A TEMPORAL ANALYSIS SYSTEM FOR EARLY DETECTION OF HEALTH CHANGES

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

JINGYI SHAO

Dr. James Keller, Thesis Supervisor

DECEMBER 2014

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

A TEMPORAL ANALYSIS SYSTEM FOR
EARLY DETECTION OF HEALTH CHANGES

presented by Jingyi Shao,

a candidate for the degree of Master Of Science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. James Keller, Ph.D.

Dr. Marjorie Skubic, Ph.D.

Dr. Mihail Popescu, Ph.D.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help, support, and friendship of so many people. In particular, I would like to thank Dr. James Keller for being my advisor and involving me in this particular research. I am grateful for his advice, time and patience during the writing of this thesis and throughout my research. Thank you Dr. Marjorie Skubic for agreeing to be on my committee and for everything I have learned from her classes. One of my very first class at MU was Dr. Skubic' s Building Intelligent Robot class, which, I would still like to say, is the most exciting and inspiring class I have ever had. Thanks are due to Dr. Mihail Popescu for agreeing to be on my committee. I would also like to thank my parents and my friends for their ceaseless support, without which this could not happen.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

LIST OF TABLES

1 Introduction

Rapid-growing population of elders in America causes concern for health care providers. While older adults want to live independently, many of the health changes go undetected, such as dementia, frailty and urinary tract infections (UTI). Early detection of health changes is critical to promote health while controlling healthcare costs. Identifying and assessing problems early provides a window of opportunity for intervention to solve the problems before they become serious. To make it possible for elders to live independently at home and yet get help from health care providers when small changes in health conditions take place, smart home technologies are developed to enhance safety and monitor health conditions via noninvasive sensors and other devices.

According to a population study from Stanford University, as new anti-ageing treatments become available, humans will live even longer. Soon, the average age of death will jump by a year every year [1]. Few would argue that living longer is not an attractive idea, but is it still such a good idea to live for an extra decade if it just involves 10 more years of illness and frailty? The answer is no, which is why health care providers endeavor to elongate people's health expectancy to be as close as possible to their life expectancy. Figure 1.1 illustrates researchers' goal on matching the health expectancy with life expectancy. The solid line shows a typical trajectory of decline in functional ability based on research and practice with older adults [2]. The typical trajectory includes plateaus where no measurable decline occurs and precipitous step-downs illustrating dramatic functional decline, often the result of a significant health event. The goal with proactive

care is to stop the decline through early detection and prediction of health problems, extend the length of the plateaus and reduce the depth of the steps.

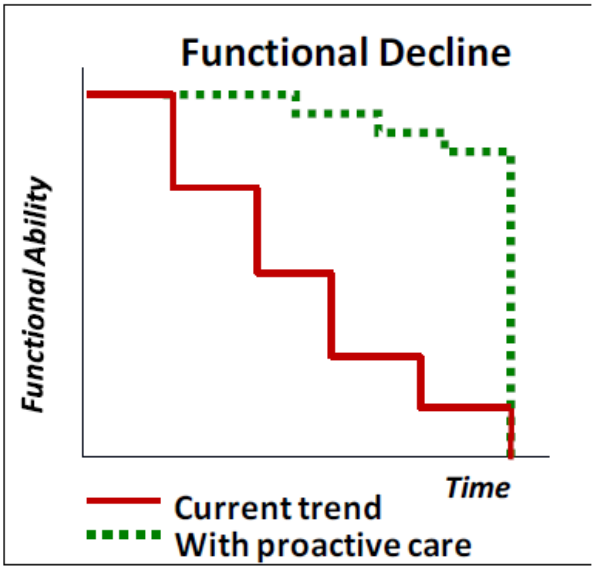


Figure 1.1 Trajectory of typical functional decline and the goal with proactive care

To promote this independent and ideal living model, the University of Missouri Sinclair School of Nursing and Americare Systems Inc. collaborated to create the TigerPlace domicile complex [3]. TigerPlace offers various kinds of services as needed, promoting independence and helping residents remain healthier and active longer by providing ongoing assessments for early illness recognition, and health promotion activities. This environment is designed to help residents avoid expensive and debilitating hospitalizations and, for most residents, to avoid relocation to a nursing home. An integrated monitoring system has been developed to capture data about a resident using non-wearable, environmentally-mounted sensors. This system is now being tested with TigerPlace residents using very simple algorithms to generate alerts on health change of a

single alert parameter and capture the clinical relevance of the alerts through feedback from the clinical staff. Although evidence shows that early illness alerts do improve health outcomes [2], half of the alerts generated are false alarms due to the current one-dimensional strategy.

