Applying an Improved MRPS-GMM Method to Detect Temporal Patterns in Dynamic Data System

Shaobo Wang
Marquette University

APPLYING AN IMPROVED MRPS-GMM METHOD TO DETECT TEMPORAL PATTERNS IN DYNAMIC DATA SYSTEM

by

Shaobo Wang, B.S.

A Thesis submitted to the Faculty of the
Graduate School, Marquette University,
in Partial Fulfillment of the Requirements for the
Degree of Master of Science

Milwaukee, Wisconsin

December 2013

ABSTRACT

APPLYING AN IMPROVED MRPS-GMM METHOD TO DETECT TEMPORAL PATTERNS IN DYNAMIC DATA SYSTEM

Shaobo Wang

Marquette University, 2013

The purpose of this thesis is to introduce an improved approach for the temporal pattern detection, which is based on the Multivariate Reconstructed Phase Space (MRPS) and the Gaussian Mixture Model (GMM), to overcome the disadvantage caused by the diversity of shapes among different temporal patterns in multiple nonlinear time series. Moreover, this thesis presents an applicable software program developed with MATLAB for users to utilize this approach.

A major study involving dynamic data systems is to understand the correspondence between events of interest and predictive temporal patterns in the output observations, which can be used to develop a mechanism to predict the occurrence of events. The approach introduced in this thesis employs Expectation-Maximization (EM) algorithm to fit a more precise distribution for the data points embedded in the MRPS. Furthermore, it proposes an improved algorithm for the pattern classification process. As a result, the computational complexity will be reduced.

A recently developed software program, MATPAD, is presented as a deliverable application of this approach. The GUI of this program contains specific functionalities so that users can directly implement the procedure of MRPS embedding and fit data distribution with GMM. Moreover, it allows users to customize the related parameters for specific problems so that users will be able to test their own data.

ACKNOWLEDGMENTS

Shaobo Wang, B.S.

I would like to give my deep gratitude to my academic advisor, Professor Xin Feng. This thesis won't be completed without his support, guidance and encouragement. I have learned not only his insightful vision of academic research, but also the wisdom of life through my study at Marquette University.

In addition, this research has benefited from Dr. Povinelli's and Dr. Wenjing Zhang's work in this field. Their kind help and inspirations are gratefully acknowledged. I would also like to thank the committee members, Professor Edwin Yaz and Professor Naveen Bansal, for their assistance on my research. Special thanks to Professor George Corliss for his kind support, patience, and encouragement during the past two years.

I am grateful to the EECE Department at Marquette University for its financial support and to the members of GasDay lab for many interesting discussions and friendships.

Last, I want to give my sincere thanks to my girlfriend, Miss Ying Zhang, for her consistent support, love and sacrifice. I would also like to thank my parents for their support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 The Dynamic Data System

The concept of Dynamic Data System (DDS) was proposed as a new modeling method by Professor S. M. Wu in 1970s [1]. Although the detailed definition varies among different research areas, a notable property of DDS is that the system observations are in a form of time series with dynamics, which is called dynamic data [2]. Here, “dynamics” is a term to describe the correlations between the current observations and the previous ones on the timeline. In general, the basic structure of DDS can be shown as:

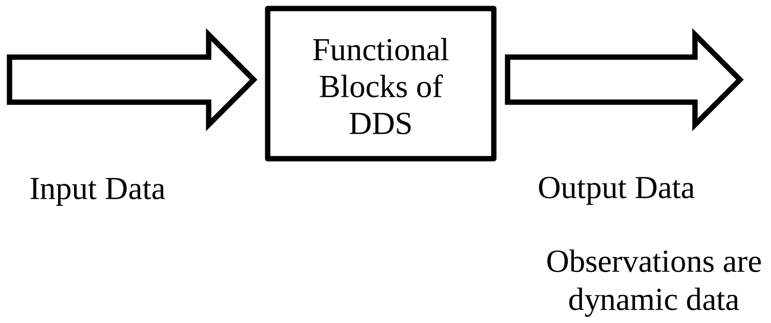


Figure 1.1 Basic structure of DDS

DDS is common and widely used in the real world. However, it is impossible for people to understand every detailed part in the system in most cases. In other words, this kind of DDS has to be seen as a “black box”. For example, an integrated circuit (IC) chip can be considered as a typical DDS. However, due to the reason of patents, the inner structure of some important components in the chip is protected and can’t be examined. Sometimes, we have to measure the system observations of input and output signals in multiple test process to understand the functionalities of those components. Meanwhile, lacking of applicable methods or facilities for measurement could be another problem that will make it difficult to study a DDS as well.

So, finding a practical method for DDS research based on “black box” testing is very necessary to overcome the obstacles mentioned above that prevent people from analyzing DDS effectively. Although it may not provide accurate parameter estimations to model the inner system, the method will contribute to the qualitative analysis to the system structure which is unacquainted to people.

1.2 Research Background

A major research direction on DDS is the detection of temporal patterns [3]. A temporal pattern is defined as a segment of signals that recurs frequently in the whole time series data [4]. In other words, temporal patterns are some different segments of a sequence that share a similar time-ordered structure. To discuss the use of temporal patterns, let’s see the concept of “event of interest” first.

Assume an output variable of a DDS can be expressed as a time series with N observations as:

$$\mathbf{x}(t) = \begin{bmatrix} x_1 & x_2 & \dots & x_t & \dots & x_N \end{bmatrix}, \tag{1.1}$$

where x_t is a single observation at time t. In this time sequence, some observations may have a special property that is different from the rest, such as their values could be above or below a certain threshold. This kind of observation is called “event of interest”, which implies about the change of state in the underlying DDS. Predicting the occurrence of events is very important for many applications. For example, in financial prediction, there is a significant interest in determining the timings of positions of securities [5]. However, most DDSs in the real world, such as the stock market, are extremely complex and chaotic so that it is hard to establish an accurate mathematical model to simulate the dynamics that causes the events to occur.

Despite the complexity of the DDS, these events can be related very closely to the temporal patterns which are lying in multivariate time sequences. If we can actually find there is a certain correspondence between events and temporal patterns, then it would be possible for us to use the detected temporal patterns to predict the occurrence of events.

There are two major research directions in detecting temporal patterns: the univariate and multivariate approaches. The univariate approach focuses on discovering the correspondence between events and patterns within the same time sequence, such as the innovative time series data mining (TSDM) method introduced by Povinelli and Feng in [3], which laid the foundation for the subsequent research. The multivariate approaches proposed in [6] by Zhang and Feng can be considered as an extension of the univariate approaches to solve the problem of analyzing the dynamic relationship between events and temporal patterns in multiple data sequences.

1.3 Statement of Problem

Assume that the observations of each output variable of a multivariate DDS can be expressed as:

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_m(t) \end{bmatrix}, \tag{1.2}$$

where

each element $\mathbf{x}(t)$ is a univariate time sequence representing one of the output variables of the system;

m is the total number of variables;

t is the time index;

In practical multivariate applications, one of the output variables can be considered as target sequence in which the events of interest will be defined and discovered. For most problems in the area of data mining and pattern analysis, people are not only interested in detecting the events in the target sequence, but also in exploring the causal relationship between the underlying factors and variables represented by other related sequences. Therefore, the expression can be re-written as:

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}_1(t) \\ \vdots \\ \mathbf{x}_m(t) \\ \mathbf{x}_e(t) \end{bmatrix}, \tag{1.3}$$

where $\mathbf{x}_e(t)$ denotes the target sequence. If each output variable has N observations; $\mathbf{X}(t)$ can be seen as an $(m+1)\times N$ matrix, where each column consists of the observations in each output variables at time *t*.

Several pattern identification approaches based on the Reconstructed Phase Space (RPS) [7] [8] have been proposed. These approaches proved that the RPS embedding is capable of representing temporal patterns in nonlinear dynamic data which is chaotic and irregular. The algorithm based on the RPS embedding typically involves selecting the embedding dimension [9] and time delay [10]. The following equation is an example of the embedding vectors with embedding dimension Q and time delay τ for a single time sequence $\mathbf{x}(t)$ as (1.1):

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_t \\ \vdots \\ \mathbf{x}_{N-(Q-1)\tau} \end{bmatrix} = \begin{bmatrix} x_1 & x_{1+\tau} & \cdots & x_{1+(Q-1)\tau} \\ x_2 & x_{2+\tau} & \cdots & x_{2+(Q-1)\tau} \\ \vdots & \vdots & \ddots & \vdots \\ x_t & x_{t+\tau} & \cdots & x_{t+(Q-1)\tau} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-(Q-1)\tau} & x_{N-(Q-2)\tau} & \cdots & x_N \end{bmatrix} \tag{1.4}$$

Each row of the right side matrix can be considered as an embedding vector, which can be seen as the coordinates of a point. By this method, a time sequence can be described as a set of points embedded in a Q-dimensional RPS. To keep it consistent to the previous research, in this thesis, we will apply the backward RPS embedding where an embedding vector is defined as:

$$\mathbf{x}_t = \begin{bmatrix} x_{t-(Q-1)\tau} & \cdots & x_{t-\tau} & x_t \end{bmatrix} \tag{1.5}$$

Actually, these two kinds of expressions for the embedding vectors are equivalent where the only difference is the subscript.

As an extension of this univariate RPS embedding method, the multivariate RPS (MRPS) embedding method proposed by Zhang and Feng in [6] makes it possible to embed multiple time sequences simultaneously into the same RPS. The detailed procedure will be discussed in Chapter 2.

According to the approach in [6], the embedding points will be categorized into three different states—event, normal and pattern by the event characterization function [11]. A Gaussian mixture model can be applied to fit the distribution of these points in the RPS. This MRPS-GMM approach has been tested to be efficient for detecting temporal patterns among multiple related time sequences. However, there are still several problems need to be solved. For example, the current method is not sufficiently accurate to handle the diversity of shapes among different temporal patterns. Meanwhile, the computational amount needs to be reduced for the classifier training process. Moreover, a deliverable application for temporal pattern detection is still missing.

Therefore, the primary objective of this thesis is to propose some contributive improvements on the existing methods and make an applicable framework to overcome the obstacles mentioned above. The overview of the proposed work is:

1) Propose a series of improvements on the Gaussian mixture model (GMM) applied in the existing MRPS-GMM approach to handle the situation when patterns with different shapes gather into separated regions in the RPS after the embedding process.
2) Reduce the computational amount by reducing the number of unknown coefficients in the classifier.

3) Develop a deliverable software program with an applicable framework for wide utilization.

This thesis will propose a series of discussion of these feasible solutions in the following chapters.

1.4 Thesis Outline

This thesis includes six chapters:

Chapter 1 discusses the background of DDS, the problem statement, the current status and the preview of the proposed work.

Chapter 2 will present a literature review for the fundamental theory of temporal pattern detection and event identification. Meanwhile, several contributive approaches developed in previous research will be discussed in this chapter.

Chapter 3 will analyze the existing GMM-based approach and propose the improved GMM to overcome the misclassifying problem caused by the diversity of shapes among different patterns embedded in the MRPS. The improved approach can fit a more precise probability density function to describe the distribution of the embedded temporal patterns. As a result, it will help the classifier to enhance the accuracy rate. Meanwhile, improvements on the classification and optimization stages will be proposed to make the computational process more efficient. In addition, performance of the newly proposed approach will be evaluated with simulated data.

Chapter 4 will introduce a newly developed software program based on the improved MRPS-GMM approach to detect temporal patterns in multivariate DDS. This program applies a computational framework based on MATLAB development environment. Meanwhile, as a deliverable software program, it can be executed on

Windows platforms that do not have MATLAB installed. The graphic user interface (GUI) of this software program is designed for users to easily understand the basic process of temporal pattern detection. The detailed functions and the operation procedure will be discussed.

Chapter 5 will perform several experiments by applying datasets from real world problems to evaluate the performance of the improved MRPS-GMM method.

Conclusions and suggestions of future work are included in Chapter 6.

CHAPTER 2 LITERATURE REVIEW

The fundamental theories applied in the thesis are based on the successive research from many scholars who have been dedicating themselves on this Temporal Pattern Detection subject for more than a decade of years. This chapter will take a review of conventional research related to time series analysis at first. Then, it will review those methods presented in previous research which are contributive to temporal pattern detection in the DDS.

2.1 Review of Conventional Time Series Analysis

The primary research objective for time series analysis is to develop mathematical models that reveal patterns in the underlying system and make prediction. An example of a conventional procedure for time series analysis is shown as the following flow chart:

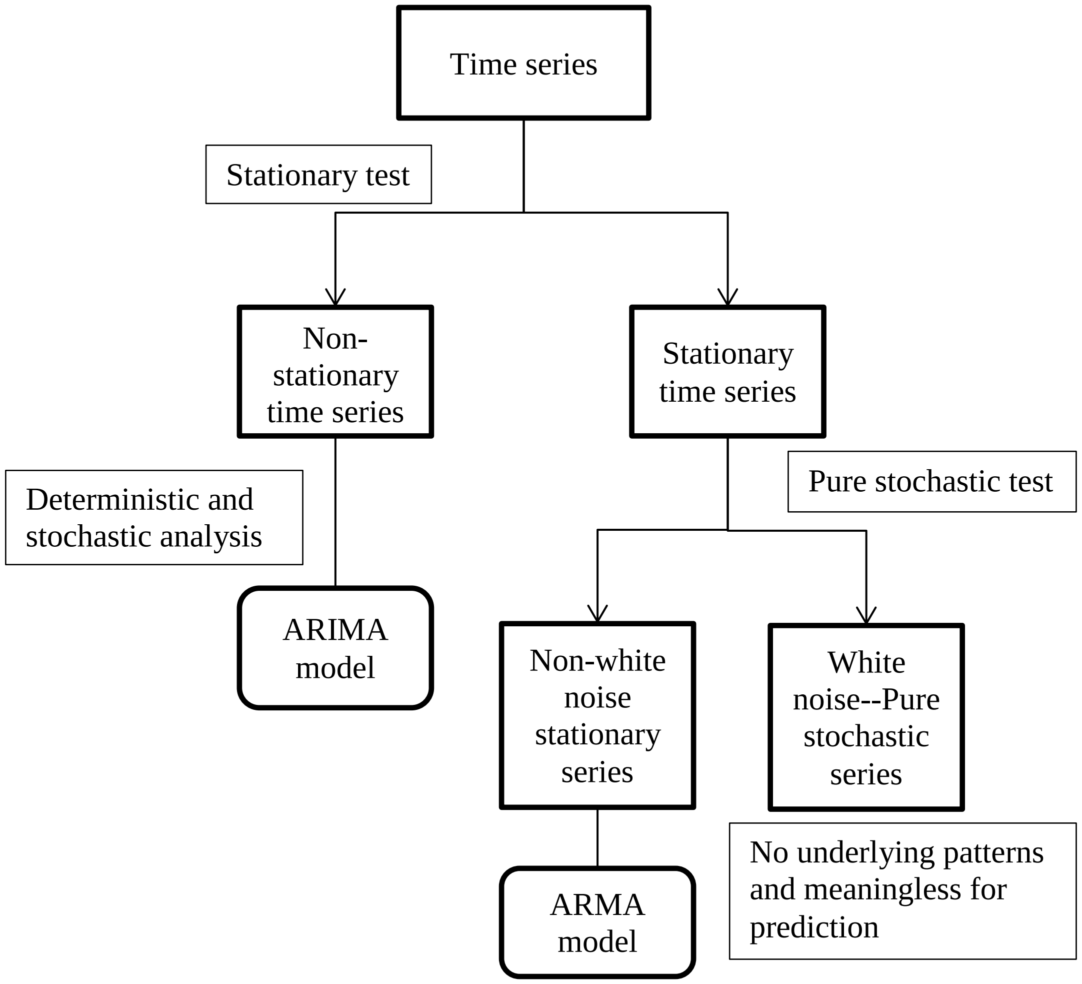

Time series
Stationary test
Non-stationary time series
Stationary time series
Deterministic and stochastic analysis
Pure stochastic test
ARIMA model
Non-white noise stationary series
White noise--Pure stochastic series
ARMA model
No underlying patterns and meaningless for prediction

Figure 2.1 Illustration of conventional procedure for time series analysis

This model-based procedure applies Box-Jenkins modeling approach [12] , such as Autoregressive–moving-average (ARMA) model and Autoregressive integrated moving average (ARIMA) [13], to analyze the behavior of a system. The focus of this approach is on a point-by-point “curve-fitting” strategy. However, not enough attention has been given to the explanation of dynamic relationships between the signal segments in the data sequences, e.g. temporal patterns, and critical occurrences of the events of interest. For reference, the basic structure of ARMA and ARIMA model is shown as:

1) The ARMA model

Given a real number time series data X_t where t is the time index, then the ARMA(p, q) model can be expressed as:

$$\left(1-\sum_{i=1}^{p}\phi_i L^i\right)X_t = \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t\,, \tag{1.6}$$

where

ε_t is the error term, which is generally assumed to be white noise series following the distribution $N(0,\sigma^2)$;

L denotes a lag operator such that $L^i X_t = X_{t-i}$ and $L^i\varepsilon_t = \varepsilon_{t-i}$;

ϕ_i and θ_i are parameters of the ARMA model, which can be estimated by a numerical optimization technique [14] or the Yule–Walker method [15].

In addition, the polynomial on the left side of the equal sign is the autoregressive model of order p, which can be denoted as AR(p); the right side is the moving average model of order q, denoted as MA(q).

2) The ARIMA model

The ARIMA model can be considered as an augmented model based on AMRA. As it for the non-stationary series, assume that the polynomial $\left(1-\sum_{i=1}^{p'}\phi_i L^i\right)$ has a unit root of multiplicity d [16]. Then it can be rewritten as:

$$\left(1-\sum_{i=1}^{p'}\phi_i L^i\right) = \left(1-\sum_{i=1}^{p'-d}\varphi_i L^i\right)\left(1-L\right)^d\,, \tag{1.7}$$

If we set $p = p'-d$, so an ARIMA(p,d,q) model is given by:

$$\left(1-\sum_{i=1}^{p}\varphi_i L^i\right)(1-L)^d X_t=\left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t, \tag{1.8}$$

As an innovated framework which is unlike the model-based methods, the approaches based on the RPS embedding for the DDS analysis is generally motivated by the presumption that the system dynamics can be explained in terms of a dependence between observations in the past and ones in the current or future time. The RPS approach is not focusing on building models for numerical analysis and prediction. Instead, it offers another perspective to discover the dynamics in the underlying system.

2.2 Review of the Contributive Research for Temporal Pattern Detection

In this section, several innovative methods and approaches for temporal pattern detection will be briefly reviewed in chronological order. Furthermore, the fundamental methodology applied in this thesis will be concluded based on the previous research.

2.2.1 Time Series Data Mining

In 1998, the framework based on RPS embedding for Time Series Data Mining (TSDM) was proposed by R. Povinelli and X. Feng [3]. A series of important concepts and definitions were introduced as the basic components for the research. In order to understand this method which is inherited by many successive studies, some of those basic components will be reviewed in the following discussion.

1. **Embedding dimension Q and time delay τ**

As it was shown in (1.4), these two parameters are important features for a RPS embedding process. The dimension Q of the RPS can be determined by using a false nearest-neighbors technique [9] [17]. For each embedding point $\mathbf{x}_i^Q$ in a Q-dimensional RPS, its nearest neighbor $\mathbf{x}_j^Q$ can be found by:

$$\mathbf{x}_j^Q = \underset{\mathbf{x}_j^Q, i \neq j}{\arg\min} \left\| \mathbf{x}_i^Q - \mathbf{x}_j^Q \right\|, \tag{1.9}$$

where $\left\| \mathbf{x}_j^Q - \mathbf{x}_i^Q \right\|$ is the Euclidean distance between the two points. If the embedding dimension is increased to $Q+1$, the change rate of the distance is measured by:

$$r_i = \sqrt{\frac{\left\| \mathbf{x}_j^{Q+1} - \mathbf{x}_i^{Q+1} \right\|^2 - \left\| \mathbf{x}_j^Q - \mathbf{x}_i^Q \right\|^2}{\left\| \mathbf{x}_j^Q - \mathbf{x}_i^Q \right\|^2}} \ , \tag{1.10}$$

If r_i exceeds a given threshold ρ, $\mathbf{x}_i^Q$ will be marked as having a false nearest neighbor. The criterion for an adequate value of Q is that the number of data points whose $r_i > \rho$ is zero, or small enough, in a Q-dimensional RPS.

The time delay τ can be calculated by finding the first zero point of the autocorrelation function [7]. The autocorrelation function was found to be effective in estimating the time delay. Meanwhile, it is an important reference for stationary test. Given a stochastic process X_t and time delay τ, the autocorrelation is:

$$R(\tau) = \frac{E\left[(X_t - \mu)(X_{t+\tau} - \mu) \right]}{\sigma^2} \tag{1.11}$$

where

μ and σ^2 are the mean and the variance of the observations of X_t respectively;

E denotes the expectation function.

