olivier, M.S. (2005). Using the internet protocol detail record standard for next generation network billing and fraud detection. *Information Security South Africa 2005 New Knowledge Today Conference*, 1-11. ISBN 1-86854-625-X.

Bilmes, J.A. (1998). *A Gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian Mixture and Hidden Markov Models*. Technical Report, University of Berkeley, ICSI-TR-97-021.

Bolton, R.J, & Hand, D.J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, 17(3), 235-255.

Box, G.E.P, & Jenkins, G. (1976). *Time Series Analysis: Forecasting and Control (Revised Edition).* San Francisco: Holden-Day.

Box, G.E.P, & Muller, M.E. (1958). A note on the generating of random normal deviates. *Ann. Math. Statistic*, 29(2), 610-611.

Cahill, M.H., Lambert, D., Pinheiro, J.C., & Sun, D.X. (2002). Detecting fraud in the real world. In J.Abello, P.M. Pardalos, & M.G.C. Resende, (Eds.), *Handbook of Massive Datasets,* 911-929, Dordrecht: Kluwer.

Celeux, G., & Soromenho, G. (1996). An Entropy Criterion for assessing the number of clusters in a mixture model. *Journal of Classification*, 13(2), 195-212.

Cox, K.C., Erick, S.G., & Wills, G.J. (1997). Visual data mining: Recognizing telephone calling fraud. *Data Mining and Knowledge Discovery*, 1(2), 225-231.

Dempster, A.P., Laird, N.M. & Rubin, D.B. (1977). Maximum Likelihood from incomplete data via the EM algorithm. *Journal Royal Statistics Society Series B (Methodological)*, 39(1), 1-38.

Efron, B. & Tibshirani, R.J. (1993). *An introduction to the Bootstrap*. New York: Chapman and Hall.

Everitt, B.S. & Hand, D.J. (1981). *Finite Mixture Distributions.* London: Chapman and Hall.

Fishman, G.S. (2001). *Discrete-event simulation: modeling, programming, and analysis*. New York: Springer-Verlag.

Girolami, M., & Kaban, A. (2005). Sequential activity profiling: latent Dirichlet allocation of Markov Chains. *Data Mining and Knowledge Discovery*, 10, 175-196.

Gomez-Restrepo, J. & Cogollo-Florez, M.R. (2012). Detection of fraudulent transactions through a Generalized Mixed Linear Models. *Ing.cienc [online]*, 8(16), 221-237.

Guenther, W.C. (1977). Power and sample size for approximate Chi-Square tests. *The American Statistician*, 31(2), 83-85.

Han, J., Kamber, M. (2001). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

Hilas, C.S. (2009). Designing an expert system for fraud detection in private telecommunication networks. *Expert Systems with Applications*, 36, 11559-11569.

Hilas, C.S., & Mastorocostas, P.A. (2008). An application of supervised and unsupervised learning approaches to telecommunication fraud detection. *Knowledge Based Systems*, 21(7), 721-726.

Hollmen, J. and Tresp, V. (1998). Call based fraud detection in Mobile communication networks using a hierarchical regime-switching model. In Kearns, M., Solla, S. and Cohn, D. (Eds). Advances in Neural information processing systems II. *Proceedings of the 1998 Conference (NIPS 'II)*, 889-895. Massachusetts: MIT Press.

Hollmen, J. & Tresp, V. (2000). A hidden Markov model for metric and event-based data. *EUSIPCO 2000 – X European Signal Processing Conference*, 2, 737-740

Hollmen, J., Tresp, V. & Simula, O. (2000). A learning vector quantization algorithm for probabilistic models. *EUSIPCO 2000 – X European Signal Processing Conference*, 2, 721-724.

Jain, A.K., Duin, R.P.W., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 22(1), 4-37.

Johnson, R.A. & Wichern, D.W. (1998). *Applied Multivariate Statistical Analysis*. New Jersey: Prentice-Hall International Inc.