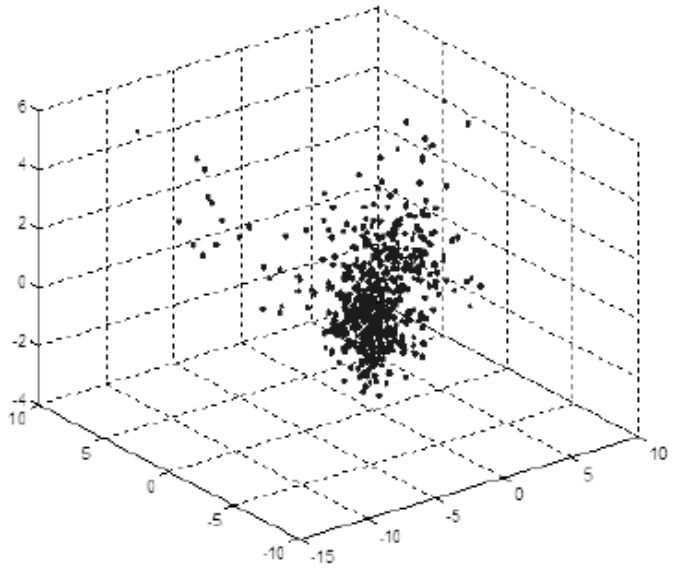


Figure 1.2 PCA reduction of sensor data collected on a TigerPlace resident with 21 features extracted from the motion and bed sensor data; each point represents one day. Self-report from this healthy elder showed 32 abnormal days out of 599 days total.

After looking at years of embedded sensor data, it is observed that normal days tend to form clusters in Euclidean feature space while abnormal days appear as outliers (Figure 1.2) [4]. Yet to determine the real health outliers can be complicated by the noise in the sensor data caused by sensor failure, visitor activity or extended absence. Figure 1.3 shows a subset of data with early illness alerts; the red dots represent normal days, and the blue dots indicating abnormal alert days according to clinicians' rating. While the

normal days tend to cluster, there are some normal days that appear as outliers. Sometimes there are also cases in which a new cluster forms due to a change in the resident's baseline health condition or daily routine.

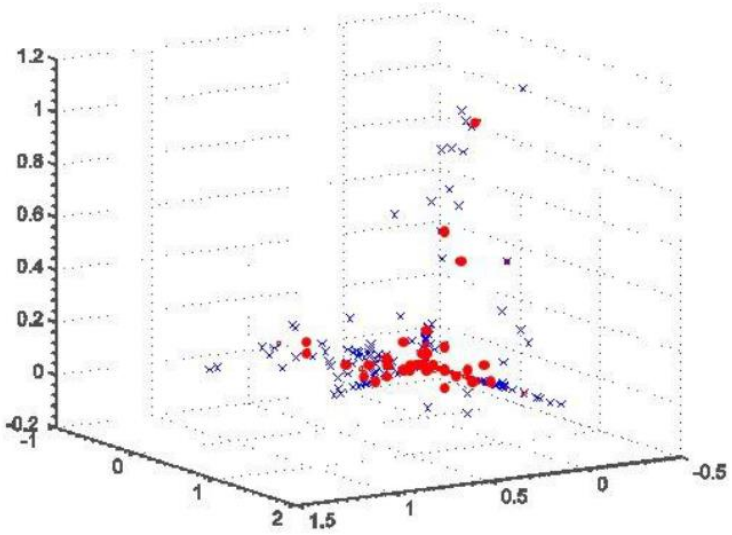


Figure 1.3 PCA reduction of six-dimension feature vectors from the early illness alert study. Red dots show normal days while blue dots indicate abnormal alert days as rated by clinicians.

To better analyze the wealth of the activity information to locate trends that correspond states of wellbeing, this thesis proposes a new system to build adaptive models for detecting health changes based on temporal analysis, including outlier detection, customization and adaption to new changes. Our hope is that by combining more than one alert parameter, we can find more of the potential alerts we are missing; and by using more sophisticated temporal analysis method we can capture more predictive alerts and more customized alerts that can help us detect more meaningful health changes before

they become big problems. Since we cannot have full access to all the embedded sensor data from TigerPlace at the moment, the system is tested using synthetic datasets which simulate gradual changes, sudden changes, changes of baseline health condition and system noise that might happen in the real-world data. This thesis will investigate different parameters settings, and evaluate the effectiveness of the temporal analysis method on accommodation to gradual trends and adaption to different patterns.