2. Event and event function

In a certain DDS, an event is an important occurrence which reflects the state changing of the internal system. In [3], a Synthetic Seismic Time Series was used as an example to illustrate the events in a time sequence. The figure is shown as:

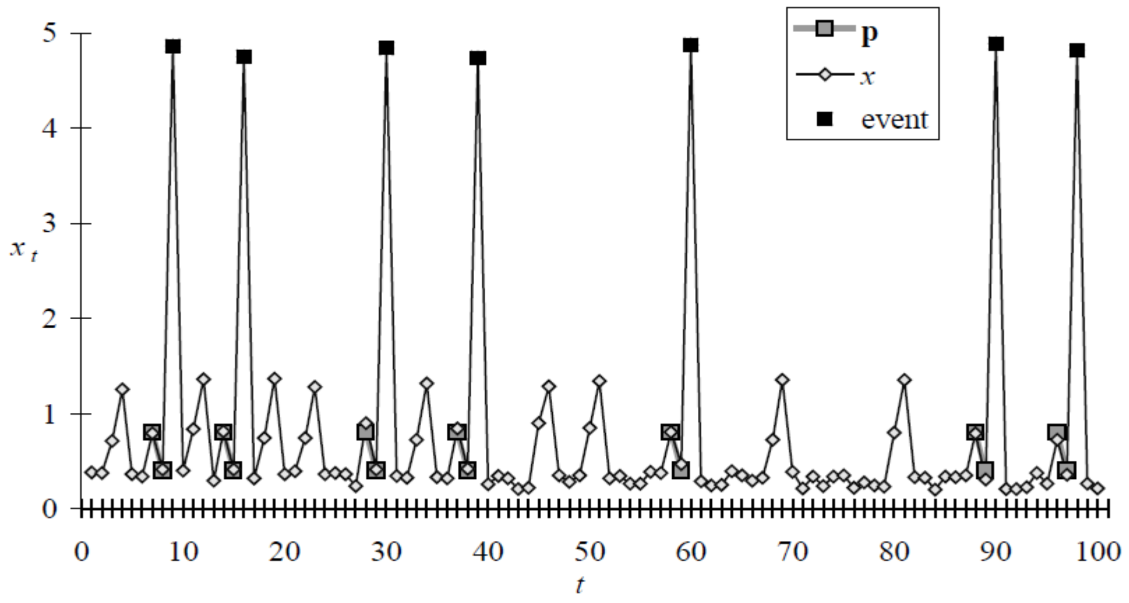


Figure 2.2 Synthetic Seismic Time Series with Temporal Pattern and Events

In order to discover the correspondence between temporal patterns and events, we can define an event function to characterize embedding vectors in the RPS and label each of them in different categories such as pattern and non-pattern. For example, if $\mathbf{x}_t = \left[x_{t-(Q-1)\tau} \quad \cdots \quad x_{t-\tau} \quad x_t \right]$ is an embedding vector, for a one-step-ahead prediction problem, the event function can be defined as:

$$g\left(\mathbf{x}_t\right) = x_{t+1}, \tag{1.12}$$

or a k-step-ahead event function:

$$g\left(\mathbf{x}_t\right) = \max\left\{x_{t+1}, \quad x_{t+2}, \quad \cdots \quad x_{t+k}\right\} \tag{1.13}$$

In some cases, the primary focus is on the percentage change rather than the actual values. An event function can be defined as the percentage change in the next step such as:

$$g(\mathbf{x}_t) = \frac{x_{t+1} - x_t}{x_t} \tag{1.14}$$

As a common method to categorize the embedding vectors, if it is predefined that the value of $g(\mathbf{x}_t)$ beyond a threshold c can be considered as an event, the embedding vector $\mathbf{x}_t$ will be labeled as a potential pattern to predict it. The labeled vectors will be used for pattern identification process with clustering algorithms.

3. **Objective function for pattern identification**

The problem of searching predictive temporal patterns in the RPS can be transformed into an optimization problem to maximize or minimize the objective function with respect to the underlying parameters, such as the center $\mathbf{v}$ and radius δ of the clusters. The objective function can be defined in different forms depending on the goal of pattern identification. In [3], several objective functions are presented for different tasks.

1) The Maximal event function

$$f(\mathbf{v},\delta)=\begin{cases}\mu_M & \|\mathbf{x}_i-\mathbf{v}\|<\delta, \mathit{if}\ M>\beta N \\ (\mu_M-g_0)\dfrac{M}{\beta N}+g_0 & \mathit{otherwise},\end{cases} \tag{1.15}$$

where

g_0 is the smallest event value in the cluster;

M is the number of data points in the cluster;

N denotes the number of total data points;

β is the rate of the minimum cluster size;

$\mu_M=\frac{1}{M}\sum_{i=1}^{M}g(\mathbf{x}_i)$ is the mean of the event function within the cluster.

2) Maximal statistical significance between data points in the cluster and outside cluster. This is simply based on the statistical t-test

$$f(\mathbf{v},\delta)=\frac{\mu_{M_c}-\mu_{\tilde{M}_c}}{\sqrt{\dfrac{\sigma^2_{M_c}}{C(M_c)}+\dfrac{\sigma^2_{\tilde{M}_c}}{C(\tilde{M}_c)}}}\ , \tag{1.16}$$

where

M_c represents the set of data points within cluster;

$\tilde{M}_c$ is the set of data points outside of the cluster;

$C(M_c)$ and $C(\tilde{M}_c)$ are the number of data points in these two sets;

μ_{M_c} and $\mu_{\tilde{M}_c}$ are the mean event function values of the two sets;

$\sigma^2_{M_c}$ and $\sigma^2_{\tilde{M}_c}$ are the variances of M_c and $\tilde{M}_c$ respectively.

3) Maximal event characterization accuracy

$$f(\mathbf{v},\delta) = \frac{tp+tn}{tp+tn+fp+fn}, \tag{1.17}$$

where

tp represents the number of true positives;

tn represents the number of true negatives;

fp represents the number of false positives;

fn represents the number of false negatives.

2.2.2 Fuzzy Set Based Clustering

This new method was proposed by Huang and Feng in 2005 [18]. It utilizes a fuzzy set based objective function with a Gaussian-shaped membership function to cluster temporal patterns in the time-delay embedding RPS. As it is proposed in [18], the fuzzy set objective function is defined as:

$$f(\mathbf{v},\delta) = \sum_{i=1}^{N} \exp\left(-\frac{\left\| \mathbf{v}-\mathbf{x}_i \right\|^2}{2\delta^2} \right) g(\mathbf{x}_i), \tag{1.18}$$

where

$\mathbf{x}_i$ is a vector embedded in RPS;

$\mathbf{v}$ represents the center of a fuzzy cluster;

δ represents the radius of the fuzzy cluster;

N denotes the number of total data points;

β is the rate of the minimum cluster size.

This objective function represents the summation of the weighted event functions. The weight tells how much an embedding vector contributes to the temporal pattern cluster. It takes the pattern density into consideration and will be robust to noisy points.

The objective function is continuously differentiable so the gradient descent optimization such as quasi-Newton methods can be applied to search the optimal clusters with a faster speed of convergence. The computational stability is significantly improved over the genetic algorithm which is originally used in the early developed TSDM mentioned above. The Broyden–Fletcher–Goldfarb–Shanno algorithm [19] [20] [21] [22] is applied as one efficient approach from the class of quasi-Newton methods to solve unconstrained nonlinear optimization problems. If $f(\mathbf{x})$ is an objective function to be minimized, meanwhile, from an initial guess $\mathbf{x}_0$ and an approximate Hessian matrix $\mathbf{B}_0 = x * \mathbf{I}$ (x is a scalar), the following steps are iterated as $\mathbf{x}_k$ converges to the solution:

1. Obtain a direction $\mathbf{p}_k$ by solving: $\mathbf{B}_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$ where ∇ is the gradient operator;
2. Perform a line search to find an acceptable step size α_k in the direction found in the first step, then update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$;
3. Set $\mathbf{s}_k = \alpha_k \mathbf{p}_k$;
4. $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$;
5. $\mathbf{B}_{k+1} = \mathbf{B}_k + \dfrac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \dfrac{\mathbf{B}_k \mathbf{s}_k (\mathbf{B}_k \mathbf{s}_k)^T}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k}$.

Convergence can be checked by observing if the norm of the gradient $\left|\nabla f(\mathbf{x}_k)\right|$ is less than a predefined threshold. The first step of the algorithm is carried out using the inverse of the matrix $\mathbf{B}_k$, which is usually obtained efficiently by applying the Sherman–Morrison formula [23] to the fifth line of the algorithm, which is

$$\mathbf{B}_{k+1}^{-1} = \mathbf{B}_k^{-1} + \frac{\left(\mathbf{s}_k^T \mathbf{y}_k + \mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k\right)\left(\mathbf{s}_k \mathbf{s}_k^T\right)}{\left(\mathbf{s}_k^T \mathbf{y}_k\right)^2} - \frac{\mathbf{B}_k^{-1} \mathbf{y}_k \mathbf{s}_k^T + \mathbf{s}_k \mathbf{y}_k^T \mathbf{B}_k^{-1}}{\mathbf{s}_k^T \mathbf{y}_k} \qquad (1.19)$$

2.2.3 The Existing MRPS-GMM Method

This approach was presented by W. Zhang and X. Feng in 2012 [6] to identify predictive temporal patterns in a multivariate dynamic data system. The new Multivariate Reconstructed Phase Space (MPRS) method is based on the multivariate RPS transformation, data categorization and nonlinear optimization.

One of the major limitations of the univariate RPS approach is that temporal patterns are typically assumed to exist only in the event sequence so it is impossible for the method to discover the correspondence between patterns and events lying in different, but correlated, sequences. This method can be widely applied on multivariate data sequences and can result in a better performance as well. For example, to monitor a patient’s cardiovascular conditions, besides measuring the electrocardiography (ECG) signals, the other measurements, such as blood pressure and body temperature, are also monitored constantly as the related factors to track the patient’s overall conditions. As an augmented method derived from univariate RPS embedding, the MRPS method is elaborated in the following discussion:

For instance, based on the equation (1.3), the observations of a DDS can be expressed as:

$$\mathbf{X}(t)=\begin{bmatrix} x_1(1) & x_1(2) & \cdots & x_1(N) \\ x_2(1) & x_2(2) & \cdots & x_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_m(1) & x_m(2) & \cdots & x_m(N) \\ x_e(1) & x_e(2) & \cdots & x_e(N) \end{bmatrix}_{(m+1)*N} \quad ,t=1\ldots N \tag{1.20}$$

The embedding vector at time t of the sequence on the j th row is:

$$\mathbf{x}_j(t)=\left[x_j(t-(Q_j-1)\tau_j) \quad \cdots \quad x_j(t-\tau_j) \quad x_j(t)\right], j=1\ldots m+1 \tag{1.21}$$

where Q_j and τ_j are the embedding dimension and time delay for the j th sequence respectively. In addition, when $j=m+1$, it means the embedding sequence is the target sequence in the observations. So the embedding vector at time t in the multivariate RPS can be constructed as:

$$\mathbf{x}(\mathrm{t})=\left[\mathbf{x}_1(\mathrm{t}) \quad \mathbf{x}_2(\mathrm{t}) \quad \ldots \quad \mathbf{x}_m(\mathrm{t}) \quad \mathbf{x}_e(\mathrm{t})\right] \tag{1.22}$$

Apparently, the dimension Q for the multivariate embedding is the summation of all embedding dimension Q_j, which means $Q=\sum_{j=1}^{m+1}Q_j$.

Another contribution of this method is the concept of pattern similarity. It was proposed to solve the problem that vectors which should be categorized as the same type of temporal patterns may fall into different regions when embedding a data sequence with an outstanding trend into the RPS. Furthermore, inspired by the fuzzy-set method mentioned above, Gaussian mixture model (GMM) is applied to describe the distributions of the data points in different categories with a sloped boundary instead of a crisp one. The detailed applications for these two methods will be discussed in Chapter 3.

2.3 Summary of the Research Developments

From the reviews in the above sections, the relationship of inheritance and extension between different approaches is an important factor that keeps improving the methodology of temporal pattern detection theoretically and practically. The study of TSDM proposed the original problem statement and established the framework for the future study. The Fuzzy-set based method offers an innovative idea which is different from the conventional algorithms that search clusters with crisp boundary. This method inspires the study of applying GMM to define the objective function. The newly developed method based on GMM and multivariate RPS embedding greatly extends the applicable scope. It is possible to detect patterns in multiple related sequences simultaneously. Besides, there are many approaches of temporal pattern detection have been proposed based on other mathematical models and algorithms, such as the logistic regression based approach by Feng, Senyana [24], the Gaussian mixture model-Support Vector Machine (GMM-SVM) hybrid approach by Zhang, Feng and Bansal [25] and the Hidden Markov Model (HMM) based approach by Bansal, Feng, Zhang et al [26].

CHAPTER 3 AN IMPROVED MRPS-GMM APPROACH

Gaussian Mixture Model (GMM) is a probabilistic model which can be used to represent the presence of subpopulations within an overall population by assuming that the distribution for the overall population can be described as a mixture of multiple Gaussian distributions. In a recent research of temporal pattern detection, a GMM based approach has been proposed by W. Zhang and X. Feng in [6], which enhance the performance of the classifier by an innovative method. However, there is still work to do to improve it. In this chapter, we will review the background of GMM and EM algorithm at first. Then we will discuss the disadvantage of the existing GMM approach and present the newly improved method. Meanwhile, we will test the performance of the improved method on a series of simulated data.

3.1 Background of GMM

A Gaussian Mixture Model (GMM) can be defined as a parametric probability density function which is represented as a weighted sum of Gaussian-distributed component densities. GMM parameters can be estimated from the training data using the iterative Expectation-Maximization (EM) algorithm. In the section, we only discuss the GMM based on multivariate normal distribution which is more applicable to fit probability density functions in a multidimensional phase space.

The multivariate normal distribution of a Q-dimensional random vector $\mathbf{x} = \left[\mathrm{X}_1 \ \mathrm{X}_2 \ \dots \mathrm{X}_Q \right]$ can be written as:

$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) , \tag{1.23}$$

where

$\boldsymbol{\mu}$ is a Q-dimensional mean vector;

$\boldsymbol{\Sigma}$ is an $m \times m$ positive definite covariance matrix.

The value of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be calculated by

$$\boldsymbol{\mu} = \left[\mathrm{E}(\mathrm{X}_1)\ \mathrm{E}(\mathrm{X}_2) \ldots \mathrm{E}(\mathrm{X}_Q) \right], \tag{1.24}$$

$$\boldsymbol{\Sigma} = \left[\Sigma_{i,j} \right] = \left[\mathrm{Cov}\left(\mathrm{X}_i, \mathrm{X}_j \right) \right], i = 1, 2, \ldots, Q;\ j = 1, 2, \ldots, Q, \tag{1.25}$$

The probability density function is defined as:

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^Q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}) \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})^T \right), \tag{1.26}$$

where $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix [27]. In order to guarantee that the exponential part in (3.4) is multipliable, $\mathbf{x}$ and $\boldsymbol{\mu}$ must be row vectors. As long as both of the mean vector and covariance matrix are known, it will be able to fit a probability density function for the data. The following picture is showing a 2-dimentional normal distribution with the predefined mean vector $\boldsymbol{\mu} = [0\ 0]$ and covariance matrix

$\boldsymbol{\Sigma} = \begin{bmatrix} 1.5 & 0.3 \\ 0.3 & 0.25 \end{bmatrix}$:

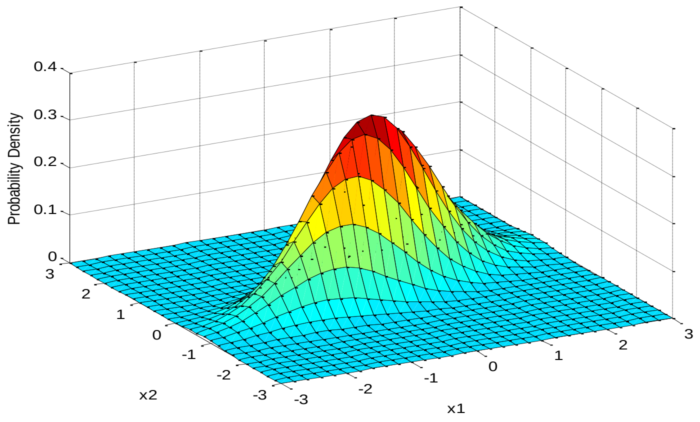


Figure 3.1 Example of a 2-dimentional normal distribution

However, in some cases, it will be more sufficiently accurate to describe the probabilistic distribution of the observed data by the GMM instead of using just one Gaussian probability density function. For example, we can apply a 2-component GMM to fit the probability density function of the 2-dimesional data shown as:

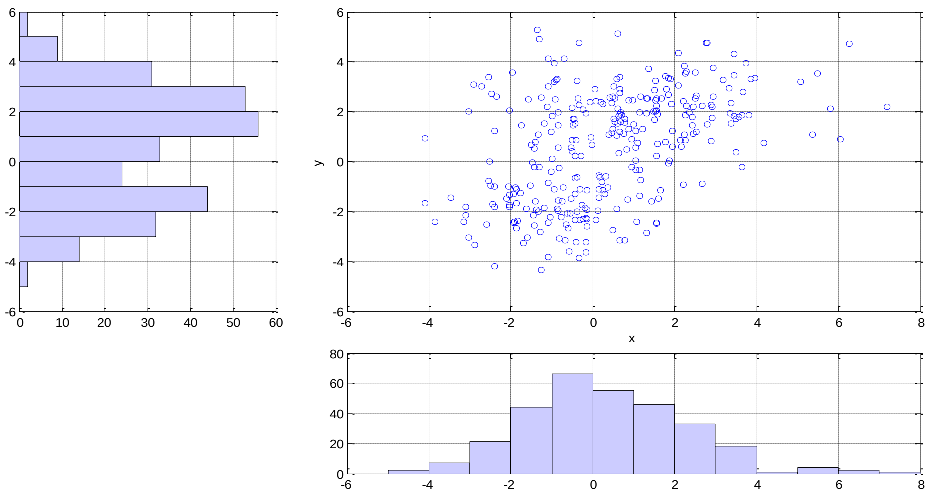


Figure 3.2 The scatter-histogram plot of a set of 2-dimensional data

The figure shows the scatter plot and margin distributions of 300 observations of a simulated dataset which can be generated by the following commands in MATLAB:

```
m1 = [1 2];
s1 = [3 0; 0 2];
m2 = [-1 -2];
s2 = [2 0; 0 1];
X = [mvnrnd(m1,s1,200);mvnrnd(m2,s2,100)];
```

The 2-component GMM to fit the distribution for the dataset is shown as:

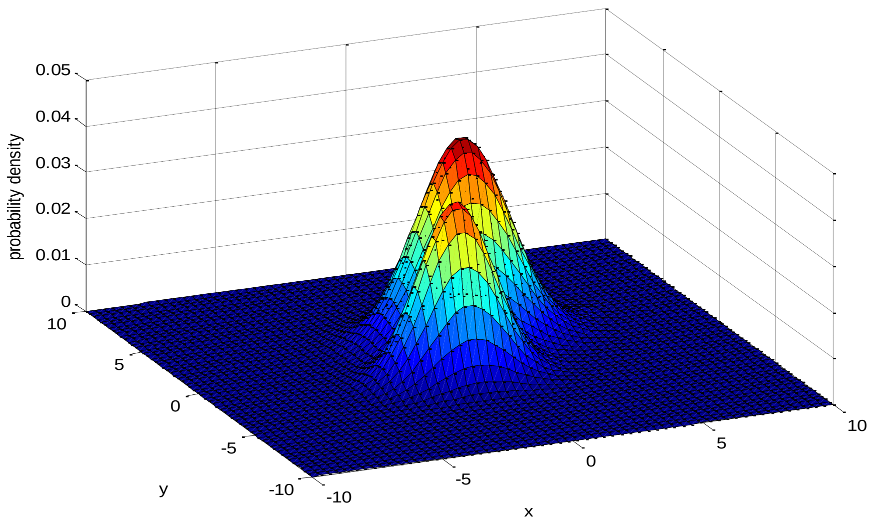


Figure 3.3 Distribution of the dataset by a 2-component GMM

Also, we can use contour map to display the GMM in a 2-D plot:

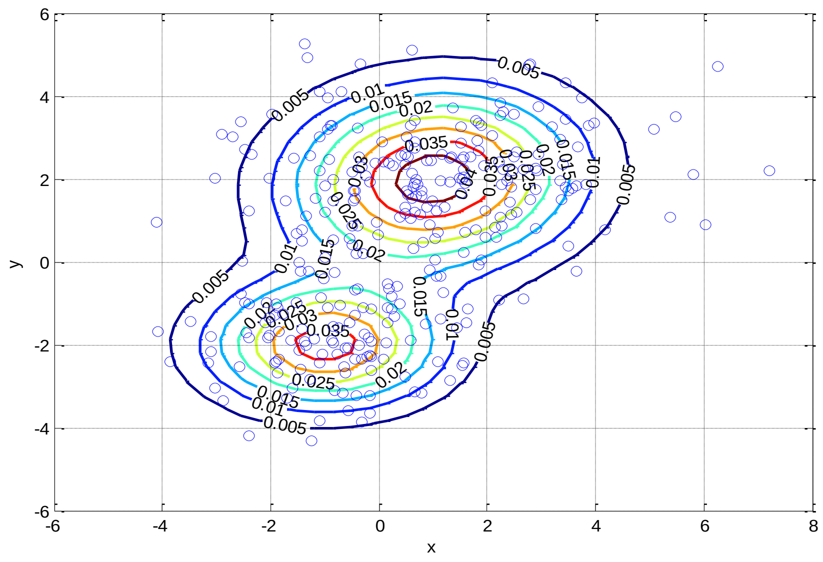


Figure 3.4 Contours for the 2-component mixture distribution

As we can see from the figures above, the two Gaussian distributed components have different mean vectors, covariance matrices and mixing proportions. These three parameters are fundamental to the GMM and can be estimated by the EM algorithm.