Juang, B.H., & Rabiner, L.R. (1985). Mixture Autoregressive Hidden Markov Models for Speech Signals. *IEEE Trans. On Acoust. Speech And Signal Processing*, 33(6), 1404-1413. doi:10.1109/TASSP.1985.1164727

Lee, Y., Lee, K.Y. & Lee, J. (2006). The estimating optimal number of Gaussian Mixtures based on incremental k-means for Speaker Identification. *International Journal of Information Technology*, 12(7), 13-21.

Linoff, G.S. (2004). Survival Data Mining for Customer Insight. *Intelligent Enterprise*, 7(12), 28-33.

Mardia, K.V., Kent, J.T. & Bibby, J.M. (1979). *Multivariate analysis*. London: Academic Press Inc.

McLachlan, G.J. (1987). On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Appl. Stat.*, 36, 318-324.

Miloslavsky, M. & Van Der Laan, M.J. (2003). Fittting of mixture with unspecified number of components using cross validation distance estimate. *Computational Statistics and Data Analysis*, 41, 413-428.

Moreau, Y., Preneel, B., Burge, P., Shawe-Taylor, J., Stoermann, C., & Cooke, C. (1996). Novel techniques for fraud detection in mobile communications. *ACTS Mobile Summit*, 1-6.

Pyle, D. (1999). *Data Preparation for Data Mining*. San Francisco: Morgan Kaufmann.

Rabiner, L.R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286. doi:10.1109/5.18626

Redner, R. & Walker, H. (1984). Mixture densities, maximum likelihood and the em algorithm. *SIAM Review*, 26(2), 195-239.

Reynolds, D.A. (1995). Automatic Speaker Recognition using Gaussian Mixture Speaker Models. *The Lincoln Laboratory Journal*, 8(2), 173-192.

Reynolds, D.A. & Rose, R.C. (1995). Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models. *IEEE Transactions on speech and audio processing*, 3(1), 72-83. doi:10.1109/89.365379

Rohanizadeh, S.S. & Moghadam, M.B. (2009). A proposed data mining methodology and its application to industrial procedures. *Journal of Industrial Engineering*, 4(2009), 37-50.

Sachs, L. (1984). *Applied Statistics: A Handbook of Techniques*. New York: Springer-Verlag.

Sain, S.R., Gray, H.L., Woodward, W.A., Fisk, & M.D. (1999). Outlier detection from a mixture distribution when training data are unlabeled. *Bulletin of the Seismological Society of America*, 89(1), 294-304.

Schlattmann, P. (2003). Estimating the number of components in a finite mixture model: the special case of homogeneity. *Computational Statistics and Data Analysis*, 41, 441-451.

Schonlau, M., Dumouchel, W., Ju, W., -H., Karr, A.F., Theus, M., & Vardi, Y. (2001). Computer intrusion: Detecting masquerades. *Statist. Sci.*, 16, 58-74.

Shawe-Taylor, J., Howker, K., Gosset, P., Hyland, M., Verrelst, H., Moreau, Y., Stoermann, C., &, Burge, P. (2000). Novel techniques for profiling and fraud detection in mobile telecommunications. In P.J.G. Lisboa, A. Vellido & B.Edisbury, (Eds), *Business Application of Neural Networks*, 113-139. Singapore: World Scientific.

Silverman, B.W. (1986). *Density estimation for statistics and data analysis*. London: Chapman and Hall.

Spiegel, M.R. (1974). *Shaum's outline series: Theory and problems of advanced calculus: SI (Metric) edition*. Singapore: Mc Graw-Hills Inc.

Taniguchi, M., Haft, M., Hollmen, J., & Tresp, V. (1998). Fraud detection in communications networks using neural and probabilistic methods. *1998 IEEE International Conference in Acoustics, Speech and Signal Processing (ICASSP '98)*, 2, 1241-1244.