In Chapter 2, we briefly introduce the sensor monitoring system at TigerPlace as well as the previous work on early disease detection at MU. The GMM, the PCM and the AMPCM are also described in this chapter. In Chapter 3, we describe our goal and then propose the system algorithm. We first design a method for the initialization of the GMM using the PCM and the AMPCM, in which we discuss the advantage and disadvantage of the PCM and the AMPCM. Then we determine the mechanism of updating the model, firing an alert for anomaly and incorporating a new Gaussian for emerging normal behavior. In Chapter 4, several experiments targeting different parts of the system are conducted and multiple synthetic datasets are made to demonstrate the effectiveness of the proposed system. More details regarding the system is discussed in Chapter 5. Finally, conclusion is stated in Chapter 6.

2 Background and Related Work

This section will focus on previous work on early disease detection at MU as well as detailing the theories behind the Guassian mixture models, Possibilitic c-means clustering algorithm and Automatic merging possibilistic clustering method (AMPCM) algorithm that make up the heart of the proposed system.. To gain insight into any type of data, feature computation depends heavily on the quality and type of recorded signals. To ensure that a broad spectrum of activities is preserved to unearthing this information, let's discuss the sensor monitoring system at TigerPlace.

2.1 Sensor monitoring system at TigerPlace

MU Center for Eldercare and Rehabilitation Technology has been developing a sensor monitoring system for embedded health assessment (Figure 2.1). Various kinds of sensors are installed in TigerPlace apartments and the data from which are logged and stored on a secure server. A typical installation of sensors for a one bedroom apartment at TigerPlace is shown in Figure 2.2. Passive infrared (PIR) sensors are used to capture motion and also for localized activity, for example, in the refrigerator and kitchen cabinets to detect kitchen activity. A motion sensor is installed on the ceiling over the front door to detect apartment entrances and exits. A hydraulic bed sensor is installed on the bed mattress and used to capture sleep patterns like restlessness, heart rate and respiration rates [5]. For residents who always sleep in a recliner chair, the hydraulic sensor is also installed in the chair. Web cameras and the Microsoft Kinect are used for

fall detection and activity recognition [6][7]. A temperature sensor for the stove and oven is also included in the sensor networks to capture kitchen activity.