3.2 Basic Theories of the EM Algorithm

The Expectation–Maximization (EM) algorithm [28] is an iterative algorithm to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, when the observations can be viewed as incomplete data [29].

The EM algorithm iteratively performs two steps: expectation (E) step and maximization (M) step. In E step, it creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters. In M step, it computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step unless the results satisfy the condition to terminate.

The EM algorithm has been widely used in many statistical problems. To make it simple to understand, we will only discuss its procedure to estimate the parameters of a K-component GMM in this section.

Assume that $\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_i \cdots \mathbf{x}_N \in \mathbb{R}^Q$ are N observations of a dataset and each one is a Q dimensional vector. In order to build a K-component GMM to describe the distribution of the dataset, first we need to initialize the mean vector $\boldsymbol{\mu}_j$, covariance matrix $\boldsymbol{\Sigma}_j$ and mixing proportion π_j for each component. Especially, it is necessary to emphasize that the sum of the mixing proportion of each component is one, which means:

$$\sum_{j=1}^{K} \pi_j = 1 \ , \tag{1.27}$$

If we define $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\mu}_1 & \boldsymbol{\mu}_2 & \cdots & \boldsymbol{\mu}_j & \cdots & \boldsymbol{\mu}_K \\ \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_2 & \cdots & \boldsymbol{\Sigma}_j & \cdots & \boldsymbol{\Sigma}_K \\ \pi_1 & \pi_2 & \cdots & \pi_j & \cdots & \pi_K \end{bmatrix}$ to represent the set of all current parameters, we can evaluate the log-likelihood function as:

$$\ln L(\boldsymbol{\theta} \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \ln\, p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \mid \boldsymbol{\theta}) \, , \tag{1.28}$$

If the observations are mutually independent, the log-likelihood can be written in:

$$\ln L(\boldsymbol{\theta} \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \sum_{i=1}^{N} \ln\, p(\mathbf{x}_i \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \ln(\sum_{j=1}^{K} \pi_j\, p(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) \, , \tag{1.29}$$

where

L denotes the likelihood function;

p, in this case, denotes the probability density function of multivariate normal distribution , as it shown in (3.4), given the current parameters in $\boldsymbol{\theta}$.

The iteration will begin in the E step with the initialized parameters. One of the most important variables in the algorithm is the probability that the ith observation belongs to the jth Gaussian distributed component of the GMM. It can be computed by the following expression:

$$\gamma_{i,j} = \frac{\pi_j p(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{j=1}^{K} \pi_j p(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \ , \tag{1.30}$$

This is a membership variable called “responsibility” which can be seen as an assignment score to indicate how much the jth Gaussian component is responsible for $\mathbf{x}_i$.

In M step, the current membership variables can be used to estimate $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$ and π_j for each component. The new estimations can be iteratively computed as:

$$\pi_j^{new} = \frac{1}{N} \sum_{i=1}^{N} \gamma_{i,j} \ , \tag{1.31}$$

$$\boldsymbol{\mu}_j^{new} = \frac{\sum_{i=1}^{N} \gamma_{i,j} \mathbf{x}_i}{\sum_{i=1}^{N} \gamma_{i,\mathrm{j}}} \ , \tag{1.32}$$

$$\boldsymbol{\Sigma}_j^{new} = \frac{\sum_{i=1}^{N} \gamma_{j,i} (\mathbf{x}_i - \boldsymbol{\mu}_j)^T (\mathbf{x}_i - \boldsymbol{\mu}_j)}{\sum_{i=1}^{N} \gamma_{j,i}} \ , \tag{1.33}$$

Still, the observations and mean vectors must be row vectors so that the dimensions of the results are matched on both sides of the above equations. With the newly estimated parameters, the new log-likelihood function can be computed as:

$$\ln L(\boldsymbol{\theta}^{new} \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \sum_{i=1}^{N} \ln(\sum_{j=1}^{K} \pi_j^{new} p(\mathbf{x}_i \mid \boldsymbol{\mu}_j^{new}, \boldsymbol{\Sigma}_j^{new})), \tag{1.34}$$

Then, we can compare the new log-likelihood with the previous one computed in (3.7) to see if the increment is less than a preset threshold ε. If so, the iteration can be terminated. Otherwise, we can substitute the current $\boldsymbol{\theta}$ by the newly estimated $\boldsymbol{\theta}^{new}$ to start over the whole process from the E step until the log-likelihood is convergent.

The iteration can be illustrated by the following flow chart:

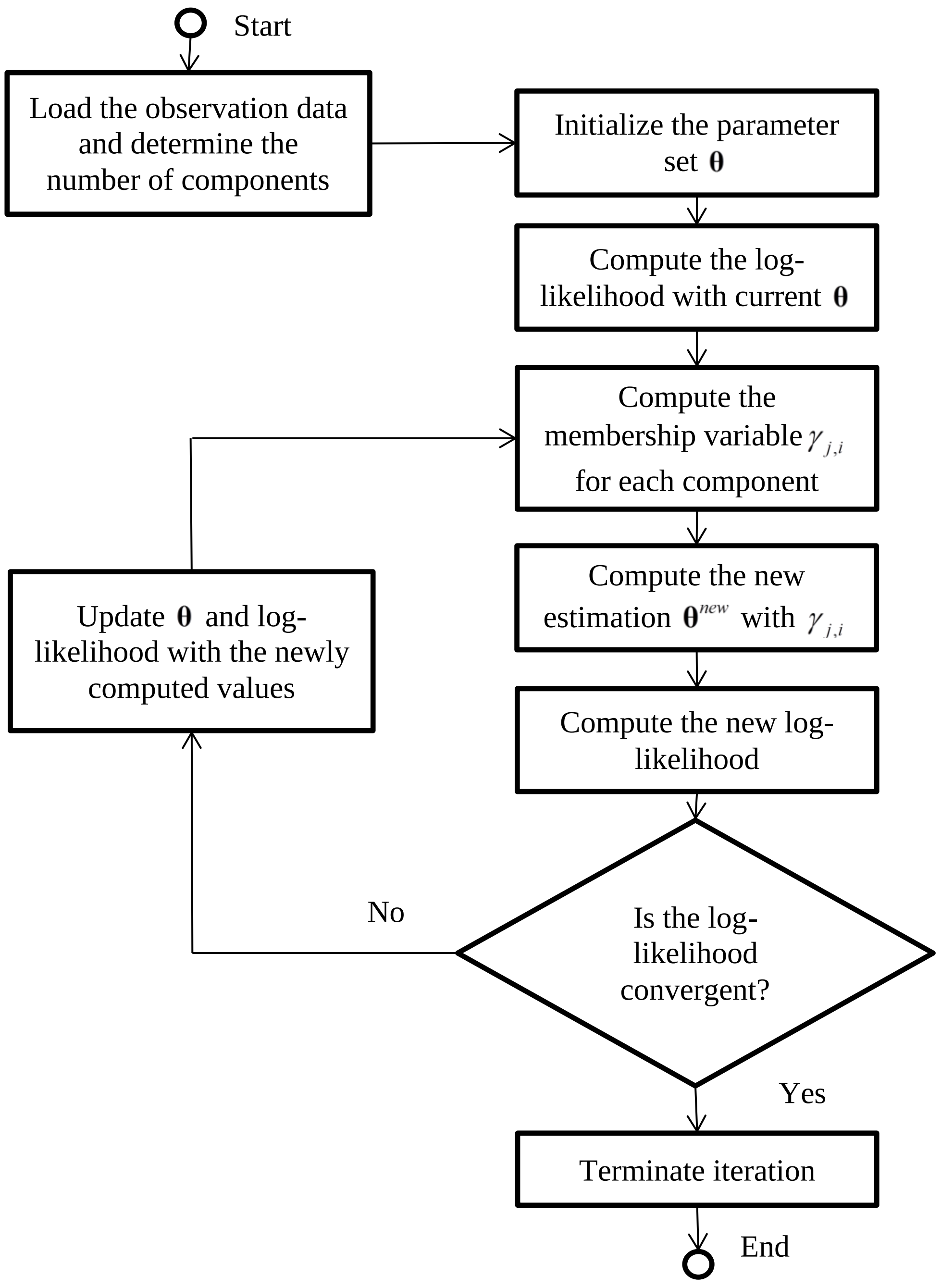

Start
Load the observation data and determine the number of components
Initialize the parameter set $\boldsymbol{\theta}$
Compute the log-likelihood with current $\boldsymbol{\theta}$
Compute the membership variable $\gamma_{j,i}$ for each component
Compute the new estimation $\boldsymbol{\theta}^{new}$ with $\gamma_{j,i}$
Compute the new log-likelihood
Is the log-likelihood convergent?
No
Update $\boldsymbol{\theta}$ and log-likelihood with the newly computed values
Yes
Terminate iteration
End

Figure 3.5 Flow chart of the EM algorithm for GMM

3.3 Proposed Improvements to the Existing MRPS-GMM Approach

GMM based approach for signal classification has been introduced by R. J. Povinelli, M. T. Johnson et al. in [30], which inspires the original idea that applying GMM for temporal pattern detection. Unlike the unsupervised learning approach in [30] which is focusing on clustering, the MRPS-GMM hybrid method proposed by Zhang and Feng in [6] separated the embedding vectors into three states--pattern, normal and event--by the event function. The multivariate GMM is utilized to fit the probability distribution by applying the training dataset to gain the initial values of the mean vector, covariance matrix and mixing proportion for each state respectively to ensure convergence. Then we can acquire the log-odds of an embedding vector $\mathbf{x}_t$ as following which can be used in a classifier to categorize $\mathbf{x}_t$ into different states:

$$\varphi(\mathbf{x}_t) = \log \frac{p(\omega_p \mid \mathbf{x}_t)}{p(\omega_n \mid \mathbf{x}_t)}, \tag{1.35}$$

where ω_p and ω_n denote the states of pattern and normal [6]. As a conclusion and modification, the detailed procedure of the temporal pattern detection subject applied in this thesis is shown as the following figure:

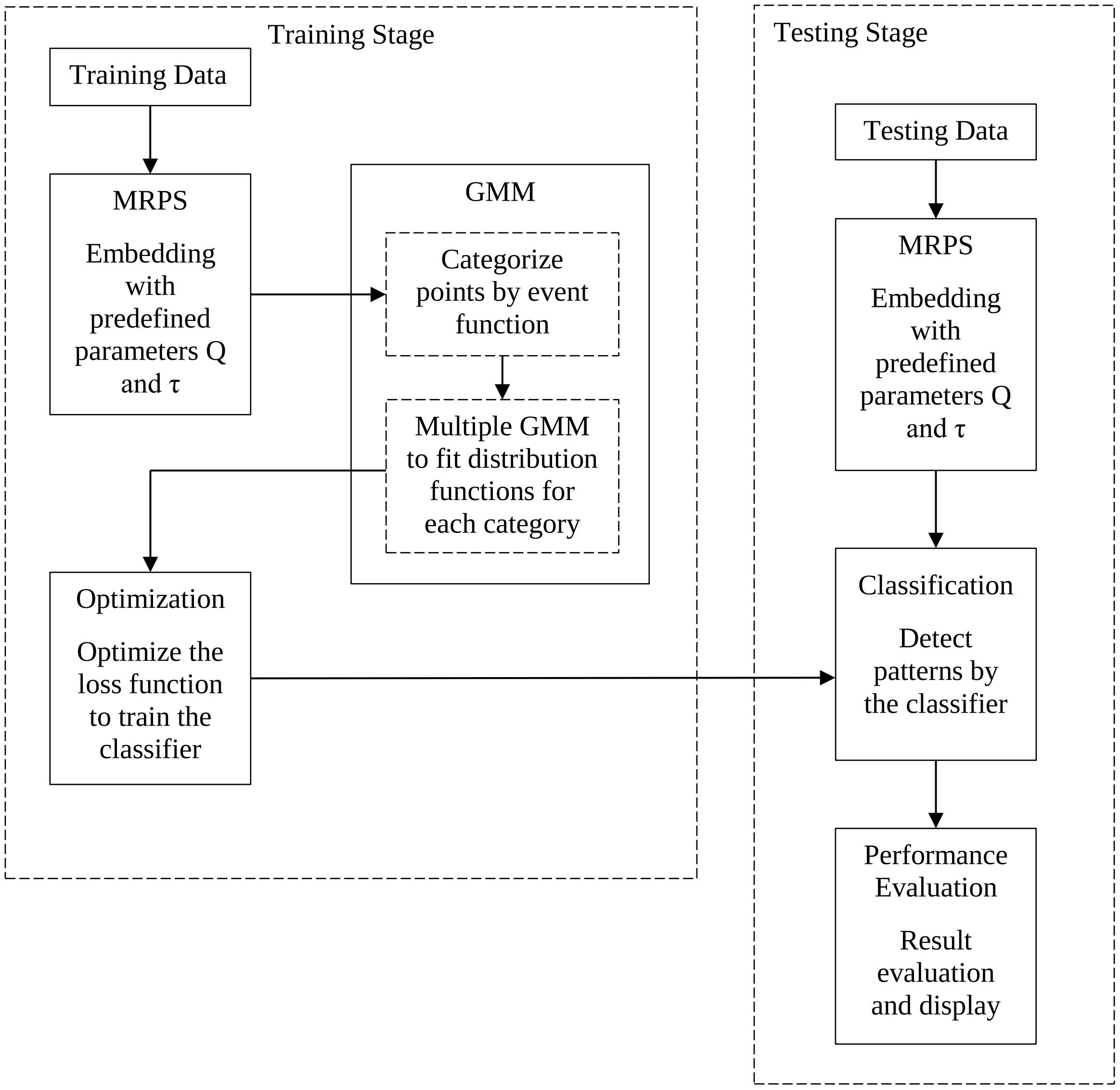

Training Stage
Training Data
MRPS
Embedding with predefined parameters Q and τ
GMM
Categorize points by event function
Multiple GMM to fit distribution functions for each category
Optimization
Optimize the loss function to train the classifier
Testing Stage
Testing Data
MRPS
Embedding with predefined parameters Q and τ
Classification
Detect patterns by the classifier
Performance Evaluation
Result evaluation and display

Figure 3.6 Framework for temporal pattern detection based on the previous research

Although it is efficient for classifying, this method is still not sufficiently accurate to handle the diversity of shapes among the embedding vectors. For example, we can create a simulated dataset by the following commands in MATLAB:

```
mu1 = [1 2];
sigma1 = [3 .2; .2 2];
mu2 = [-1 -2];
sigma2 = [0.5 0; 0 0.6];
mu3 = [4 -1];
sigma3 = [0.6 0.3; 0.3 0.4];
X1 = [mvnrnd(mu1,sigma1,200)];
```

```
figure
scatter(X1(:,1),X1(:,2))
hold on
X2 = [mvnrnd(mu2,sigma2,50);mvnrnd(mu3,sigma3,50)];
scatter(X2(:,1),X2(:,2))
hold on
legend('normal','pattern');
```

This dataset consists of a set of vectors embedded in a 2-D RPS illustrated as:

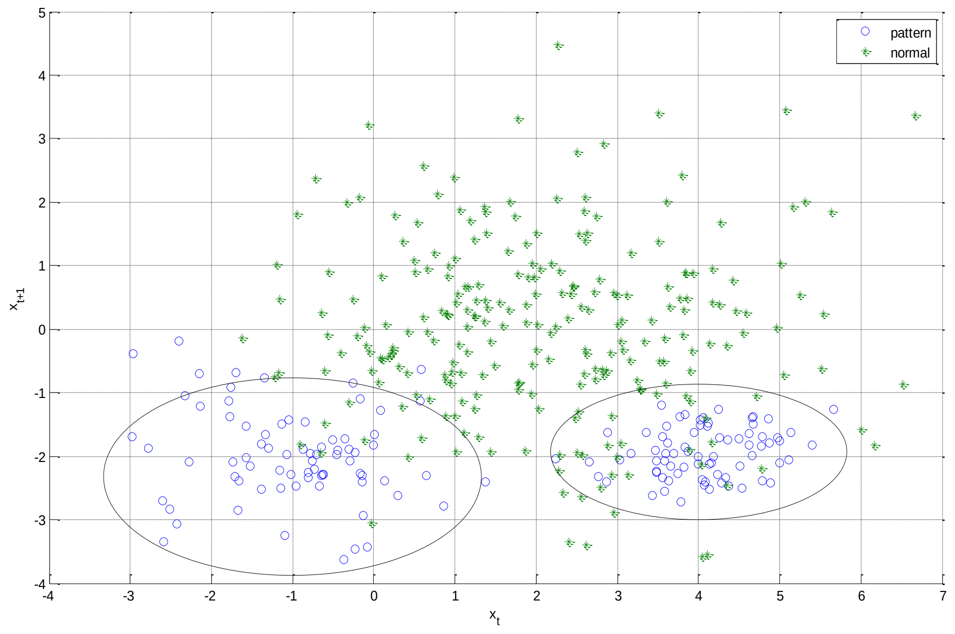


Figure 3.7 Embedding vectors in a 2-D RPS

There are 200 normal vectors and 100 pattern vectors in the RPS. Apparently, the pattern vectors are gathering into two separated regions. In this case, the existing approach will cause a large amount of misclassifications by fitting the distribution for pattern vectors with a single Gaussian distribution function. This misclassification issue is illustrated by the following figure:

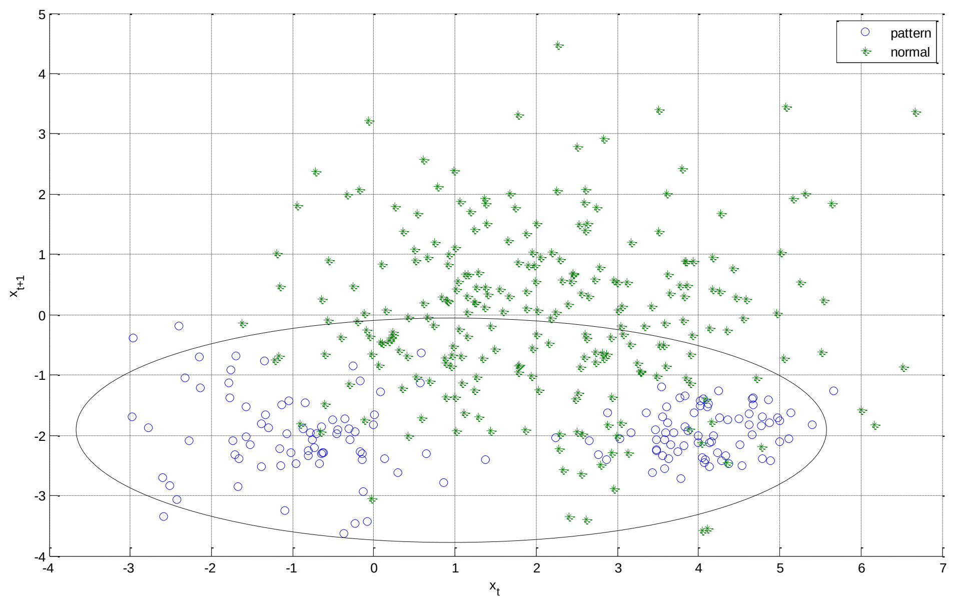


Figure 3.8 Misclassification issue caused by single Gaussian distribution function

To overcome this obstacle, we can apply the improved GMM approach. Before we start the process, the vectors standing for the event states have already been eliminated from the RPS. The reason is simple: because the embedding vectors belonging to event state are signal segments with the appearance of events, which means this state won't be helpful to predict the event in testing data. The improvements can be concluded in the following two aspects:

1. Unlike the existing method which acquires the initial parameters' values from the training data, the improved method applies a specific unsupervised learning algorithm that randomly samples the initial values from the dataset according to the predefined number of components in pattern and normal states. This algorithm allows users to try different numbers of Gaussian distributed components for a specific problem to see which one will make the

log-likelihood converge in the least iterations. Moreover, this algorithm can initially assign a data point into one of the components by finding out which mean vector is closer to the point. It can prevent the circumstance that multiple components converge to the same area from happening.

2. Instead of using loop statement, the algorithm of the improved method applies matrix multiplication to rapidly compute the estimations for each parameter in the iteration process. In MATLAB development environment, the computing speed of matrix multiplication is much faster than using loop statement. With this convenient feature, it will be more efficient to compute the polynomial summations in each step of iteration and finish the whole process with less running time. Besides, the algorithm will have a better performance for classification and optimization process by using matrix multiplication instead of loop statement.