Tsay, R.S. (2005). *Analysis of Financial Time Series: Financial Econometrics*. New Jersey: John Wiley and Sons, Inc.

Turkmen, M. (2013). Investigation of the relationship between academic and sport motivation orientations. *Middle-east Journal of Scientific Research*, 16(7), 1008-1014. doi:10.5829/idosi.mejsr.2013.16.07.765

Wang, H.X., Luo, B., Zhang, Q.B. & Wei, S. (2004). Estimation for the number of components in a mixture model using stepwise split-and-merge EM algorithm. *Pattern Recognition Letters*, 25, 1799-1809.

Xing, D., & Girolami, M. (2007). Employing latent Dirichlet allocation for fraud detection in telecommunications. *Pattern Recognition Letters*, 28, 1727-1734.

Xu, L. & Jordan, M.I. (1996). On convergence properties of the em algorithm for Gaussian mixtures. *Neural Computation*, 8, 129–151.

Xu, W., Pang, Y., Ma, J. Wang, S.-Y., Hao, G., Zeng, S., Qian, Y.-H. (2008). Fraud detection in telecommunication: a rough fuzzy set based approach. *2008 International Conference on Machine Learning and Cybernetics*, 3, 1249-1253. doi:10.1109/CMLC.2008.4620596

Zhuang, X., Huang, Y., Palaniappan, K. & Zhao, Y. (1996). Gaussian Mixture Density Modelling, Decomposition, and Applications. *IEEE Transactions on Image Processing*, 5(9), 1293-1302. doi:10.1109/83.535841

Zuber, Md., Suman, N., Gouse Pasha, Md. & Adam, Md. (2013). A study on data mining approaches. *International Journal of Emerging Trends in Engineering & Development*, 3(1), 676-683.