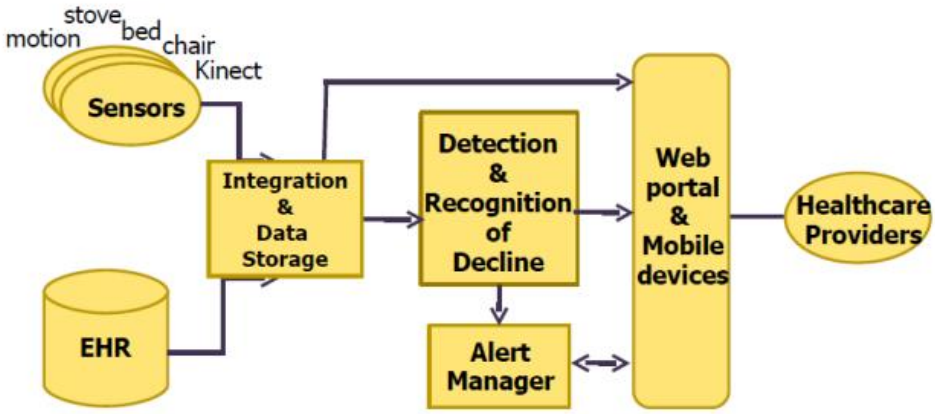


Figure 2.1 Integrated sensor network at TigerPlace with early illness alerts

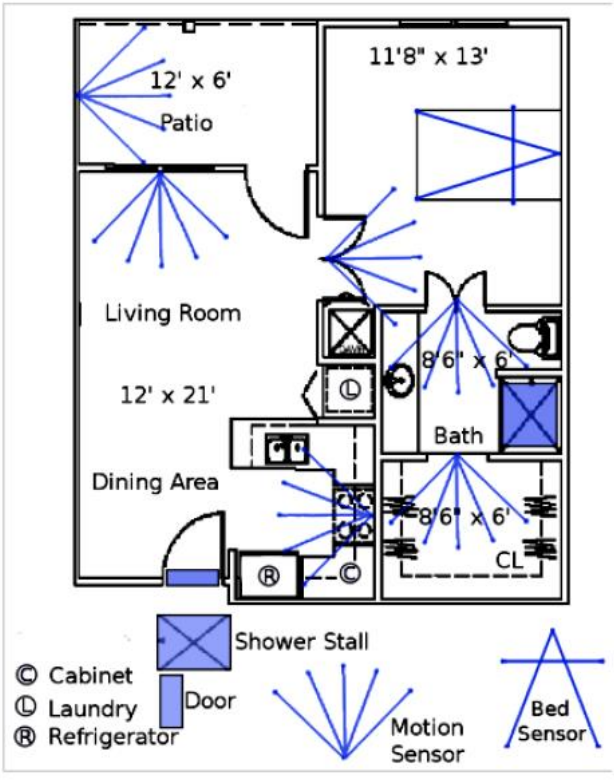


Figure 2.2 A typical installation of sensors for a one bedroom apartment at TigerPlace

The sensor networks has been deployed in TigerPlace apartments since 2005 fall and 49 networks has been installed to date with an average installation time of about five years [8][9]. Along with the electronic health record created by the nursing staff at TigerPlace, this longitudinal data capture has allowed the researchers to investigate the logged sensor data and corresponding health conditions retrospectively. By manually looking at data displays and visualizations, correlations between health conditions and logged sensor data have been observed. For example, changing sensor patterns have been observed with congestive heart failure and other cardiac problems, urinary tract infections , acute pain, depression, dementia, and rehabilitation [10][11][12][13]. It is shown that environmental sensors can be used for embedded health assessment.

2.2 Previous work on early disease detection at MU

2.2.1 Current alert system at TigerPlace

A one year prospective study to automatically analyze the data has been recently concluded at MU. In this study, a simple algorithm is used to look for changes in an individual's data patterns. If a change is detected, and alert is generated in the form of an email and sent to the TigerPlace clinical staff [14]. After receiving the alert, the clinician will inspect a two week window of sensor data before the alert, determine whether the alert is relevant and provide feedback to rate the clinical relevance of the alert on a five point scale, thus aiding the development of the alert algorithm.

The algorithm used to generate the alert was intentionally designed to be simple in hope that critical health changes would be captured. Table 2.1 shows the alert parameters and corresponding sensor data monitored for the early illness alerts. For each alert parameter,

the system computes a mean and standard deviation for the two week baseline window. Each day when a new sensor value comes in, if it varies from the mean beyond a pre-determined number of standard deviations, an alert is generated.

Table 2.1 Alert parameters and sensor data monitored for the current early illness alerts

Alert parameter	Sensors
Bathroom Activity	Sum of motion sensor hits in the bathroom (bathroom, shower, laundry)
Bed Restlessness	$x_{bedrest} = \sum_{i=1}^{4} i * x_i$, where x_i is the no. of bed restlessness level i hits
Bed Breathing Low	No. of bed breathing low hits
Bed Breathing Normal	No. of bed breathing normal hits
Bed Breathing High	No. of bed breathing high hits
Bed Pulse Low	No. of bed pulse low hits
Bed Pulse Normal	No. of bed pulse normal hits
Bed Pulse High	No. of bed pulse high hits
Kitchen Activity	Sum of kitchen motion sensor (kitchen, fridge, etc.) hits and stove/oven temperature high
Living Room Activity	No. of living room motion sensor hits
Chair Sensor Activity	Equivalent to bed sensor parameters above