We can explain the detailed procedure of the improved GMM approach by the following block diagram:

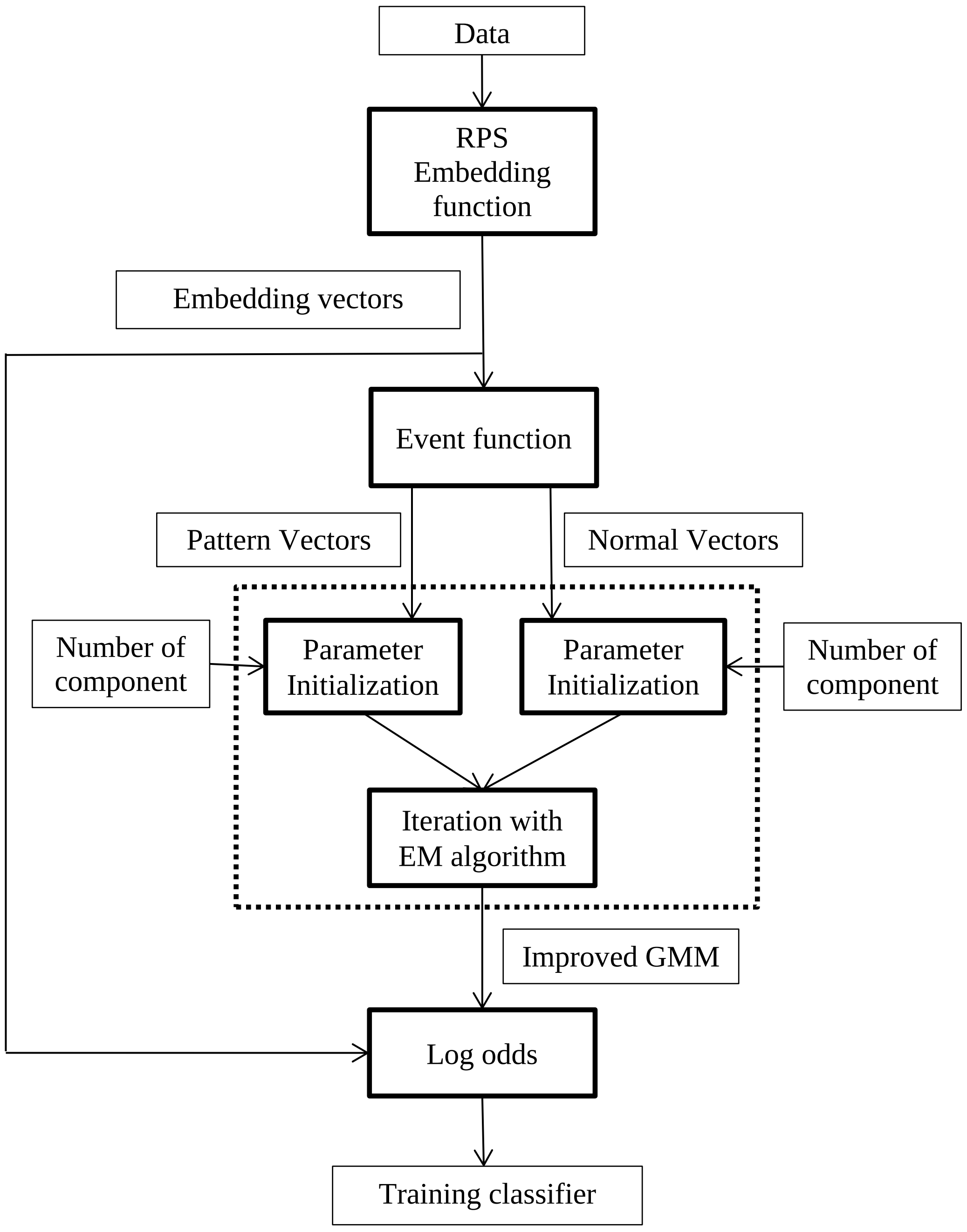


Figure 3.9 Block diagram of the improved GMM approach

The following figure shows the distribution of the dataset on the previous figure created by the improved GMM approach. Obviously, the probability distribution can precisely reflect that pattern points gather into separated regions in the space.

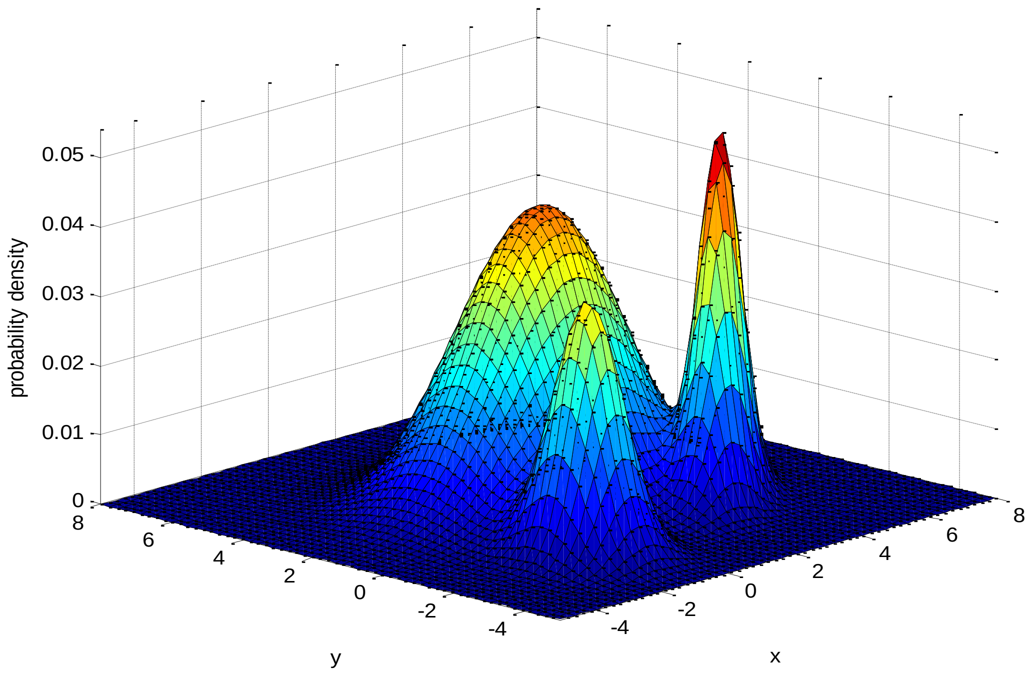


Figure 3.10 The probability distribution with the improved GMM approach

In addition, based on the concept of "pattern similarity" proposed by Zhang in [31], we can build a simplified and efficient linear classifier as:

$$f(\mathbf{x}_t) = \beta_1 \sum_{i=1}^{N_p} \exp\left(-\frac{\left\| \phi(\mathbf{x}_t) - \phi(\mathbf{x}_i^p) \right\|^2}{\sigma^2} \right) + \beta_2 \varphi(\mathbf{x}_t) + \beta_3, \tag{1.36}$$

where

N_p is the total number of vectors in pattern state ω_p;

$\mathbf{x}_i^p$ is the ith vector in pattern state;

σ is a predefined parameter for similarity measure;

$\varphi(\mathbf{x}_t)$ is the log odds denoting the Gaussian mixture log likelihood score of $\mathbf{x}_t$;

$\phi(.)$ is a mapping from Q-dimensional space to $(Q-1)$-dimensional space.

Actually, it has been proved in [31] that if we denote the similarity of two different Q-dimensional vectors as $d(\mathbf{x}_1,\mathbf{x}_2)$, there exists a mapping $\phi(.):\mathbb{R}^Q\to\mathbb{R}^{Q-1}$, such that $d(\mathbf{x}_1,\mathbf{x}_2)=\|\phi(\mathbf{x}_1)-\phi(\mathbf{x}_2)\|^2$. By a series of mathematical derivation, $\phi(.)$ can be given as:

$$\phi(.)=\mathbf{L}\nabla(.) \tag{1.37}$$

where

$\mathbf{L}$ is the Cholesky decomposition of matrix $\mathbf{P}=\begin{bmatrix} Q-1 & Q-2 & \cdots & 1 \\ Q-2 & Q-2 & \cdots & 1 \\ \vdots & \vdots & \ddots & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ such that

$\mathbf{P}=\mathbf{L}^T\mathbf{L}$;

∇ is the backward difference operator.

For the optimization process, we can apply the exponential loss function proposed in [31], which has been proved to be robust and stable. The function is shown as:

$$\min_{\boldsymbol{\beta}}\left\{L(g(.),f(.))\right\}=\min_{\boldsymbol{\beta}}\sum_{t=1}^{N}\exp(-g(\mathbf{x}_t)f(\mathbf{x}_t))+\frac{\eta}{2}\|\boldsymbol{\beta}\|^2, \tag{1.38}$$

where

N is the total number of vectors in the RPS;

η is the penalty coefficient;

$\boldsymbol{\beta}=\begin{bmatrix}\beta_1 & \beta_2 & \beta_3\end{bmatrix}$ in $f(\mathbf{x}_t)$.

3.4 Experiments with Simulated Data

In this section, we will apply the improved approach to predict the event occurrences in a set of simulated data sequences:

In the first example, we will artificially create several data observations by inserting two different types of patterns in the pattern sequence on purpose to test if the improved approach is able to detect them.

The second example will apply Henon map, which is a dynamical discrete-time chaotic system, to test the performance on a data system which is strictly defined according to its state equations. In addition, it will make a comparison between the existing MRPS-GMM approach and the improved one to show that the latter has a better performance.

Example 1: Detect two types of temporal patterns in white noise series

The following figure shows part of the data applied in this example:

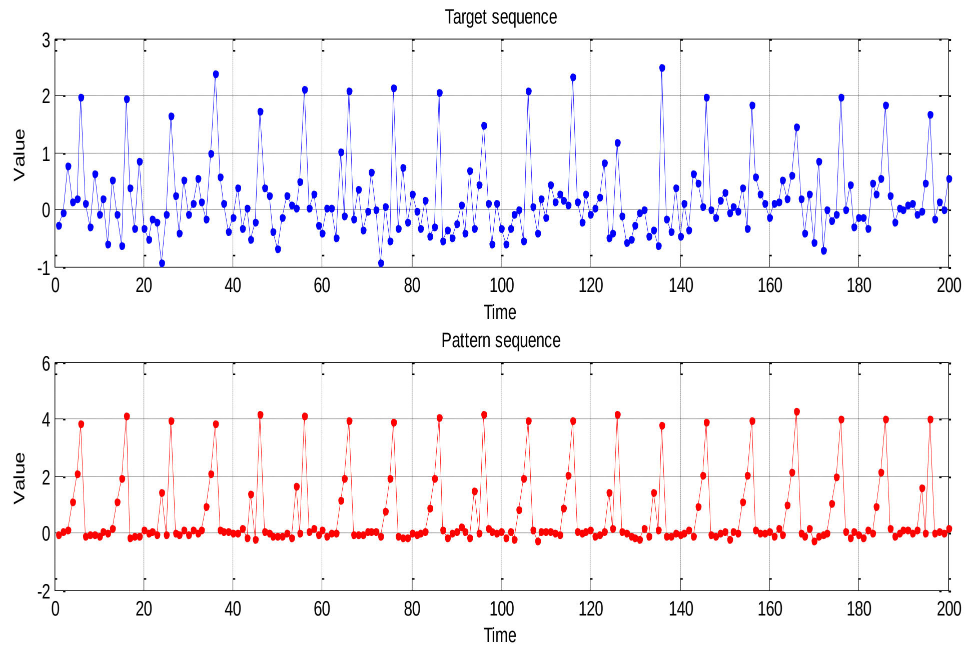


Figure 3.11 The artificial data with created patterns (200 data points)

This simulated data is generated by the following commands:

```
x = 0.4*randn(200,1);
y = 0.1*randn(200,1);

for i=1:20
    x(6+10*(i-1)) = x(6+10*(i-1))+2;
    x(6+10*(i-1)-2) = 0.1+0.4*randn(1);
end
target_sequence = x;

for i=1:20
    y(6+10*(i-1)) = y(6+10*(i-1))+4;
    y(6+10*(i-1)-2) = 1.5+0.1*randn(1);
end
rs = randsample(20,13);
for i=1:13
    y(10*(rs(i)-1)+6-1) = 2+0.1*randn(1);
    y(10*(rs(i)-1)+6-2) = 1+0.1*randn(1);
end
pattern_sequence =y;
```

Both target and pattern sequences are synchronized and correlated by time. The target sequence is interfered by a high power white noise series. In result, it will be difficult to detect the patterns directly. Oppositely, the noise interference in the pattern sequence is less than 10%, which is more acceptable. Also, we have inserted two different types of 3-dimensional temporal patterns in the sequence for 1 step-ahead prediction, which means each pattern can predict the occurrence of an event after $k=1$ time-point. These patterns can be embedded in a 3D RPS as:

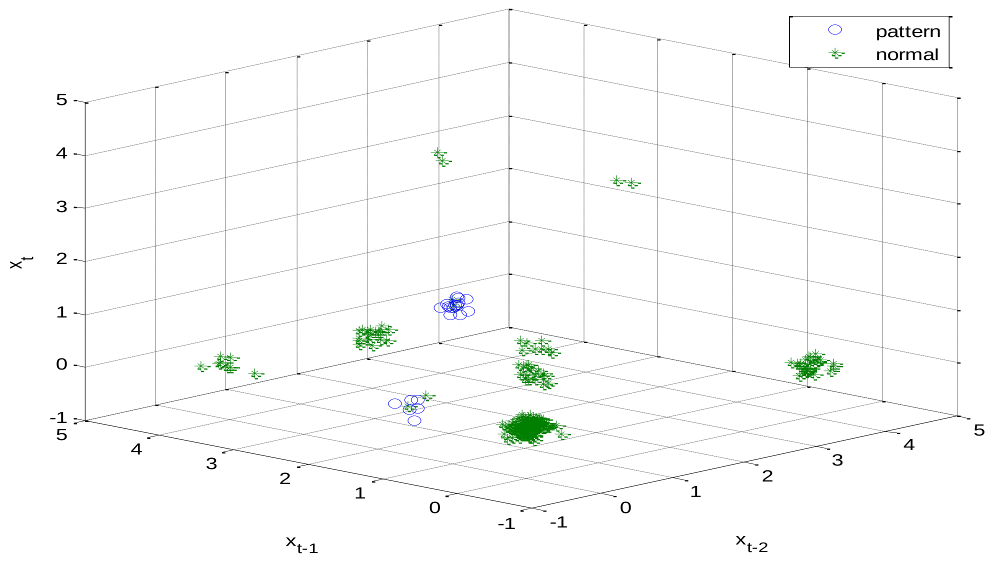


Figure 3.12 Embedding vectors in a 3D RPS

However, the event we want to predict is in the target sequence, which is the point whose value is above 1.5. In this situation, to apply multivariate embedding method, we can denote the combined embedding vector $\mathbf{x}_t$ as:

$$\mathbf{x}_t = \left[x_{t-(Q_x-1)\tau_x} \quad \dots \quad x_{t-\tau_x} \quad x_t \quad y_{t-(Q_y-1)\tau_y} \quad \dots \quad y_{t-\tau_y} \quad y_t \right], \tag{1.39}$$

where x_t and y_t are observations at time t in target sequence and pattern sequence respectively. Also, we can predefine the embedding dimension and time delay for both sequences: $Q_x = 1, Q_y = 3, \tau_x = \tau_y = 1$.

Therefore, the event characterization function can be defined as:

$$g(\mathbf{x}_t) = \begin{cases} 1, & x_{t+1} > c \\ -1, & x_{t+1} <= c \end{cases}, \tag{1.40}$$

where $c = 1.5$ is the event threshold.

We can simply set the number of components in the GMMs for pattern state (NC_p) and normal state (NC_n) to 2 and 1respectively. Technically, in this case, the distribution of the normal state is not a GMM. However, we can still treat single Gaussian distribution as a special case of GMM. The following table shows all the parameters that need to be initialized:

Process	Parameters
MRPS	$Q_x = 1, \tau_x = 1, Q_y = 3, \tau_y = 1, k = 1$
GMM	$c = 1.5, NC_p = 2, NC_n = 1$
Optimization	$\sigma = 0.5, \eta = 1, \boldsymbol{\beta} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}$

Table 3.1 List of the initialized parameters for Example 1

Generally, it might take a long time for the optimization process to find optimal values of σ and η because of the computational amount. Besides, the initial value of $\boldsymbol{\beta}$ also plays an important role in the optimization process because quasi-Newton method, the optimization method applied in this thesis, converges to local minima, which means this method depends on the initial values significantly. So the initial values for the optimization process are reference values in both examples. They only ensure the result's accuracy is acceptable, not optimal.

There are 400 observations in both sequences. We apply the first 250 data observations for training and use the remaining 150 observations for testing. The following figure shows the validation result on training data:

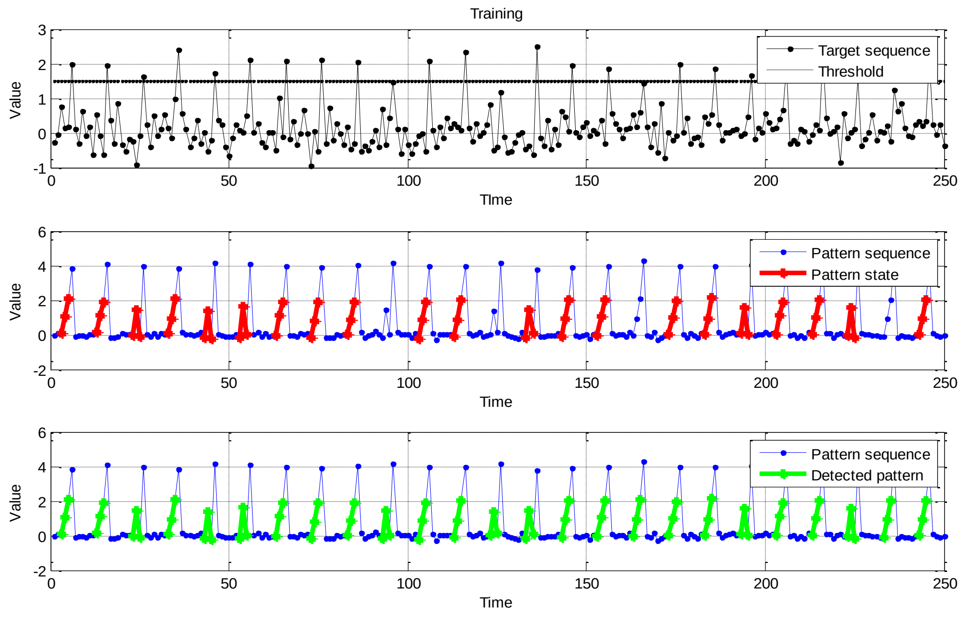


Figure 3.13 Validation result with training data

Since the patterns are inserted on purpose, the over-fitting problem can be ignored in this example. The red marks for “Pattern state” denote the data segments are classified in the pattern state by the event function.

The following figure shows the testing result with the remaining data observations:

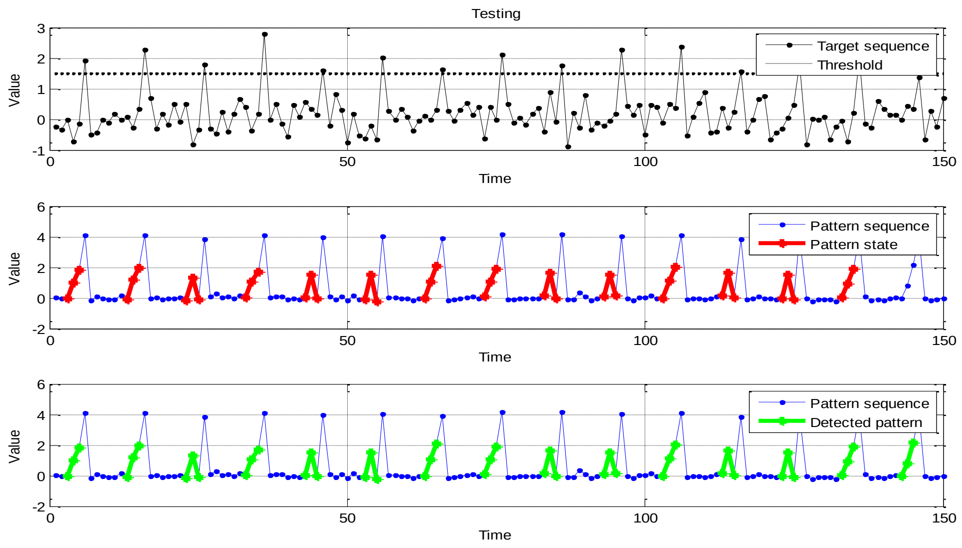


Figure 3.14 Testing result of the artificial data set

We can conclude from the above figures that the improved approach can accurately detect the inserted patterns of two different types, although the error, especially the false positive error, is inevitable. However, this experiment does prove that this improved approach can be applied as expected to overcome the obstacle that patterns gather into separated regions.

Example 2: Detect temporal patterns in Henon map sequences

The Henon map [32] is a typical dynamical chaotic system. This discrete-time system can be defined as the following equations with two output variables x and y:

$$\begin{cases} x_{t+1} = -ax_t^2 + y_t + 1 \\ y_{t+1} = bx_t \end{cases}, \tag{1.41}$$

where

t is the time index;

a and b are the parameters to determine whether the system state is chaotic.