APPENDIX A

The results presented in this thesis were produced from using the computer facility in the Institute of Mathematical Sciences (University of Malaya) and Telekom Research and Development Sdn. Bhd. The applications used were SPSS, Microsoft® Excel and NetBeans, which is an Integrated Development Environment, IDE, for developing primarily with Java and it is an application platform framework for Java desktop applications and others. The first two applications were used for producing graphical presentations. Due to its size, only a portion of Java script will be displayed.

```
class EM {

    double sum1, sum2, sum3, sum4;
    double maximum;
    double a_ih,a_il;
    double eq1,eq2,eq3;
    double diff1, diff2, diff3;
    double epsilon=0.000001;
    double[][]alpha;
    double[][][]mu;
    double[][][] sigma;
    double[][]sigma_determinant;
    double[][][] sigma_inverse;
    double[][][]prob;
    double[][] x;
    int dim;
    int h,i,j,k,l;
    int count;
    int h1,h2;
    int N,K;
    int p, P;
    int iterate;
    int Stop;
    int Stop_limit;
    //
    String[][]statement;
    int count_statement=0;
    //
    PrintWriter output=null;
    BufferedReader input = null;
    String character=null;
    StringTokenizer token;
    //
    String filename=null;
    DecimalFormat sixDigits;
```

```
public int EM_Algorithm()
{
    //
    try
    {
        //
        //P=1;//fixed
        //
        /*if (Math.abs(Type) == 1)
        {
           K=2; dim=1;
        }
        else if (Math.abs(Type) == 2)
        {
           K=2; dim=1;
        }
        else if (Math.abs(Type) == 3)
        {
            K=6; dim=1;
            //original
           //K=3; dim=1;
           //original

        }
        else if (Math.abs(Type) == 4)
        {
           K=3; dim=4;
        }*/
        //
        //
        //
        prob=new double[P+1][][];
        for (i=0;i<=P;i++) prob[i]=new double[N+1][];
        for (i=0;i<=P;i++)
            for (j=0;j<=N;j++)
                prob[i][j]=new double[K+1];
        //
        //
        /*
         1  2  3  4
         5  6  7  8
         9  10 11 12
         13 14 15 16
         (4X4 matrix)
         */
        //
        sigma_inverse=new double[P+1][][];
        for (i=0;i<=P;i++) sigma_inverse[i]=new double[K+1][];
        for (i=0;i<=P;i++)
            for(j=0;j<=K;j++)
                sigma_inverse[i][j]=new double[dim*dim+1];
        //
        sigma_determinant=new double[P+1][];
        for (i=0;i<=P;i++)
            sigma_determinant[i]=new double[K+1];
        //
        //Initialize parameters
        /*for (k=1; k<=K; k++)
        {
                        if (Type < 0)
                        {
                            //
                            if (Type == -1)
```

```
{
    //
    if (k==1)
    {
        alpha[0][k]=0.5;
        mu[0][k][1]=0.0;
        sigma[0][k][1]=1.0;
    }
    else
    {
       alpha[0][k]=0.5;
       mu[0][k][1]=2.0;
       sigma[0][k][1]=1.0;
    }
    //
}
else if (Type == -2)
{
    //
    if (k==1)
    {
        alpha[0][k]=0.5;
        mu[0][k][1]=0.0;
        sigma[0][k][1]=1.0;
    }
    else
    {
       alpha[0][k]=0.5;
       mu[0][k][1]=1.5;
       sigma[0][k][1]=1.0;
    }
    //
}
else if (Type == -3)
{

    if (k==1)
    {
        alpha[0][k]=1.0/6.0;
        mu[0][k][1]=0.0;
        sigma[0][k][1]=1.0;
    }
    else if (k==2)
    {
       alpha[0][k]=1.0/6.0;
       mu[0][k][1]=0.0;
       sigma[0][k][1]=1.0;
    }
    else if (k==3)
    {
       alpha[0][k]=1.0/6.0;
       mu[0][k][1]=0.0;
       sigma[0][k][1]=1.0;
    }
    else if (k==4)
    {
       alpha[0][k]=1.0/6.0;
       mu[0][k][1]=0.0;
       sigma[0][k][1]=1.0;
    }
    else if (k == 5)
    {
       alpha[0][k]=1.0/6.0;
       mu[0][k][1]=0.0;
```

```
                                        sigma[0][k][1]=1.0;
                                    }
                                    else
                                    {
                                        alpha[0][k]=1.0/6.0;
                                        mu[0][k][1]=0.0;
                                        sigma[0][k][1]=1.0;
                                    }


                                }
                                else//Type == -4
                                {
                                        if (k==1)
                                        {
                                            alpha[0][k]=0.33;
                                            //
                                            mu[0][k][1]=4;
                                            mu[0][k][2]=4;
                                            mu[0][k][3]=2;
                                            mu[0][k][4]=1;
                                            //
                                            sigma[0][k][1]=1;

sigma[0][k][2]=sigma[0][k][5]=0;

sigma[0][k][3]=sigma[0][k][9]=0;

sigma[0][k][4]=sigma[0][k][13]=0;
                                            sigma[0][k][6]=1;

sigma[0][k][7]=sigma[0][k][10]=0;

sigma[0][k][8]=sigma[0][k][14]=0;
                                            sigma[0][k][11]=1;

sigma[0][k][12]=sigma[0][k][15]=0;
                                            sigma[0][k][16]=1;
                                        }
                                        else if (k==2)
                                        {
                                           alpha[0][k]=0.33;
                                            //
                                            mu[0][k][1]=7;
                                            mu[0][k][2]=2;
                                            mu[0][k][3]=3;
                                            mu[0][k][4]=2;
                                            //
                                            sigma[0][k][1]=1;

sigma[0][k][2]=sigma[0][k][5]=0;

sigma[0][k][3]=sigma[0][k][9]=0;