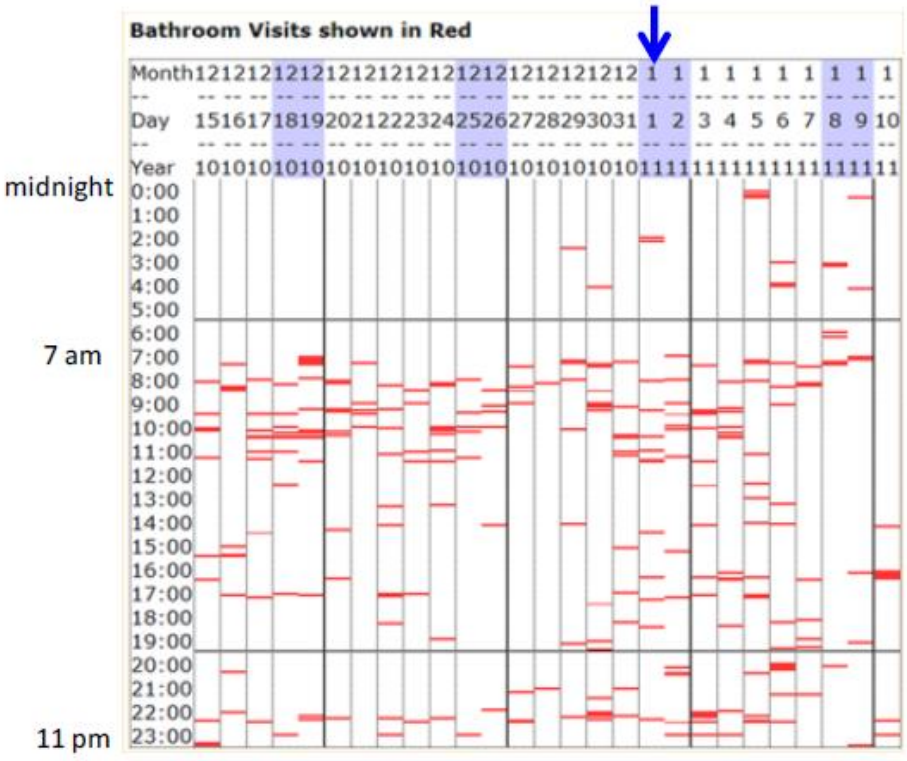


Figure 2.3 Bathroom visits for a TigerPlace resident. The arrow shows the day of the alert.

One example of an alert that detected the impending illness using this system is illustrated in Figure 2.3. An alert was generated on January 1 for increased nighttime bathroom activity. The resident was examined, diagnosed with a urinary tract infection, treated and recovered successfully [15]. Figure 2.4 shows an example from the retrospective analysis. If the alert system had been used on this resident, an early illness alert would have been generated 26 days before the emergency room visit and ultimate hospitalization.

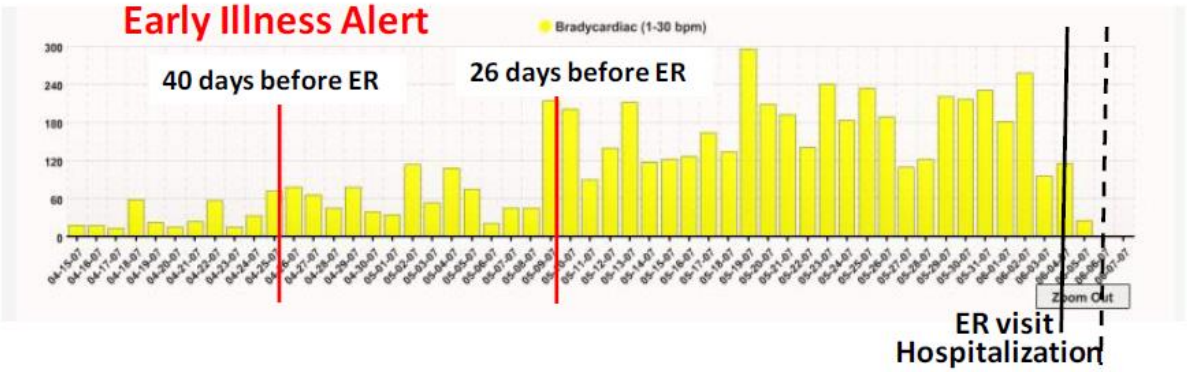


Figure 2.4 A retrospective example of an early illness alert which would have been generated 26 days before an emergency room visit, due to the low pulse events according to the bed sensor log.

Although about half of the alerts generated are false alarms using this one parameter strategy, the system manages to capture nearly all of the obvious health change [2].

2.2.2 Temporal activity analysis

Using the sensor data from TigerPlace, a matrix of computed features, $X = \{\overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_n}\}$, may contain an arbitrary number of dense regions that correspond to activity patterns. By using traditional unsupervised methods, X can be organized in to

c self-similar subsets, called clusters, based on an underlying similarity measure. Yet most conventional clustering techniques are unsuitable for this task as they rely on a pre-specified number of clusters c. Many techniques cannot support the temporal consideration of the data either.