To make sure the system is chaotic, we can set the parameters to the classical values: $a = 1.4, b = 0.3$. The following figure illustrates the Henon attractor when $a = 1.4, b = 0.3$:

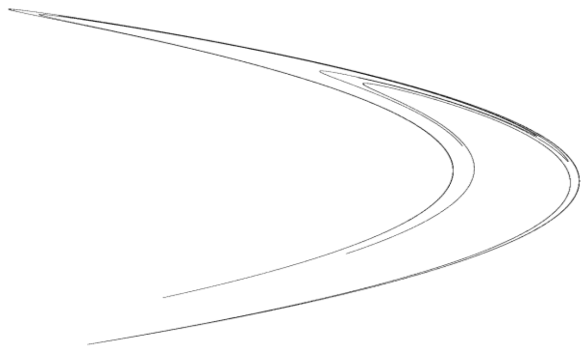


Figure 3.15 Henon map attractor when a=1.4, b=0.3

Moreover, the initial values of x and y can be defined as: $x_0 = 0.5, y_0 = 0.5$. We can generate a dataset containing 1000 time points by the equations with the predefined parameters. Part of the data is shown as:

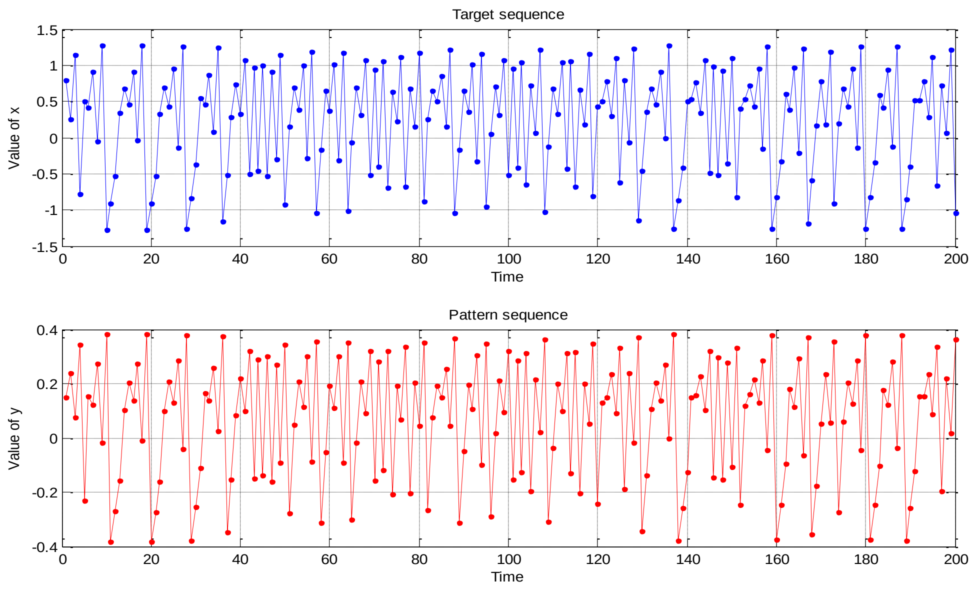


Figure 3.16 Part of the Henon map data

In this example, we choose the variables x and as the target sequence and pattern sequence respectively. Similar as the procedure in the example 1, we can initialize the parameters as the following list:

Process	Parameters
MRPS	$Q_x = 1, \tau_x = 1, Q_y = 3, \tau_y = 1, k = 1$
GMM	$c = 1.1, NC_p = 3, NC_n = 1$
Optimization	$\sigma = 0.5, \eta = 1, \boldsymbol{\beta} = \begin{bmatrix} 0.3 & 0.1 & 0.1 \end{bmatrix}$

Table 3.2 Parameter list for example 2

The event function can be defined as:

$$g(\mathbf{x}_t) = \begin{cases} 1, & x_{t+1} > 1.1 \\ -1, & x_{t+1} <= 1.1 \end{cases}, \qquad (1.42)$$

We can apply the first half of all 1000 observations for training, the remaining half for testing. The following figure illustrates the validation result with training dataset:

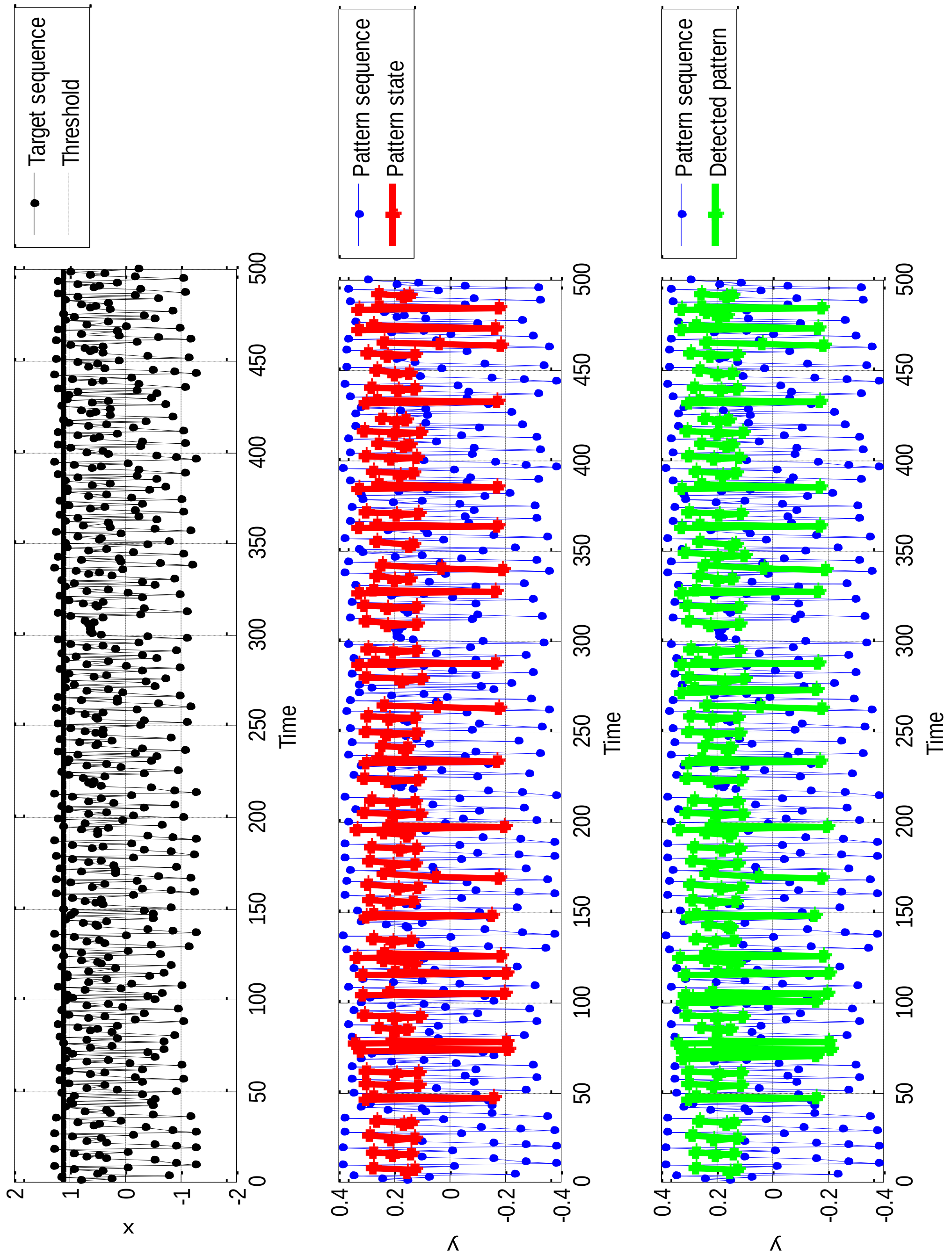


Figure 3.17 Validation with training data in example 2

The total number of embedding vectors from the training dataset is 438, 57 of which are pattern vectors. The following confusion matrix gives the accuracy of the result:

Estimation / Observation	Pattern	Normal
Pattern	57	0
Normal	6	375

Table 3.3 Confusion matrix for training data validation

The total accuracy is shown as:

True positive	False positive	True negative	False negative	Total accuracy
57	6	375	0	98.6%

Table 3.4 Accuracy of the validation result

The following figure illustrates the result on testing data:

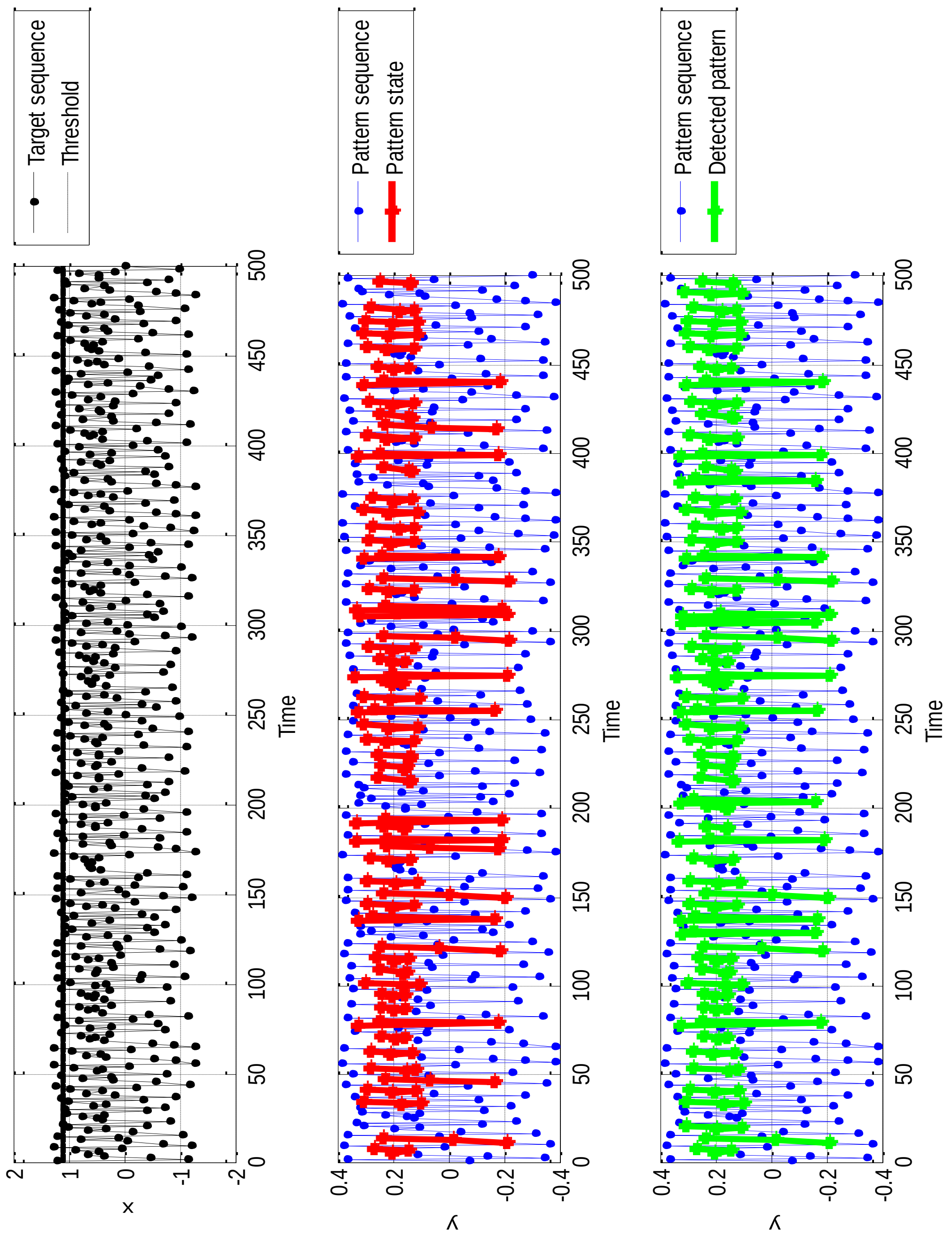


Figure 3.18 Testing Result of example 2

The total number of embedding vectors from the testing dataset is 438. The following confusion matrix gives the accuracy of the result:

Estimation Observation	Pattern	Normal
Pattern	53	5
Normal	7	373

Table 3.5 Confusion matrix for testing data

The total accuracy is shown as:

True positive	False positive	True negative	False negative	Total accuracy
53	7	373	5	97.2%

Table 3.6 Accuracy of the testing result

To make a comparison, we can apply a single Gaussian distribution to model the distribution of pattern points. With the same parameter values, the validation and testing results for the single Gaussian distribution method are shown in the following tables:

Validation results:

Estimation Observation	Pattern	Normal
Pattern	50	6
Normal	7	375

Table 3.7 Validation result for single Gaussian distribution method

True positive	False positive	True negative	False negative	Total accuracy
50	7	375	6	97.0%

Table 3.8 Accuracy of the validation result for single Gaussian distribution method

Testing results:

Observation \ Estimation	Pattern	Normal
Pattern	41	17
Normal	5	375

Table 3.9 Testing result for single Gaussian distribution method

True positive	False positive	True negative	False negative	Total accuracy
41	5	375	17	95.0%

Table 3.10 Accuracy of the testing result for single Gaussian distribution method

Meanwhile, we can compare the success rate of prediction between these two methods. The success rate of prediction can be defined as:

$$\rho = \frac{TruePositive}{TruePositive + FalsePositive + FalseNegative} \tag{1.43}$$

The success rate of prediction represents the reliability of the method. The comparison is shown as:

Process \ Method	Single Gaussian distribution	3-component GMM
Validation	79.4%	90.5%
Testing	65.0%	81.5%

Table 3.11 Comparison of the success rate of prediction

Apparently, the performance of the method with 3-component GMM is superior to the single component method in the same situation. The experiment not only proves that the newly improved approach can give a satisfactory result of classification, but also confirms that this approach is able to precisely detect multiple types of temporal patterns in multivariate data system. So this approach does contribute to reduce the error of miss classification caused by the diversity of shapes among different patterns.

CHAPTER 4 A SOFTWARE PROGRAM FOR TEMPORAL PATTERN DETECTION

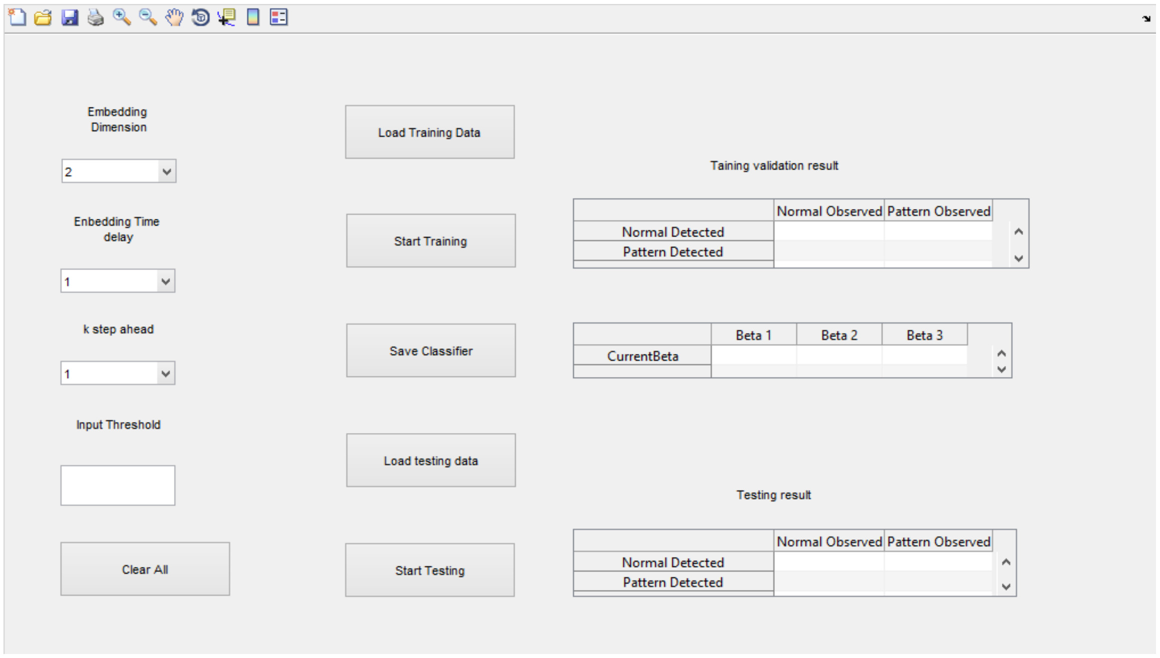
One of the major objectives of the research in this thesis is to establish an applicable framework, based on which it is possible to create a deliverable software product so that users can understand the basic procedure of the temporal pattern detection approach and test any data they are interested in. To achieve this requirement, a MATLAB-based software program, MATPAD, has been developed. In this chapter, we will follow the procedure of objected-oriented software development to elaborate basic structure and detailed functionality step by step.

4.1 Initial Problem Statement

Currently, the approach for temporal pattern detection based on multivariate reconstructed phase space (MRPS) embedding and Gaussian mixture model (GMM) has been proposed in [6] and improved in this thesis. However, there is still not a software application can be used to conduct the pattern detection process on individual computing platform.

With the support of MATLAB complier and graphic user interface (GUI) layout editor, we can create an applicable GUI system and convert it to a deliverable program so that users are able to install and use it conveniently.

The following picture shows the prototype of the MATPAD GUI:



Embedding Dimension: 2

Enbedding Time delay: 1

k step ahead: 1

Input Threshold:

Taining validation result

	Normal Observed	Pattern Observed
Normal Detected		
Pattern Detected		

	Beta 1	Beta 2	Beta 3
CurrentBeta			

Testing result

	Normal Observed	Pattern Observed
Normal Detected		
Pattern Detected		

Figure 4.1 Software GUI prototype

4.1.1 Software Requirements Specification (SRS)

Eliciting the software requirements and specifying the details plays an important role during the development procedure. In this section, we will discuss the functional requirements and nonfunctional requirements of the software respectively.

Functional requirement:

1. This software allows users to load a ".mat" data file stored in the local computing device and will plot the data in a new figure. The data has to be an $m \times n$ matrix.
2. Users can customize the initial parameter values for multivariate RPS embedding on the GUI.
3. With the preset parameters, users can apply the improved MRPS-GMM method to do the training and testing process.

4. Users should be able to see the classifier's coefficients, the validation and testing results on the GUI. Meanwhile, the software will plot the detected patterns in a new figure.
5. Users can terminate the process and start over at any time.

Nonfunctional requirements:

1. This software should be compatible with Windows operating system. Users don't have to preinstall MATLAB to use the software.
2. This software won't cause external system error.
3. The software's code must be reusable and extendable for modification.

Since the software "MATPAD" is the original version, the SRS is basically focusing on implementing the temporal pattern detection process. It can be improved for future development.

4.1.2 Use Case Model

In this section, we will use diagram and textual descriptions to discuss the use cases. The use case diagram is shown as:

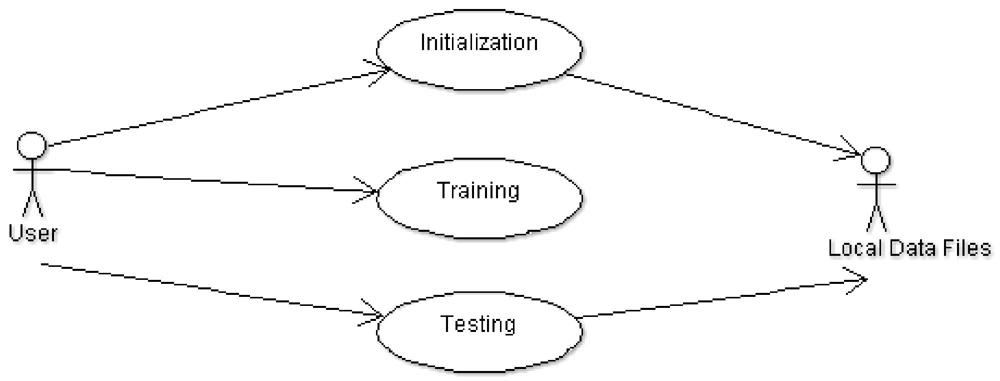


Figure 4.2 Use case diagram

Each use case can be described as the following activity diagrams and lists:

Use case 1:

Use case name	**Initialization**
Participating actor	User and Local Data Files
Flow of events	1. User pushes the Load Data button on the GUI. A selection window will be opened. 2. User selects the data from the selection window by double clicking the file. 3. User selects appropriate values for the embedding dimension, time delay and k-step ahead from the popup menus. 4. User types in the threshold value in the test edit box.
Entry condition	The GUI is deployed and visualized.
Exit condition	1. The message box shows “Load data succeed!” and each parameter is assigned with an initial value. 2. The message box shows “Load data fail! Please try again” or the initial value of at least one of the parameters is null.