sigma[0][k][4]=sigma[0][k][13]=0;
                                            sigma[0][k][6]=1;

sigma[0][k][7]=sigma[0][k][10]=0;

sigma[0][k][8]=sigma[0][k][14]=0;
                                            sigma[0][k][11]=1;

sigma[0][k][12]=sigma[0][k][15]=0;
                                            sigma[0][k][16]=1;
```

```
                                        }
                                        else
                                        {
                                           alpha[0][k]=1.0-
(0.33+0.33);
                                            //
                                            mu[0][k][1]=8;
                                            mu[0][k][2]=4;
                                            mu[0][k][3]=5;
                                            mu[0][k][4]=3;
                                            //
                                            sigma[0][k][1]=1;

sigma[0][k][2]=sigma[0][k][5]=0;

sigma[0][k][3]=sigma[0][k][9]=0;

sigma[0][k][4]=sigma[0][k][13]=0;
                                            sigma[0][k][6]=1;

sigma[0][k][7]=sigma[0][k][10]=0;

sigma[0][k][8]=sigma[0][k][14]=0;
                                            sigma[0][k][11]=1;

sigma[0][k][12]=sigma[0][k][15]=0;
                                            sigma[0][k][16]=1;
                                        }
                                    }
                                    //
                                }
                                else if (Type == 1)
                                {
                                    if (k==1)
                                    {
                                        alpha[0][k]=0.4;
                                        mu[0][k][1]=0.0;
                                        sigma[0][k][1]=1.0;
                                    }
                                    else
                                    {
                                       alpha[0][k]=0.6;
                                       mu[0][k][1]=2.0;
                                       sigma[0][k][1]=0.25;
                                    }
                                }
                                else if (Type == 2)
                                {
                                    if (k==1)
                                    {
                                        alpha[0][k]=0.85;
                                        mu[0][k][1]=0.0;
                                        sigma[0][k][1]=1.0;
                                    }
                                    else
                                    {
                                       alpha[0][k]=0.15;
                                       mu[0][k][1]=2.0;
                                       sigma[0][k][1]=0.25;
                                    }

                                }
                                else if (Type == 3)
                                {
```

```
                                    if (k==1)
                                    {
                                        alpha[0][k]=0.33;
                                        mu[0][k][1]=0.0;
                                        sigma[0][k][1]=1.0;
                                    }
                                    else if (k==2)
                                    {
                                       alpha[0][k]=0.33;
                                       mu[0][k][1]=-1.0;
                                       sigma[0][k][1]=0.25;
                                    }
                                    else
                                    {
                                       alpha[0][k]=1.0-(0.33+0.33);
                                       mu[0][k][1]=4.0;
                                       sigma[0][k][1]=4.0;
                                    }
                                }
                                else//Type == 4
                                {
                                    if (k==1)
                                    {
                                        alpha[0][k]=0.33;
                                        //
                                        mu[0][k][1]=5.01;
                                        mu[0][k][2]=3.43;
                                        mu[0][k][3]=1.46;
                                        mu[0][k][4]=0.25;
                                        //
                                        sigma[0][k][1]=0.12;
                                        sigma[0][k][2]=sigma[0][k][5]=0.1;

sigma[0][k][3]=sigma[0][k][9]=0.02;

sigma[0][k][4]=sigma[0][k][13]=0.01;
                                        sigma[0][k][6]=0.14;

sigma[0][k][7]=sigma[0][k][10]=0.01;

sigma[0][k][8]=sigma[0][k][14]=0.13;
                                        sigma[0][k][11]=0.03;

sigma[0][k][12]=sigma[0][k][15]=0.01;
                                        sigma[0][k][16]=0.3;
                                    }
                                    else if (k==2)
                                    {
                                       alpha[0][k]=0.30;
                                        //
                                        mu[0][k][1]=5.91;
                                        mu[0][k][2]=2.78;
                                        mu[0][k][3]=4.2;
                                        mu[0][k][4]=1.3;
                                        //
                                        sigma[0][k][1]=0.27;
                                        sigma[0][k][2]=sigma[0][k][5]=0.1;

sigma[0][k][3]=sigma[0][k][9]=0.18;

sigma[0][k][4]=sigma[0][k][13]=0.05;
                                        sigma[0][k][6]=0.09;

sigma[0][k][7]=sigma[0][k][10]=0.09;
```

```
sigma[0][k][8]=sigma[0][k][14]=0.04;
                                    sigma[0][k][11]=0.2;

sigma[0][k][12]=sigma[0][k][15]=0.06;
                                    sigma[0][k][16]=0.03;
                                }
                                else
                                {
                                   alpha[0][k]=1.0-(0.33+0.30);
                                    //
                                    mu[0][k][1]=6.54;
                                    mu[0][k][2]=2.95;
                                    mu[0][k][3]=5.48;
                                    mu[0][k][4]=1.98;
                                    //
                                    sigma[0][k][1]=0.38;

sigma[0][k][2]=sigma[0][k][5]=0.09;
                                    sigma[0][k][3]=sigma[0][k][9]=0.3;

sigma[0][k][4]=sigma[0][k][13]=0.06;
                                    sigma[0][k][6]=0.11;

sigma[0][k][7]=sigma[0][k][10]=0.08;

sigma[0][k][8]=sigma[0][k][14]=0.05;
                                    sigma[0][k][11]=0.32;

sigma[0][k][12]=sigma[0][k][15]=0.07;
                                    sigma[0][k][16]=0.08;
                                }
                            }
                            //
            }//end k
            */
            //
            //
            Stop = 0; iterate=0;
            while (Stop == 0 && (iterate <= Stop_limit))//while (Stop
== 0 && (iterate <= 200000))
            {
                iterate=iterate+1;
                //**Expectation**
                for (i=1;i<=N;i++)
                {
                    sum3=0.0;
                    for (k=1;k<=K;k++)
                    {
                        //
                        if (dim > 1)
                        {
                            //
                            if
(Cholesky_inverse(sigma[0][k],dim,sigma_inverse[0][k]) != 0)
                            {
                                return -1;
                            }

sigma_determinant[0][k]=Determinant(sigma[0][k],dim);
                            //
                        }
                        else
                        {
```

```
sigma_inverse[0][k][1]=(1.0/sigma[0][k][1]);
                            sigma_determinant[0][k]=sigma[0][k][1];//
                        }
                        //
                        sum1=0.0;
                        for (h1=1;h1<=dim;h1++)
                        {
                            sum2=0.0;
                            for (h2=1;h2<=dim;h2++)
                            {
                                sum2=sum2+(
                                (x[i][h1]-mu[0][k][h1])*
                                (sigma_inverse[0][k][(h1-1)*dim+h2])*
                                (x[i][h2]-mu[0][k][h2]));
                            }
                            sum1=sum1+sum2;
                        }
                        eq1=Math.exp(-sum1/2.0);

eq2=sigma_determinant[0][k]*Math.pow(2.0*Math.PI, dim);
                        eq2=1.0/Math.sqrt(eq2);
                        //
                        sum3=sum3+(
                        alpha[0][k]*
                        eq1*
                        eq2);
                    }
                    //
                    //
                    for (k=1;k<=K;k++)
                    {
                        //
                        if (dim > 1)
                        {
                            //
                            if
(Cholesky_inverse(sigma[0][k],dim,sigma_inverse[0][k]) != 0)
                            {
                                return -1;
                            }

sigma_determinant[0][k]=Determinant(sigma[0][k],dim);
                            //
                        }
                        else
                        {
                            //

sigma_inverse[0][k][1]=(1.0/sigma[0][k][1]);
                            sigma_determinant[0][k]=sigma[0][k][1];//
                            //
                        }
                        //
                        sum1=0.0;
                        for (h1=1;h1<=dim;h1++)
                        {
                            sum2=0.0;
                            for (h2=1;h2<=dim;h2++)
                            {
                                sum2=sum2+(
                                (x[i][h1]-mu[0][k][h1])*
                                (sigma_inverse[0][k][(h1-1)*dim+h2])*
                                (x[i][h2]-mu[0][k][h2]));
```

```
                            }
                            sum1=sum1+sum2;
                        }
                        eq1=Math.exp(-sum1/2.0);

eq2=sigma_determinant[0][k]*Math.pow(2.0*Math.PI, dim);
                        eq2=1.0/Math.sqrt(eq2);
                        prob[0][i][k]=(alpha[0][k]*eq1*eq2)/sum3;
                        //
                    }
                }//end i
                //
                //**Maximization**
                for (k=1; k<=K; k++)
                {

                    //1st Equation
                    sum1=0.0;
                    for (i=1;i<=N;i++)
                    {
                        sum1=sum1+prob[0][i][k];
                    }
                    alpha[1][k]=sum1/((double) N);
                    //
                    //2nd Equation
                    for (j=1; j<=dim; j++)
                    {
                        sum1=sum2=0.0;
                         for (i=1;i<=N;i++)
                        {
                            sum1=sum1+(prob[0][i][k]*x[i][j]);
                        }
                         for (i=1;i<=N;i++)
                        {
                            sum2=sum2+prob[0][i][k];
                        }
                        mu[1][k][j]=sum1/sum2;
                    }
                    //
                    //3rd Equation
                    for (h=1;h<=dim;h++)
                    {
                        for (l=1;l<=dim;l++)
                        {