Table 4.1 Description for the use case Initialization

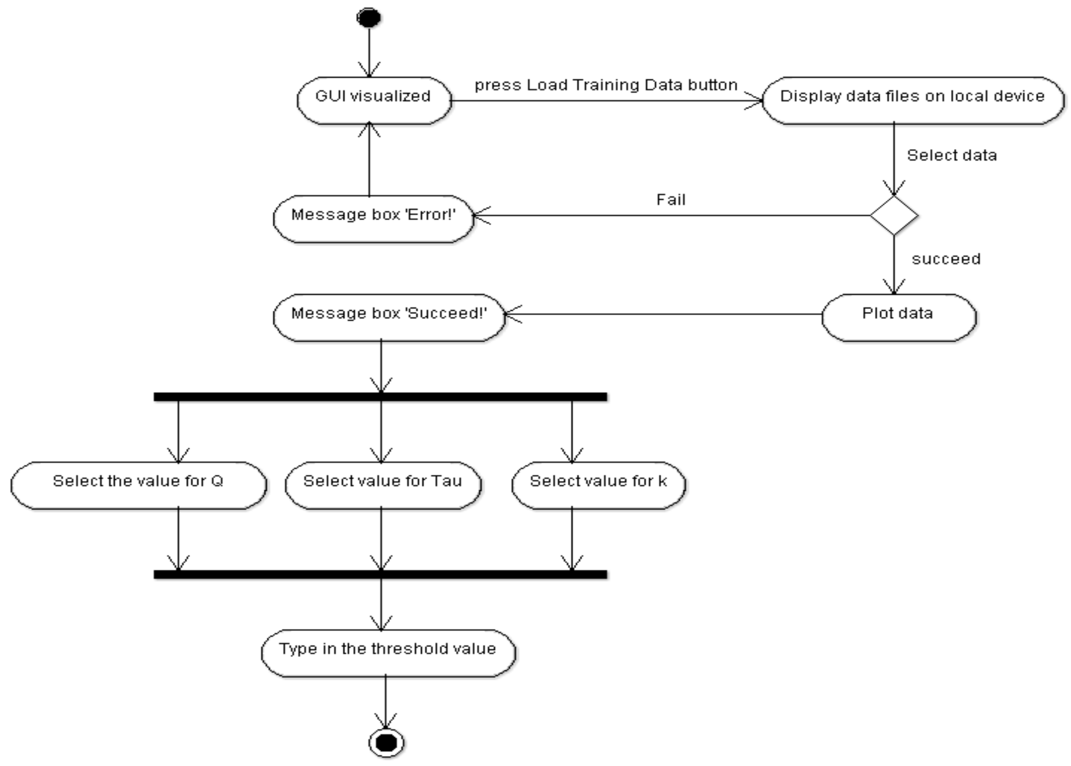


Figure 4.3 Activity Diagram for the use case Initialization

Use case 2:

Use case name	**Training**
Participating actor	User
Flow of events	1. User clicks the Training Data button on the GUI 2. User examines the result displayed on the GUI 3. User clicks the Save Classifier button on the GUI if the result is satisfactory. Otherwise the user can start over the process.
Entry condition	User clicks the Training Data button
Exit condition	User clicks the Save Classifier button

Table 4.2 Description for the use case Training

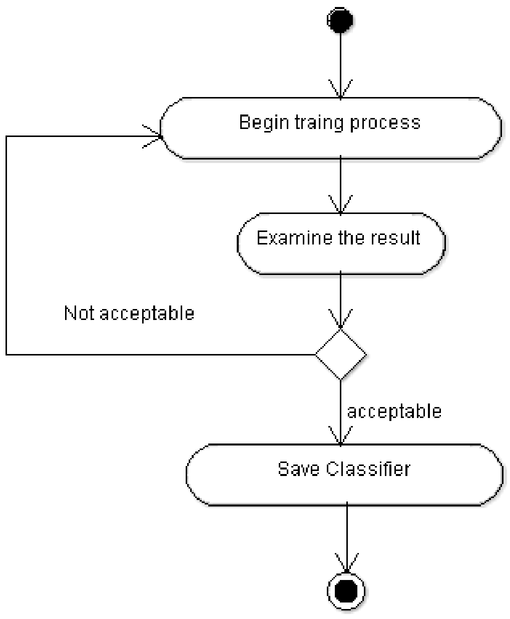


Figure 4.4 Activity Diagram for the use case Training

Use case 3:

Use case name	**Testing**
Participating actor	User
Flow of events	1. User clicks the Load Testing Data button on the GUI 2. User selects the testing data in the selection window 3. User click the Testing button on the GUI to see the testing result
Entry condition	User clicks the Load Testing Data button
Exit condition	User can see the testing result on the GUI

Table 4.3 Description for the use case Testing

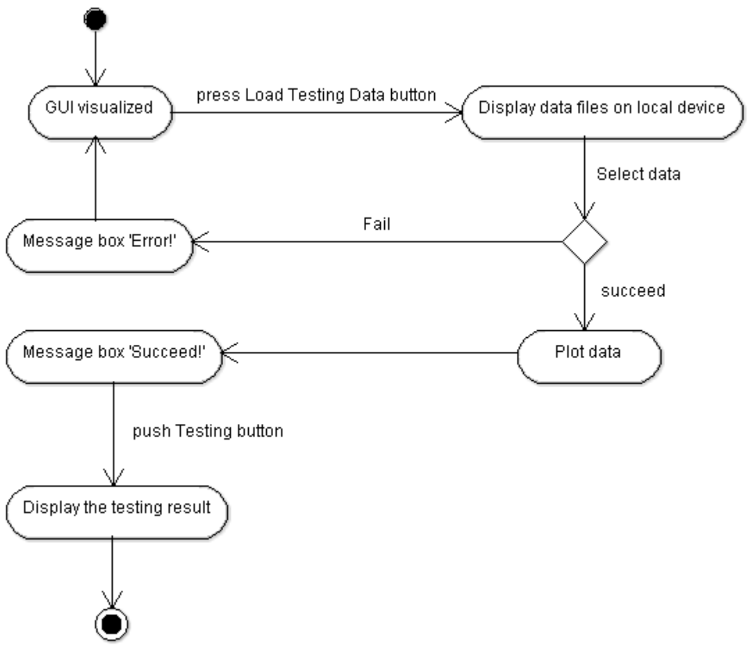
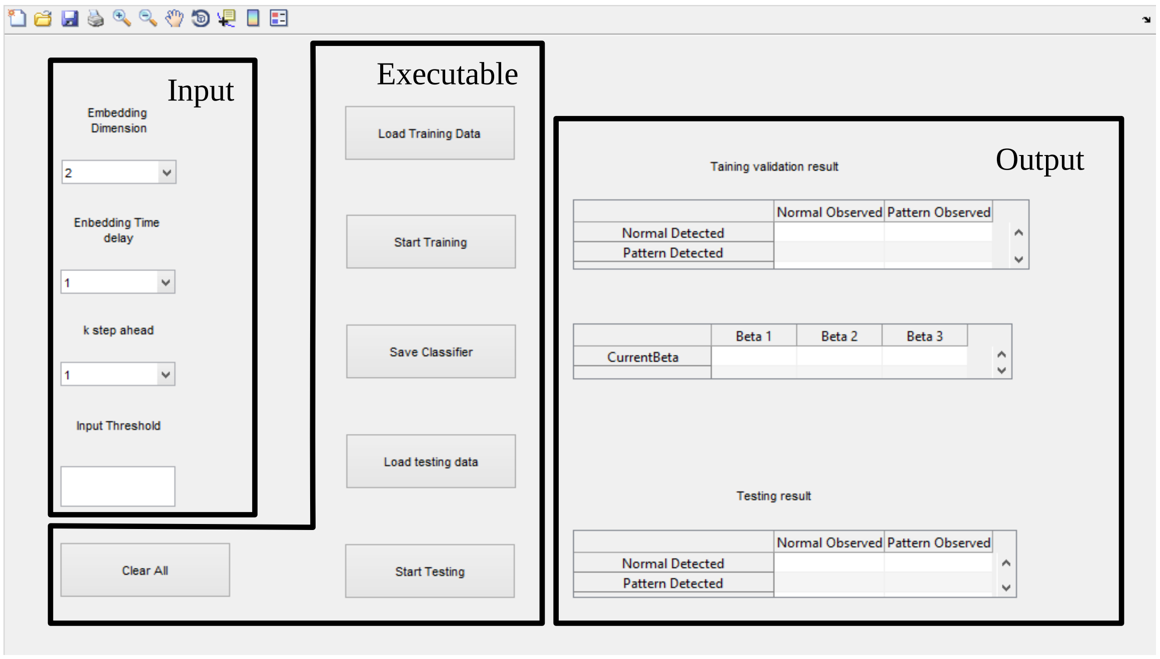


Figure 4.5 Activity diagram for the use case Testing

4.2 Function Structure

The controls on the GUI can be categorized into three different sets as the functionality: Input, Executable and Output.



Embedding Dimension: 2

Enbedding Time delay: 1

k step ahead: 1

Input Threshold:

Taining validation result

	Normal Observed	Pattern Observed
Normal Detected		
Pattern Detected		

	Beta 1	Beta 2	Beta 3
CurrentBeta			

Testing result

	Normal Observed	Pattern Observed
Normal Detected		
Pattern Detected		

Figure 4.6 Controls in three categories

The input controls can be used to get the initial values and assign them to each corresponding parameter’s handle so it can be called by external objects or functions. Each value can be changed during the process at any time.

The output controls are created to display the computing results of each step for users to examine the performance. Besides the three tables deployed on the GUI, the program will pop up a new figure window to illustrate the detected temporal patterns in the sequence after training and testing stage. Users can determine if it is necessary to re-train the model.

The executable controls are indispensable to the software. User can start the computing process by clicking the pushbutton for a certain stage. The following tables will explain the callback function of each control by pseudo code.

Load Training Data:

```
open the selection window for user to select Data;
if Data format is not correct
    pop up an error message box
else
    plot(Data);
    TrainingData = Data;
    update handles;
    pop up success message box;
end
```

Table 4.4 Pseudo code for the Load Training Data control

Start Training:

```
get values from the parameters’ handles;
get the TrainingData;
if TrainingData is empty
    pop up an error message box
else
    get EmbeddingVector by MRPSEmbedding;
    get PatternPoint and NormalPoint by Eventfunction;
    Fit GMM by the EM algorithm;
    Calculate pattern similarity;
    Calculate log odds;
    Establish the classifier;
    Establish the loss function;
    Minimize the loss function;
    get the classifier’s coefficients beta;
    Validation by training data and beta;
    Display confusion matrix;
    Pop up a figure to plot validation result;
    update handles;
    pop up success message box;
end
```

Table 4.5 Pseudo code for the Training control

Save Classifier:

```
if beta is empty
    pop up an error message box
else
    CurrentBeta = beta;
    Display CurrentBeta;
    Pop up success message box;
end
```

Table 4.6 Pseudo code for the Save Classifier control

Load Testing Data:

```
open the selection window for user to select Data;
if Data format is not correct
    pop up an error message box
else
    plot(Data);
    TestingData = Data;
    update handles;
    pop up success message box;
end
```

Table 4.7 Pseudo code for the Load Testing Data control

Start Testing:

```
get values from the parameters' handles;
get the TestingData;
get GMM fitted in Training process;
if TestingData is empty
    pop up an error message box
else
    get EmbeddingVector by MRPSEmbedding;
    get PatternPoint and NormalPoint by Eventfunction;
    Calculate pattern similarity;
    Calculate the log odds;
    Establish the classifier with CurrentBeta;
    get the classification result;
    Display confusion matrix;
    Pop up a figure to plot testing result;
    update handles;
    pop up success message box;
end
```

Table 4.8 Pseudo code for the Testing control

Clear:

```
clear every handle generated by the previous process
pop up a success message box;
```

Table 4.9 Pseudo code for the Clear control

Each control's callback function has a communicating mechanism to help user when the input variables or the output results are not acceptable.

Besides the three categories of controls mentioned above, user may notice that there are several functional buttons on the tool bar at the top of the GUI window. These buttons are built-in functional objects which can be used to modify the plots in the axes area and perform some basic file operations such as new, open, save and print. User can use these buttons as needed.

4.3 Operation Procedure

In this section, the detailed operation procedure will be demonstrated from users' view. In addition, some precautions for actual use will be pointed out as well.

4.3.1 Install and Uninstall

As it was mentioned in the previous chapters, the software was developed with MATLAB. To ensure the success of implementation, the MATLAB Compiler Runtime (MCR) is prerequisite for the software deployment. By using the MATLAB compiler, all the related functions, GUI files and the MCR can be compressed into a single Windows standalone executable package file. In this case, the package file name is MATPAD_pkg.exe.

To start installing, users need to put the package file in a new folder on the local device. To guarantee successful installation, there must be at least 1GB storage space left. Users can simply double click the package file and the installation will begin automatically. Before decompressing the executable file of the GUI, the MCR has to be installed first. This step will be skipped if the MCR has been pre-installed in the device. When the installation is done, users can see the GUI file MATPAD.exe, a readme.txt file and a MCRInstaller.exe file in the same folder with the package file MATPAD_pkg.exe. The GUI will be displayed after users double clicking the MATPAD.exe file.

To uninstall the software, users can simply delete the folder with all decompressed files.

4.3.2 Data Preprocess

Because of the functional limitation, the software can only process data files in ".mat" format. The data containing one target sequence and one pattern sequence can be saved in a ".mat" file as a 2 by N dimensional matrix where N is the total number of observations on timeline. In default setting, the first row will be seen as the target sequence. Meanwhile, it is prerequisite that the time sequences are the observations from a certain dynamic data system. Otherwise, the result will be meaningless.

In addition, the software currently is designed to detect temporal patterns to predict the occurrence of the events whose values are beyond a user-defined threshold. Users are responsible for selecting an appropriate value for the threshold. For now, the threshold based event function is the only one available in this software program, which means that users cannot customize the event function for specific problems. As a temporary solution, we can create an indicator sequence to mark the events in the original data sequence. The indicator sequence is based on a white noise series. First, users can mark the time points of the events according to the event function. Then, a predefined value will be assigned to the corresponding points on time line in the white noise series. For example, assume there is a 100-point data sequence $S(t)$ with events marked as:

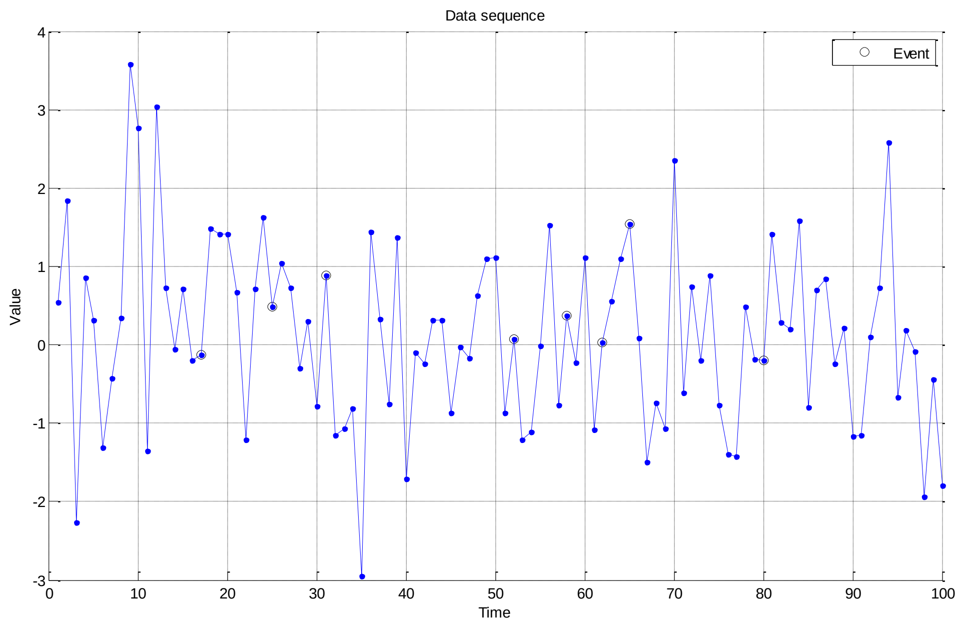


Figure 4.7 Data sequence with marked events

The indicator sequence has the same number of time point with data sequence and can be defined as:

$$A(t) = a * N(t)\,, \tag{1.44}$$

where

a is a coefficient to control the amplitude of the sequence;

$N(t)$ is a white noise series whose distribution follows $N(0,1)$.

To label the events on the indicator sequence, we can simply add a number d to the point's value. For example, if we set $a = 0.1$, $d = 3$ in this case, the indicator sequence will be modified as:

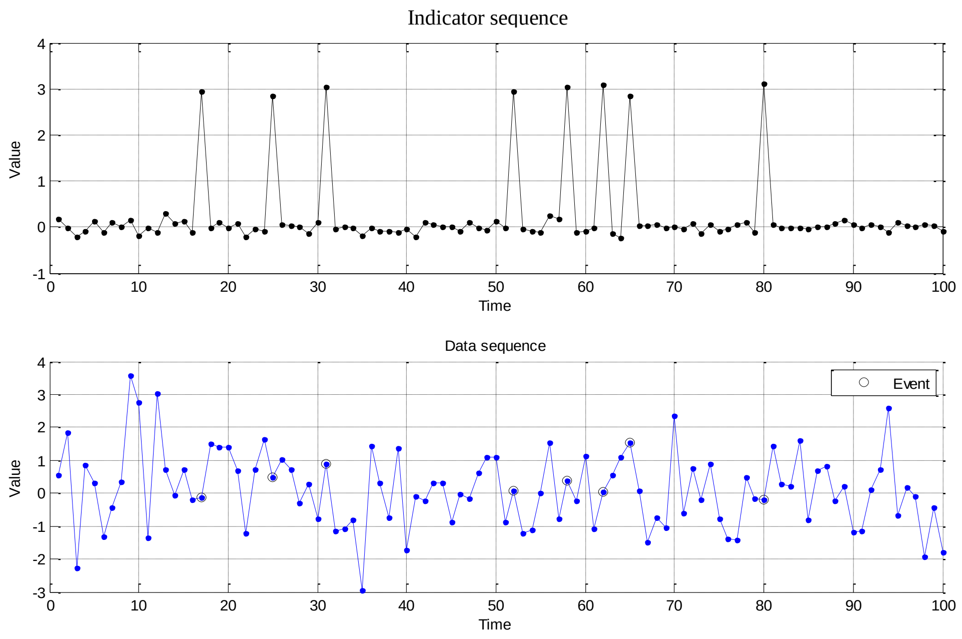


Figure 4.8 A data sequence with its corresponding indicator sequence

By adding the indicator sequence, users can easily recognize the events. Besides, even though this software program is not created for the univariate problems, the indicator sequence makes it possible to solve both univariate problems and multivariate ones with the same framework.

4.3.3 Flow Chart of the Operation Procedure

In conclusion, the whole operation procedure can be illustrated as the following flow chart:

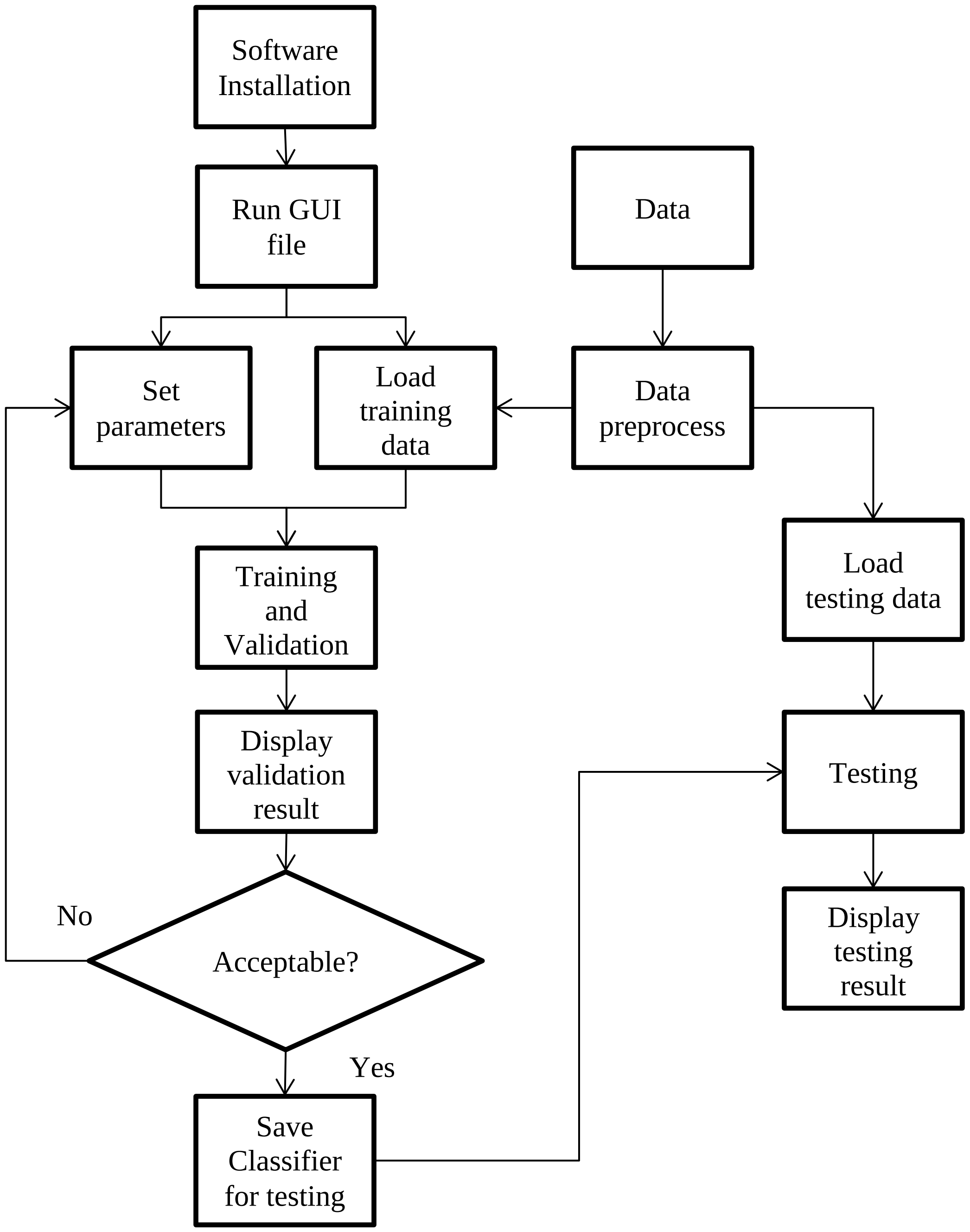


Figure 4.9 Flow chart of the operation procedure

CHAPTER 5 TEMPORAL PATTERN ANALYSIS FOR REAL WORLD PROBLEMS

In this chapter, we will utilize the improved MRPS-GMM approach and the MATPAD framework to analyze data in different study areas for real world problems.