                            //
                            sum1=sum2=0.0;
                            for (i=1;i<=N;i++)
                            {
                                a_ih=x[i][h]-mu[1][k][h];
                                a_il=x[i][l]-mu[1][k][l];
                                sum1=sum1+(prob[0][i][k]*a_ih*a_il);
                            }
                            for (i=1;i<=N;i++)
                            {
                                sum2=sum2+prob[0][i][k];
                            }
                            sigma[1][k][(h-1)*dim+l]=sum1/sum2;
                            //
                        }
                    }
                    //

                }//end k
                //
```

```
                //checking for convergence
                //iteration iterate and iterate-1 parameters are
compared. If difference is less than say 1e-6, convergence is
achieved.
                count=0;
                for (k=1; k<=K; k++)
                {
                        //1st equation
                        diff1=Math.abs(alpha[1][k]-alpha[0][k]);
                        if (diff1 < epsilon)
                        {
                            count=count+1;
                        }
                        //number of alphas = K
                        //2nd equation
                        for (j=1; j<=dim; j++)
                        {
                            diff2=Math.abs(mu[1][k][j]-mu[0][k][j]);
                            if (diff2 < epsilon)
                            {
                                count=count+1;
                            }
                        }
                        //number of mus = K*dim
                        //3rd equation
                        for (h=1;h<=dim;h++)
                        {
                            for (l=1;l<=dim;l++)
                            {
                                diff3=Math.abs(sigma[1][k][(h-
1)*dim+l]-sigma[0][k][(h-1)*dim+l]);
                                if (diff3 < epsilon)
                                {
                                    count=count+1;
                                }
                            }
                        }
                        //number of sigmas = K*dim*dim
                        //
                }//end k
                //
                if (count == (K+(K*dim)+(K*dim*dim)))
                {
                        //
                        Stop = 1;
                        //

statement[count_statement][1]=statement[count_statement][1]+"(iteratio
n# "+String.valueOf(iterate)+")";
                        //
                         for (k=1; k<=K; k++)
                        {
                                character=null;
                                //1st equation

character="(k="+String.valueOf(k)+",alpha="+String.valueOf(alpha[1][k]
);//alpha
                                //
                                //2nd equation
                                for (j=1; j<=dim; j++)
                                {

character=character+",mu_"+String.valueOf(j)+
```

```
"="+String.valueOf(mu[1][k][j]);//mu
                                    }
                                    //
                                    //3rd equation
                                    for (h=1;h<=dim;h++)
                                    {
                                        for (l=1;l<=dim;l++)
                                        {
                                            character=character+",sigma_"+
                                            String.valueOf(h)+"_"+
                                            String.valueOf(l)+

"="+String.valueOf(sigma[1][k][(h-1)*dim+l]);//sigma
                                        }
                                    }
                                    //
                                    character=character+")";
                                    //

//count_statement++;statement[count_statement][1]=(character);

statement[count_statement][1]=statement[count_statement][1]+(character
);
                                    //
                            }//end k
                            //

                    }
                    else
                    {
                            //
                            Stop=0;
                            //
                             for (k=1; k<=K; k++)
                            {
                                    //1st equation
                                    alpha[0][k]=alpha[1][k];
                                    //
                                    //2nd equation
                                    for (j=1; j<=dim; j++)
                                    {
                                        mu[0][k][j]=mu[1][k][j];
                                    }
                                    //
                                    //3rd equation
                                    for (h=1;h<=dim;h++)
                                    {
                                        for (l=1;l<=dim;l++)
                                        {
                                            sigma[0][k][(h-
1)*dim+l]=sigma[1][k][(h-1)*dim+l];
                                        }
                                    }
                                    //
                                    //
                            }//end k
                            //
                    }//end if
                    //
                }//end while
                //
                //
        }
```

```
        catch (ArithmeticException arithmeticException)
        {

count_statement++;statement[count_statement][1]=String.valueOf(arithme
ticException);
            return -1;
        }
        //
        if (Stop == 1) return -100;
        //
        return 0;
        //
    }
```

APPENDIX B

Due to its size, we tabulate only a portion of customer's call detail record (cdr) supplied by Telekom Malaysia Berhad. SERVICE NUMBER and DIALED DIGITS are not revealed to ensure confidentiality.

SERVICE NUMBER	DIALED DIGITS	AREA CODE	COUNTRY CODE	SEIZE TIME	DURATION	SYSTEM CHARGING
X	Y	null	675	1/03/2011 14:05	000054 00	3.6
X	Y	null	675	1/03/2011 15:44	000430 00	18
X	Y	null	675	1/03/2011 15:56	000049 00	3.6
X	Y	19	60	31/03/2011 8:41	000339 00	0.8
X	Y	19	60	31/03/2011 9:27	000035 00	0.5
X	Y	19	60	31/03/2011 9:43	000048 00	0.6
X	Y	16	60	31/03/2011 9:50	000031 00	0.2
X	Y	16	60	31/03/2011 10:54	000138 00	0.5
X	Y	null	675	31/03/2011 11:26	000003 00	0.4
X	Y	null	675	31/03/2011 11:27	000057 00	4
X	Y	16	60	31/03/2011 12:46	000002 00	0.1
X	Y	12	60	31/03/2011 14:03	000041 00	0.5
X	Y	19	60	31/03/2011 14:29	000233 00	0.8
X	Y	19	60	31/03/2011 14:40	000029 00	0.4
X	Y	null	675	31/03/2011 14:47	000032 00	2.4
X	Y	19	60	31/03/2011 15:37	000002 00	0.1
X	Y	19	60	31/03/2011 15:42	000230 00	1.8
X	Y	3	60	31/03/2011 15:55	000251 00	2.5
X	Y	19	60	31/03/2011 16:04	000202 00	1.5
X	Y	84	60	31/03/2011 16:22	000022 00	0.1
X	Y	19	60	30/05/2011 8:50	000015 00	0.1
X	Y	19	60	30/05/2011 8:54	000031 00	0.2
X	Y	13	60	30/05/2011 9:20	000508 00	1.6
X	Y	13	60	30/05/2011 9:27	000003 00	0.1