5.1 Detect Temporal Patterns in Hydrology Time Series

In this section, we will conduct an experiment to demonstrate the procedure to detect temporal patterns in a hydrology time series data to predict when the river flow will beyond a threshold. The time series applied in this experiment is the monthly mean value of Saugeen River Flows from 1915 to1976, which is an available material of [33].

The time sequence is illustrated as:

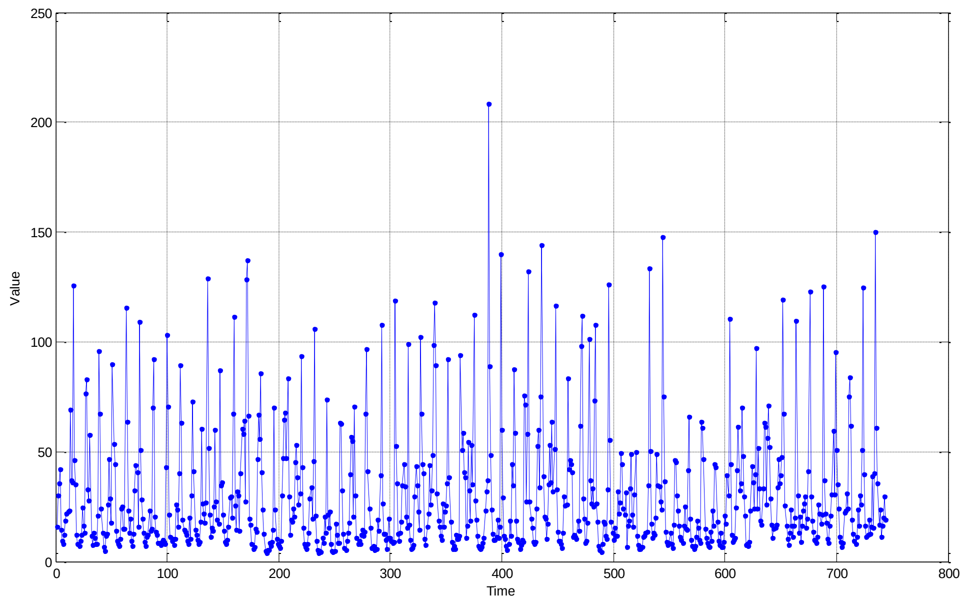


Figure 5.1 Time series of Saugeen River Flows

The total number of observations is 744. The unit of measurement is cubic meters per second $(\mathrm{m}^3/\mathrm{s})$. We use 500 points for training and the rest for testing. The event can be defined as: x_t is an event if $x_t > 80$. Although the event is already threshold-based, an indicator sequence still can be used to label the events as well. The parameters can be set to $a = 1$, $d = 20$, the artificial sequence is as:

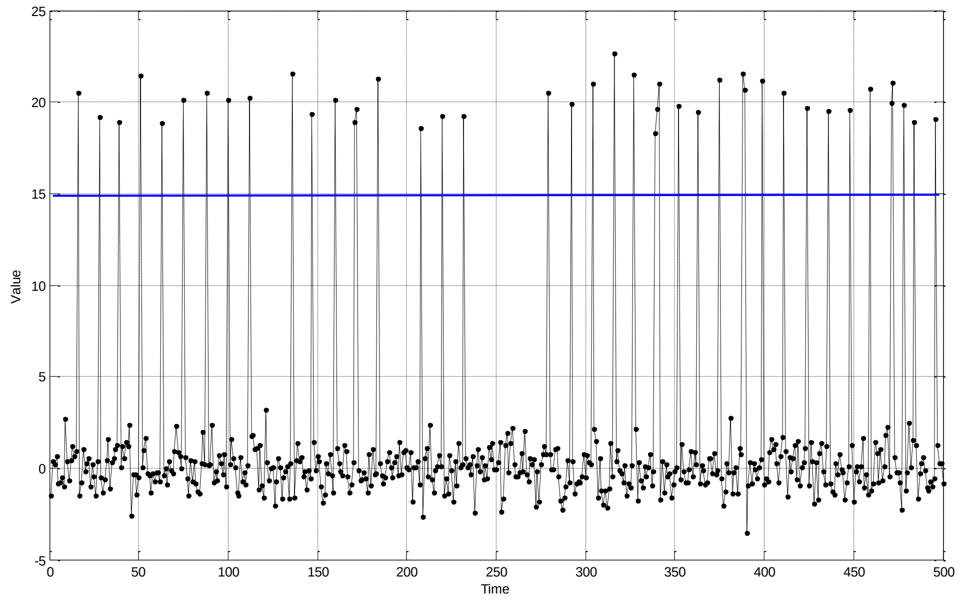


Figure 5.2 Indicator sequence for the Saugeen River Flows data

In this experiment, we choose $c = 15$ as the threshold to categorize the embedding vectors in the RPS. This threshold is an arbitrary value as long as it can separate all the events from the other points in the indicator sequence. In addition, it is noteworthy that the threshold in the original data sequence is not the one used in the MRPS-GMM approach.

The initialized parameters for training are:

Process	Parameters
MRPS	$Q_x = 1, \tau_x = 1, Q_y = 3, \tau_y = 1, k = 1$
GMM	$c = 15, NC_p = 2, NC_n = 1$
Optimization	$\sigma = 1, \eta = 1, \boldsymbol{\beta} = \begin{bmatrix} 1 & 0.1 & -2 \end{bmatrix}$

Table 5.1Parameters for training data of Saugeen River Flows

The subscripts x and y denote the indicator sequence and the original data sequence respectively. The training data will be re-used for validation. The experiment result is:

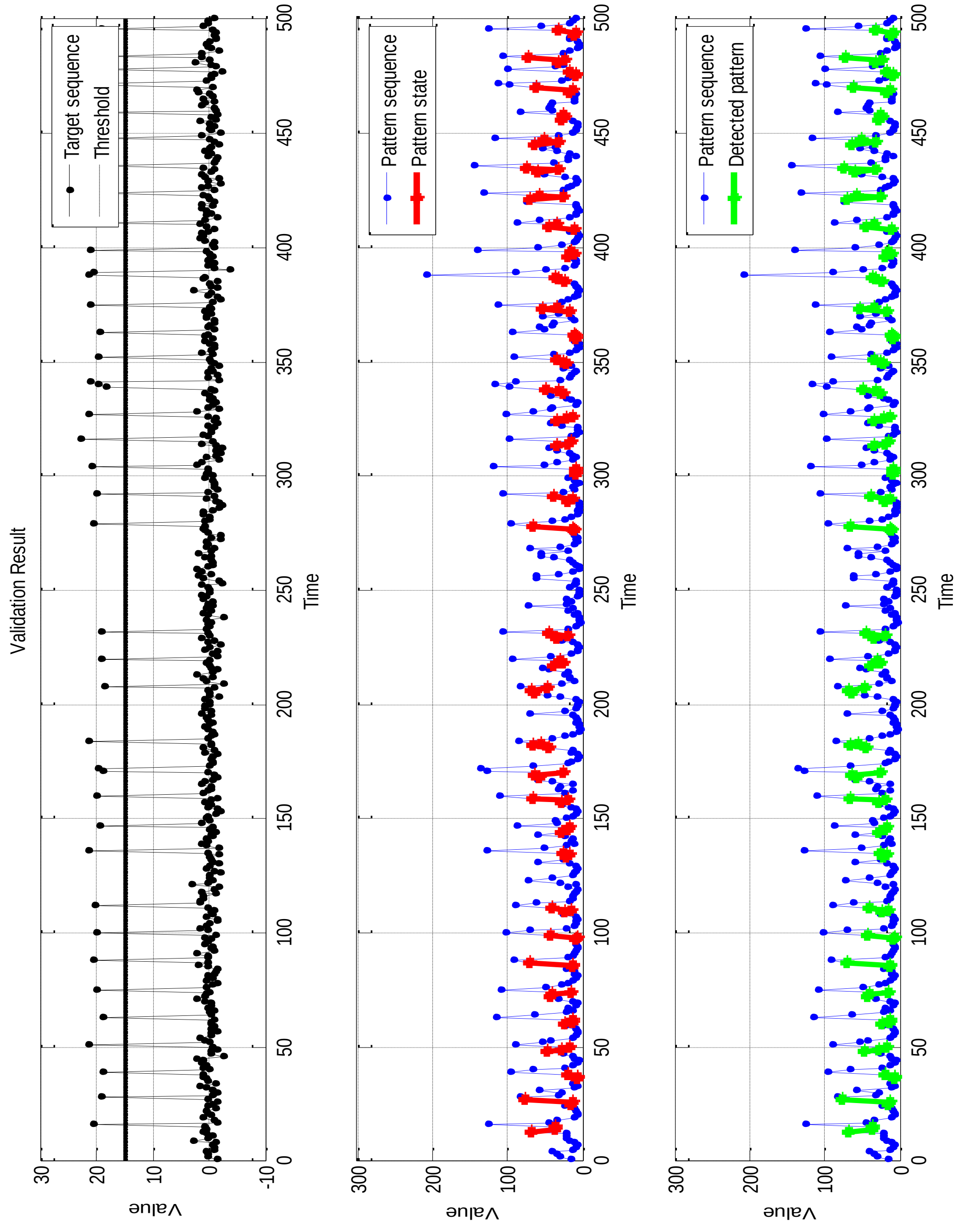


Figure 5.3 Validation result for the Saugeen River Flows data

Meanwhile, the confusion matrix for the validation result is given as:

Estimation / Observation	Pattern	Normal
Pattern	37	0
Normal	0	417

Table 5.2 Confusion matrix for the validation result of Saugeen River Flows data

The testing result is illustrated as:

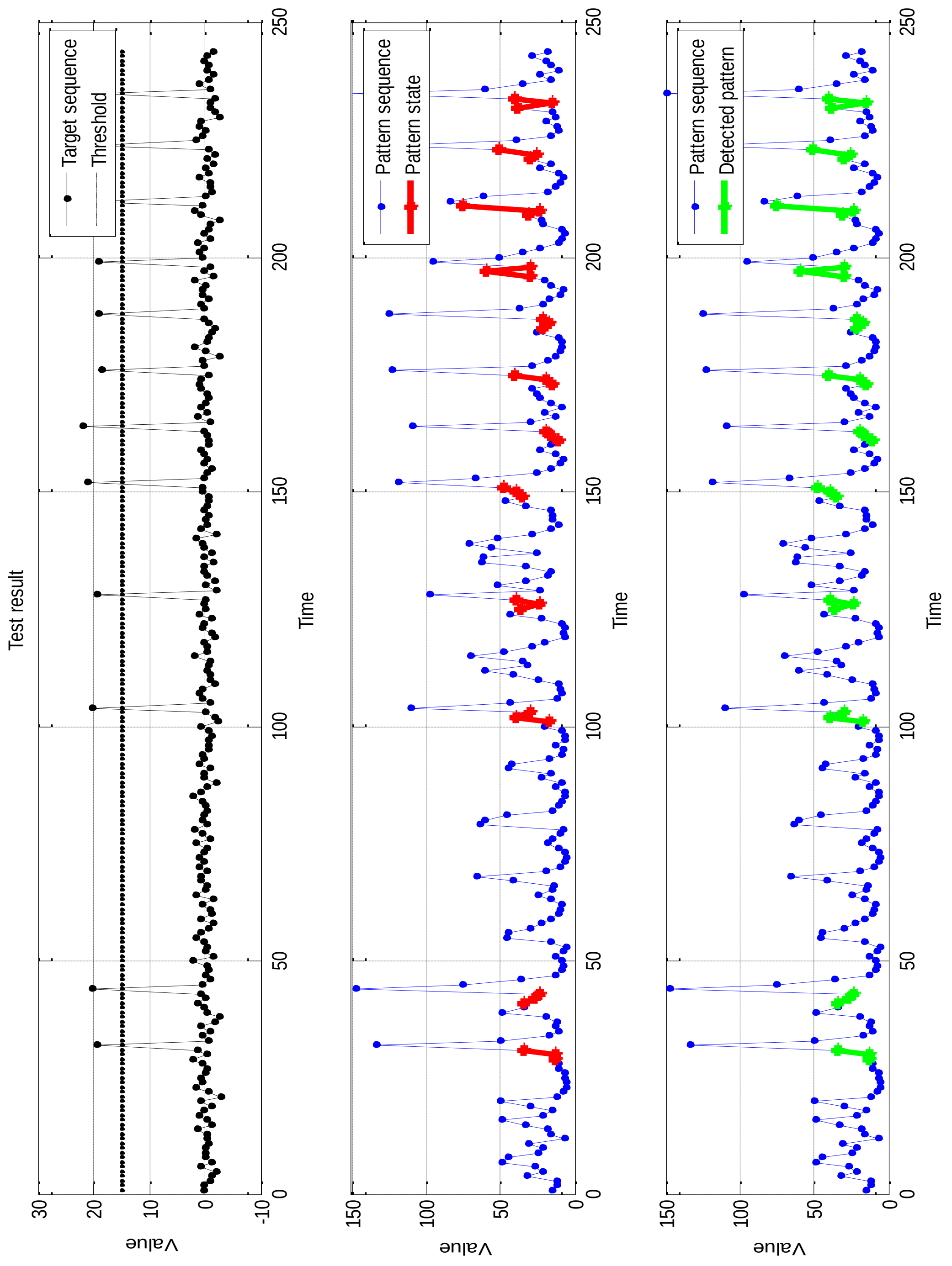


Figure 5.4 Testing result of Saugeen River Flows data

The confusion matrix is:

Estimation / Observation	Pattern	Normal
Pattern	12	0
Normal	0	216

Table 5.3 Confusion matrix for the testing result of Saugeen River Flows data

Both of validation and testing have the 100% success rate, so it can prove that there is no over-fitting problem in the approach. Also, this experiment data proves that the improved MRPS-GMM approach and the MATPAD framework are applicable to handle real problems in hydrology. The detected patterns can predict that the mean river flow speed will beyond $80\,\mathrm{m}^3/\mathrm{s}$ in the next month, which can be considered as a sign of flood jeopardy. So this approach will be contributive to flood prediction so that the prevention mechanism can be started to avoid property loss.

5.2 Detect Temporal Patterns in Finance Time Series

In this section, we will apply a dataset of Dow Jones Index at closing on 251 trading days ending on 26 August, 1994 to demonstrate how to discover the patterns to predict inflection points in a finance time series. This dataset is one of the available materials for [13].

The time series $\mathbf{x}(t)$ for this example is illustrated as:

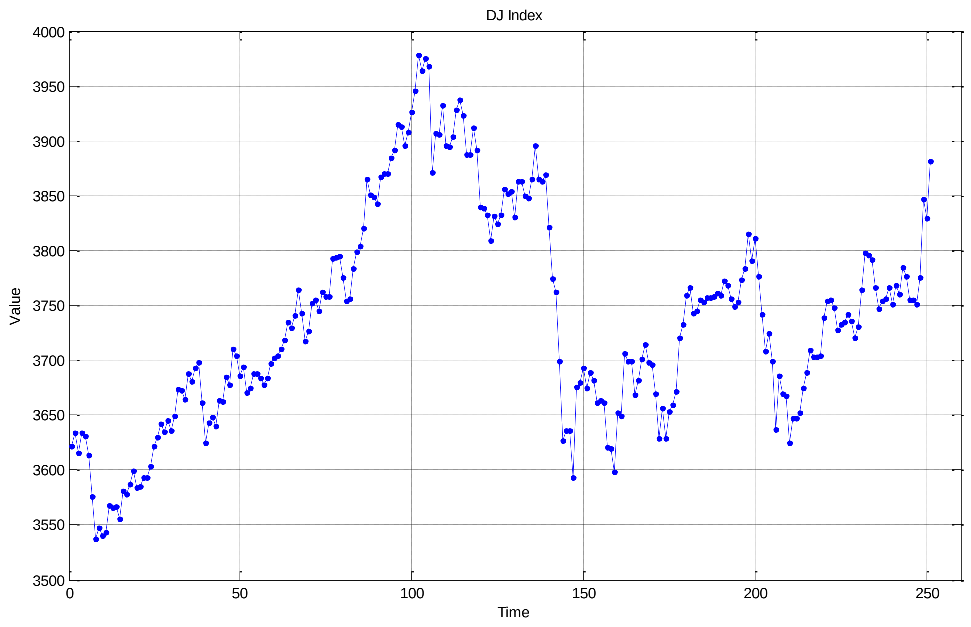


Figure 5.5 Dow Jones Index for 251 trading days

In stock market, one of the significant events is the inflection point, which is important to stock holders to analyze the market timing. In this example, the inflection point is defined as: x_t is an event if $x_{t-2} > x_{t-1} > x_t$ and $x_t < x_{t+1}$. The point's value is less than both of its adjacent points', as known as "rebound off the bottom".

To avoid the effect by the sequence's trend, we take difference to the original data sequence. Each point in the difference sequence is $y_t = x_{t+1} - x_t$. Meanwhile, the event for the difference sequence can be defined as: y_t is an event if $y_{t-2} < 0$ and $y_{t-1} < 0$ and $y_t > 0$. It is not difficult to understand that the two descriptions of the event are equivalent. The difference sequence is shown as:

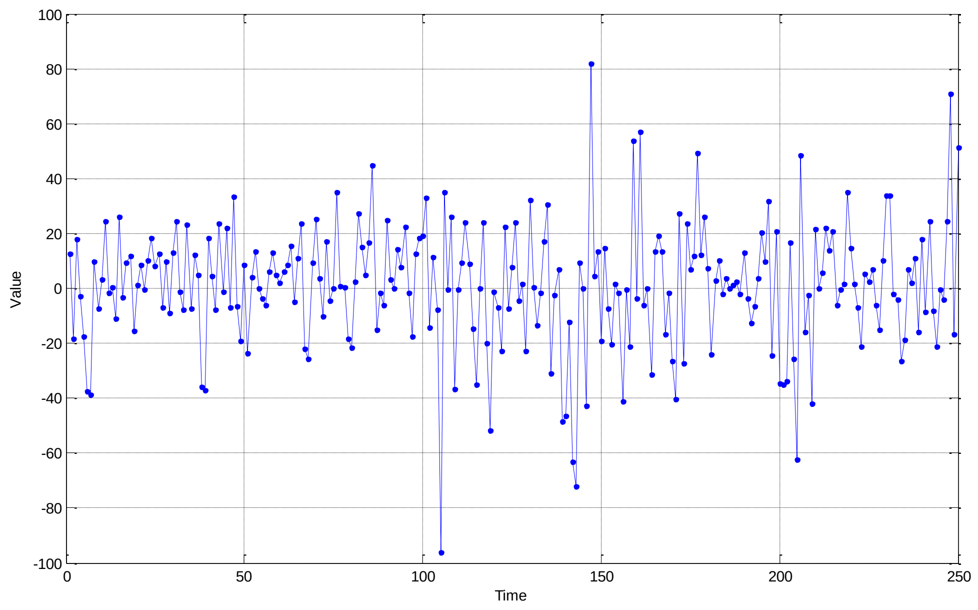


Figure 5.6 The difference sequence of DJ Index data

The first 200 points will be used for training and validation. The rest will be used for testing.

If we set the parameters $a = 0.1$, $d = 1$ for the indicator sequence, it can be created to describe the events in the difference sequence as:

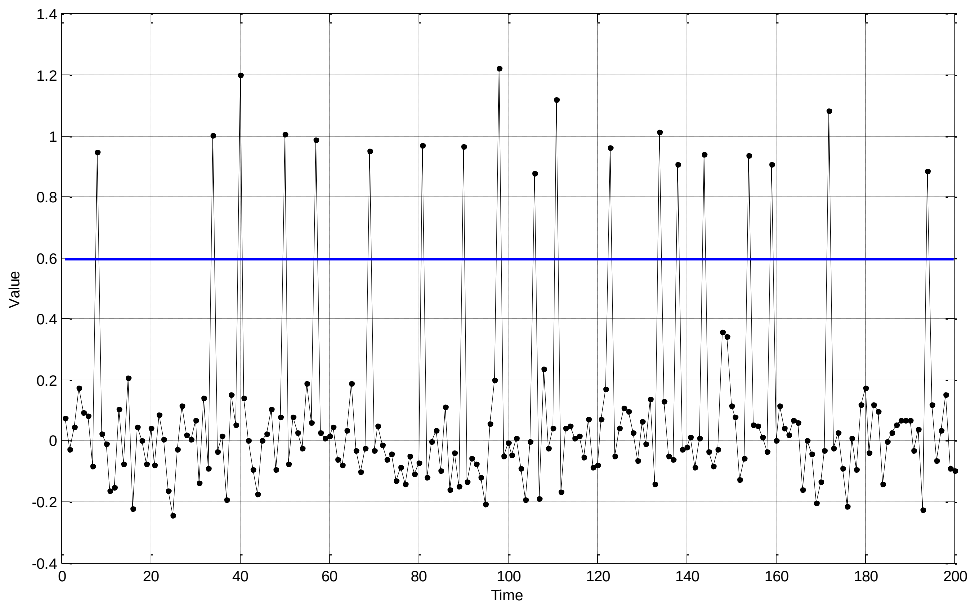


Figure 5.7 The indicator sequence for the difference sequence

The blue line on the figure is the threshold $c = 0.6$ for the threshold-based event function to categorize the embedding vectors. Each spike represents that there is an event in the difference sequence at the same time point.

The initialized parameters for reference are given in the following table:

Process	Parameters
MRPS	$Q_x = 1, \tau_x = 1, Q_y = 2, \tau_y = 1, k = 1$
GMM	$c = 0.6, NC_p = 2, NC_n = 1$
Optimization	$\sigma = 1, \eta = 1, \boldsymbol{\beta} = \begin{bmatrix} 1 & 0.1 & -2 \end{bmatrix}$

Table 5.4 Parameters for DJ Index data

The subscripts x and y denote the indicator sequence and the difference sequence respectively. The total number of embedding vectors is 178, 19 of which are categorized in pattern state. The validation result is given as:

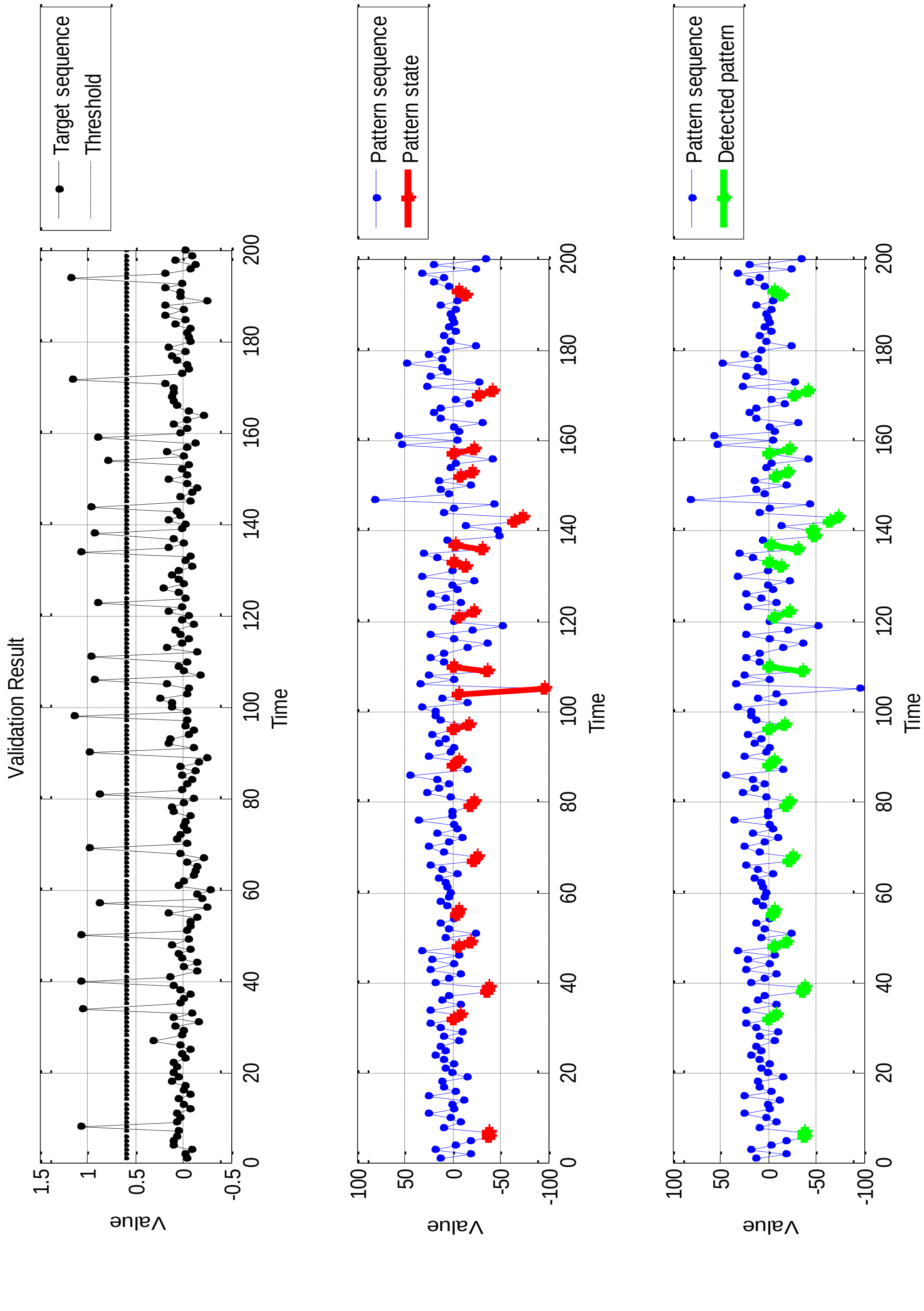


Figure 5.8 Validation result for the difference sequence of the DJ Index data

The confusion matrix for the validation result is:

Estimation / Observation	Pattern	Normal
Pattern	18	1
Normal	1	158

Table 5.5 Confusion matrix for the validation result of DJ Index data

The result can be considered as acceptable. The classifier can be used in testing process. In the testing data, there are 39 embedding vectors in total. The final test result is as:

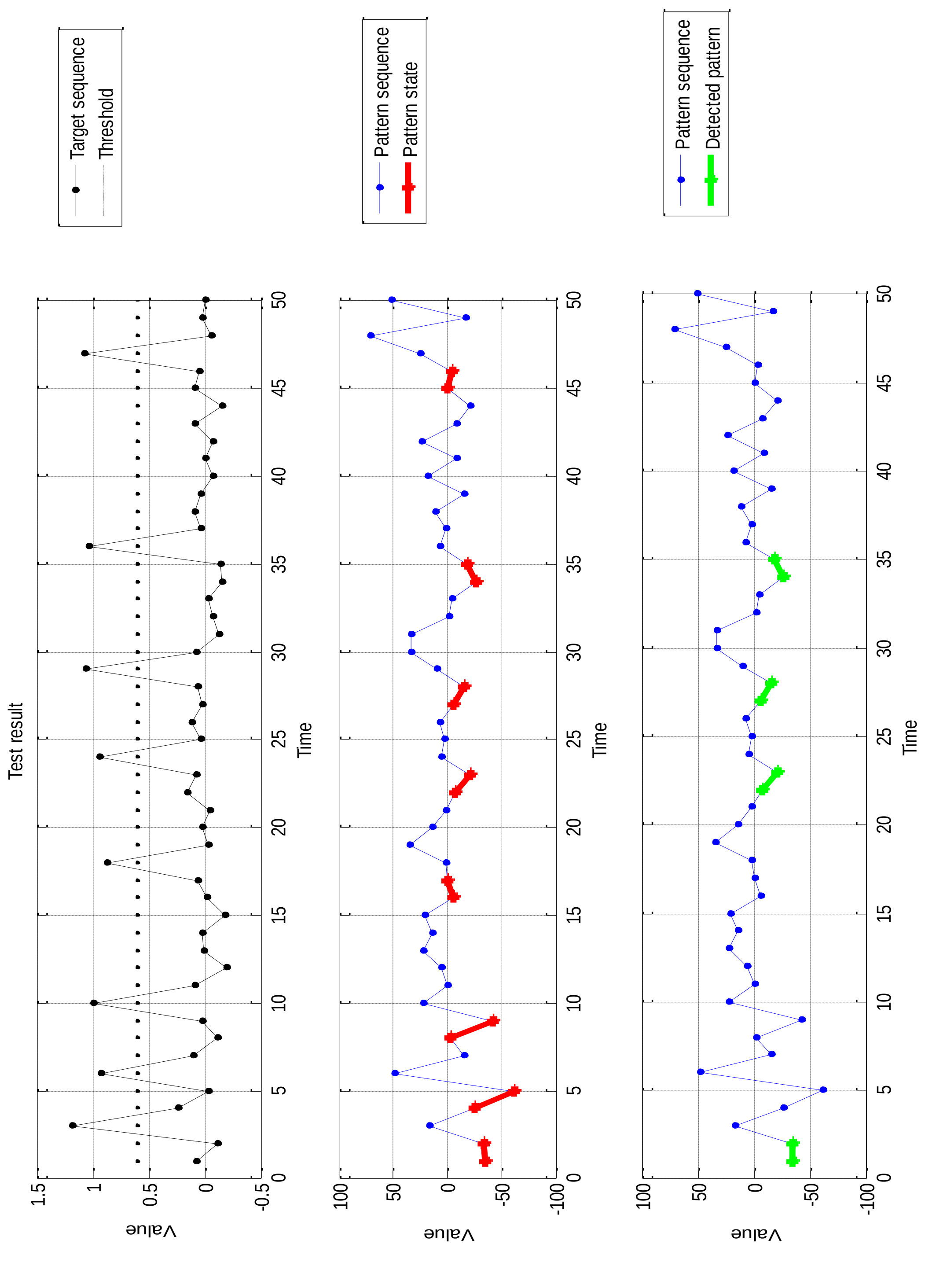


Figure 5.9 Test result of DJ Index data

The confusion matrix is:

Estimation / Observation	Pattern	Normal
Pattern	4	4
Normal	0	31

Table 5.6 Confusion matrix for test result of DJ Index data

We can map the patterns in the difference sequence back to the original data sequence as:

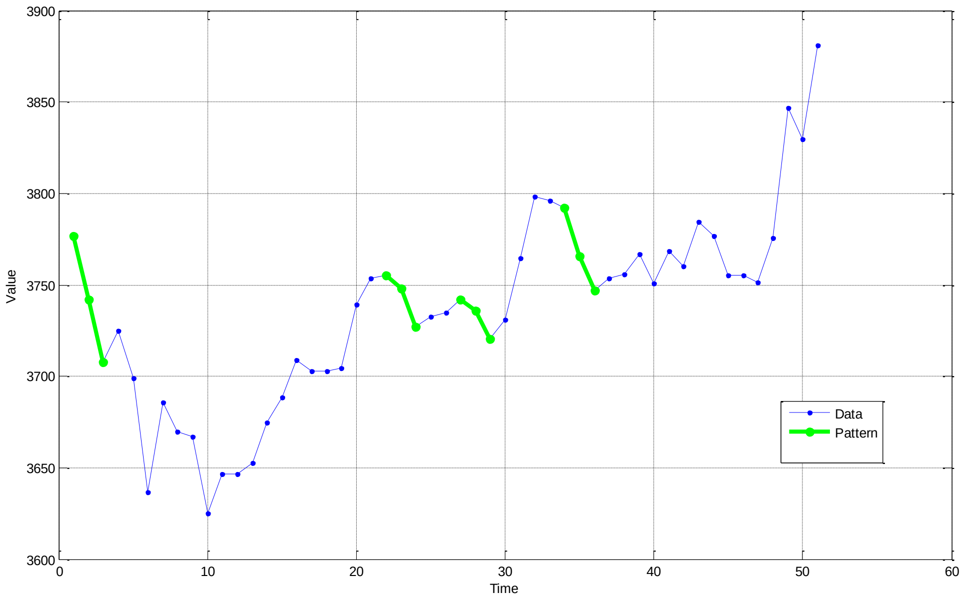


Figure 5.10 Patterns in the original test data sequence

From the figure, we can see that the detected four patterns can predict that the index will rise on the next trading day, although not all the eight events are predicted in the test data. As we know that there is an extremely complicated mechanism behind the stock index, which makes it difficult for the approach to predict every event in the sequence. However, the patterns detected in the sequence do predict the price's rebounding. Therefore, this approach is helpful and reliable indeed to be applied for finance time series analysis.

CHAPTER 6 CONCLUSIONS

6.1 Contributions

The major contributions of this thesis are:

1. The improved approach applies Gaussian mixture model on both normal state vectors and pattern state ones in the MRPS to fit a more precise distribution. Meanwhile, the approach applies a simplified classifier to reduce the computational amount.
2. A MATLAB based software is developed as a deliverable application for temporal pattern detection. This original software has a clear GUI so users can easily understand the whole process step by step. Moreover, this software is standalone executable, which means it can be used without MATLAB development environment. So it will be convenient to users to test the data on different computing devices.

6.2 Problems for Future Work

Although the newly improved approach and the MATLAB-based software have been tested to be applicable, there are still some problems cannot be ignored and need to be solved in the future.

1. The EM algorithm can only find the local minimal, which means the result depends on the initial values significantly. It is necessary to create a self-checking mechanism to tell if the final iteration result of the GMM is accurate enough and prevent the over-fitting problem from happening at the same time.

2. Currently, the software MATPAD can only perform the basic training and testing functions to detect temporal patterns in time series. Meanwhile, there is only one type of event functions can be defined and utilized. To improve the applicability, some extended functions, such as customizing event function, can be added and embedded in the GUI system. Moreover, other effective methods proposed in the previous research, such as Artificial neural network [34] , Fuzzy set [18] , can be implemented by the software as comparisons so users can choose the most appropriate method for specific problems.

BIBLIOGRAPHY

[1] S. M. Wu, "Dynamic Data System: A New Modeling Approach," *ASME Journal of Engineering for Industry,* vol. 99, no. 3, pp. 708-714, 1977.

[2] H. J. Steudel and S. M. Wu, "Dynamic Data System Modeling-Revisited," *Management Science. Jul.,* vol. 24, no. 11, pp. 1200-1202, 1978.

[3] R. J. Povinelli and X. Feng, "Temporal Pattern Identification of Time Series Data using Pattern Wavelets and Genetic Algorithms," in *Artificial Neural Networks in Engineering,* pp. 691-696, St. Louis, Missouri, 1998.

[4] P. Hong and T. Huang, "Automatic temporal pattern extraction and association," in *Acoustics, Speech, and Signal Processing (ICASSP),* pp. II-2005 - II-2008, *2002 IEEE International Conference on,* Orlando, 2002.

[5] C.-h. L. Lee, A. L. Liu and W.-s. Chen, "Pattern discovery of fuzzy time series for financial prediction," *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, no. 5, pp. 613-625, 2006.

[6] W. Zhang and X. Feng, "Predictive temporal patterns detection in multivariate dynamic data system," in *2012 10th World Congress on Intelligent Control and Automation (WCICA),* pp. 803 - 808, Beijing, 2012.

[7] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis,* Cambridge University Press, 1997.

[8] F. Takens, "Detecting strange attractors in turbulence," *Lecture Notes in Mathematics,* vol. 898, pp. 366-381, 1981.

[9] M. B. Kennel, R. Brown and H. D. I. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Phys. Rev. A,* vol. 45, no. 6, pp. 3403-3411, 1992.

[10] A. M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Phys. Rev. A* , vol. 33, no. 2, pp. 1134-1146, 1986.

[11] R. J. Povinelli, *Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events,* Ph.D. Dissertation, Marquette University, Milwaukee, 1999.

[12] G. Box and J. Gwilym, *Time Series Analysis: Forecasting and Control,* San Francisco: Holden-Day, 1976.

[13] P. J. Brockwell and R. A. Davis, Introduction to Time Series and Forecasting-Second Edition, New York: Springer-Verlag, 2002.

[14] J. Nocedal and S. J. Wright,, Numerical Optimization, New York: Springer, 2006.

[15] R. Paredes and E. Vidal, "Learning weighted metrics to minimize nearest neighbor classification error," *IEEE Transactions on Pattern Analysis and Machine,* vol. 28, no. 7, pp. 1100–1111, 2006.

[16] D. A. Dickey and W. A. Fuller, "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association,* vol. 74, no. 366, pp. 427-431, 1979.

[17] J. Pearl, Probabilistic Reasoning in Intelligent Systems, San Francisco: Morgan Kaufmann, 1988.

[18] H. Huang and X. Feng, "A fuzzy-set-based Reconstructed Phase Space method for identification of temporal patterns in complex time series," *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, no. 5, pp. 601-613, 2005.

[19] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms," *Journal of the Institute of Mathematics and Its Applications,* vol. 6, pp. 79-90, 1970.

[20] R. Fletcher, "A New Approach to Variable Metric Algorithms," *Computer Journal,* vol. 13, no. 3, pp. 317-322, 1970.

[21] D. Goldfarb, "A Family of Variable Metric Updates Derived by Variational Means," *Mathematics of Computation,* vol. 24, no. 109, pp. 23-26, 1970.

[22] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comput.,* vol. 24, no. 111, pp. 647-656, 1970.

[23] J. Sherman and W. J. Morrison, "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix," *Annals of Mathematical Statistics,* vol. 21, no. 1, pp. 124-127, 1950.

[24] X. Feng and O. Senyana, "Mining multiple temporal patterns of complex dynamic data systems," in *Proceedings IEEE Symposium on Computational Intelligence and Data Mining,* pp. 411 - 417, Nashville, TN, 2009.

[25] W. Zhang, X. Feng and N. Bansal, "Detecting temporal patterns using RPS and

SVM in the dynamic data systems," in *Proceedings IEEE Int'l Conf. Information and Automation*, pp. 209 - 214, Shenzhen, 2011.

[26] N. Bansal, X. Feng, W. Zhang, W. Wei and Y. Zhao, "Modeling temporal pattern and event detection using hidden Markov model with application to a sludge bulking data," *Procedia Computer Science,* vol. 12, pp. 218-223, 2012.

[27] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification 2nd ed, New York: John Wiley & Sons, Inc, 2001.

[28] T. Moon, "The Expectation-Maximization Algorithm," *IEEE Signal Processing Mag. ,* vol. 13, pp. 47-59, 1996..

[29] A. P. DEMPSTER, N. M. LAIRD and D. B. RUBIN, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological),* vol. 39, no. 1, pp. 1-38, 1977.

[30] R. J. Povinelli, M. T. Johnson, A. C. Lindgren and J. Ye, "Time Series Classification Using Gaussian Mixture Models of Reconstructed Phase Spaces," *IEEE Transactions On Knowledge And Data Engineering,* vol. 16, no. 6, pp. 779-783, 2004.

[31] W. Zhang, *Predictive Pattern Discovery in Dynamic Data System,* Ph.D. Dissertation, Marquette University, Milwaukee, 2013.

[32] M. Henon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics,* vol. 50, no. 1, pp. 69-77, 1976.

[33] K. Hipel and A. McLeod, Time Series Modelling of Water Resources and Environmental Systems, Amsterdam: Elsevier Science B.V., 1994.

[34] C. M. Bishop, *Pattern Recognition and Machine Learning,,* New York: Springer, 2007.

Appendix

A.1 Reusable source code for RPS embedding

```
function [ev,kv] = embedding(Ts,Ps,Q,tau,k)
%EMBEDDING A data mining function to get embedding vectors
%   This function returns embedding_vector (ev) and k_step_ahead_vector
(kv)
%   as the outputs of the embedding result. "ev" is where patterns lie
in
%   and "kv" is the same for events. Each row in ev is corresponding to
the
%   row in kv that has the same number of row (ev(n,:)~kv(n,:)). Each
pair of
%   rows shows an embedding vector and the k points after it.
%
%   TS (target_sequence): the sequence that contains events.
%
%   Ps (pattern_sequence): the sequence that is correlated with
target_sequence.
%                     pattern_sequence is used to find potential
patterns.
%
%   Q: embedding dimension.
%
%   tau: time delay.
%
%   k: k step ahead.
%
%   ev (embedding_vector): generated from pattern_sequence, which
contains
%   potential patterns.
%
%    kv (k_step_ahead_vector): k-column matrix to be used to find events.

M = length(Ts);
N = length(Ps);

if M<N
    disp('Length of target sequence must be no less than the one of
pattern sequence'); % check the pattern-sequence won't be oversized
else
    ev = zeros(N-(Q-1)*tau,Q);  % Initialization
    kv = zeros(M-k-(Q-1)*tau-1,k);

    if ~iscolumn(Ts) % check and convert row vectors into column
vectors
        Ts = Ts';
    end
    if ~iscolumn(Ps)
```

```
        Ps = Ps';
    end

    j = 1;  % build embedding vectors to search patterns
    l = N-(Q-1)*tau-1;
    for i=1:Q
        ev(:,i) = Ps(j:l+j);
        j = j+tau;
    end

    j = 1;  %  build k-step-late vectors to define events
    for i=1:k
        kv(:,i) = Ts((Q-1)*tau+1+j:M-k-1+j);
        j = j+1;
    end

    L = length(kv);
    ev = ev(1:L,:); % delete those redundant vectors in ev(no
corresponding vectors in kv)

end

end
```

A.2 Reusable source code for similarity measurement

```
function output_vector = phi(input_vector)
%Linear transformation to get the new lower-dimensional embedding
vector
%   input_vector is the original vector for transformation
%
%   output_vector is the new embedding vector
%
%   the relationship between the "similarity" and this transformation
is:
%   d(x1,x2) == ||phi(x1)-phi(x2)||^2

x = input_vector;
global Q

P = zeros(Q-1,Q-1); % initialization
a = Q-1;
for i=1:Q-1
    P(i,:) = a;
    P(:,i) = a;
    a = a-1;
end     % Create P: a symmetric positive definite matrix
L = chol(P);     % Cholesky decomposition of matrix P. L'*L==P

x = x(end:-1:1);  % reverse-ordered for easy computing
Delta_x = zeros(Q-1,1);
for j=1:Q-1
    Delta_x(j) = x(j)-x(j+1); % differenced embedding vector
```

```
end
Delta_x = Delta_x(end:-1:1);
output_vector = L*Delta_x;

end
```