For those reasons, Sledge proposed to use growing neural gas clustering (GNGC) to capture the temporal activity distributions formed by the features [16]. Different from traditional clustering, which involves optimizing the spatial location of a prototype set, $V = \{\overrightarrow{v_1}, \overrightarrow{v_2}, \dots, \overrightarrow{v_c}\}$, with respect to the set of unlabeled objects X, GNGC uses learning vector quantization to encode each manifold of signals using a finite set of reference vectors W. To accommodate the inclusion of new feature vectors, the size of W is allowed to grow as a function of the number of added attributes. A hybrid growing neural gas [17] and adaptive resonant theory scheme [18] is then utilized to update the best matching $\overrightarrow{w_k}$ and its connected neighbors on a dynamic lattice structure. Obsolete connections are allowed to die out due to an aging factor. The number of clusters, at a given time instant, can be determined by isolating non-connected lattices and find the number of unique graph paths.

It is shown that growing neural gas clustering is able to discover the underlying activity pattern from the sensor data and locate emergent activity trend. Also, GNGC provides a natural mechanism to discriminate clusters that are close spatially but separated in time. Yet we believe that the GMM approach is the best to create predictive measures for the onset of health changes.

2.3 Background modeling and clustering techniques

2.3.1 Gaussian Mixture Model and its use on adaptive background subtraction

The Gaussian distribution has important analytical properties but it suffers from significant limitations when it comes to modeling real datasets. Using linear superposition of multiple Gaussians give rise to complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy [19].

The Gaussian mixture density is described as follow,

$$P(X) = \sum_{k=1}^{K} \pi_k * N(x|\mu_k, \Sigma_k) \quad (2.1)$$

Each Gaussian $N(x|\mu_k, \Sigma_k)$ is a component of the mixture and has its own mean μ_k and covariance Σ_k. In equation (2.1), π_k is called mixing coefficient, and $\sum_{k=1}^{K} \pi_k = 1$ is required due to the normalization for both P(x) and the individual Gaussian components.

Stauffer and Gimson [20] utilized the flexibility of Gaussian mixture model to build a background model for real-time segmentation of moving regions in video sequences. The values of each particular pixel are modeled as a mixture of Gaussians. The distribution of each pixel is updated online to reflect the pixel value change over time. Certain Gaussians in the mixture are then selected to represent the background colors according to the weights and persistence of the Gaussians. Pixel values that do not fit the background distributions are considered foreground. Because most video sequences involve lighting changes, scene changes, and moving objects, it is more reasonable to

model the recent history of each pixel with a mixture of K Gaussian distributions. Each of these models has a probability, P, related to the number of occurrences in the data thus far.

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * N(X_t|\mu_{i,t}, \Sigma_{i,t}) \quad (2.2)$$

where $\mu_{i,t}$ and $\Sigma_{i,t}$ are the mean and covariance matrix of a normal distribution *N*, *N* is the Gaussian probability function, and $\omega_{i,t}$ is the weight of ith mixture at time t.

$$N(X_t|\mu_t, \Sigma_t) = \frac{1}{(2\pi)^{d/2}|\Sigma_t|^{1/2}} exp[-\frac{1}{2}(X_t - \mu_t)^T {\Sigma_t}^{-1}(X_t - \mu_t)]. \quad (2.3)$$

Here d is the feature dimensionality, and the weights are subject to the constraint $\sum_{i=1}^{K} w_{i,t} = 1$. The covariance matrix $\Sigma_{i,t}$ is always assumed to be of the form $\Sigma_{i,t} = \sigma_i^2 \boldsymbol{I}$ to avoid costly matrix inversion and enable the real-time performance.

To determine which of the Gaussians are most likely produced by background processes, the Gaussians are ordered by the value of w_i/σ_i. This value shows how confident that corresponding distribution is in representing the background distribution, the smaller the variance and the higher the weight, the more confident the Gaussian. The first B distributions are chosen as the background model, where

$$B = argmin_b(\sum_{k=1}^{b} w_k > T), \quad (2.4)$$

where T is a measure of the minimum portion of the data that should be accounted for by the background.

A new pixel value will be checked against the existing K Gaussian distributions and the closest match will be found. A match is defined as pixel value within N times the standard deviation of a distribution. The parameters and probability associated with this

winning distribution are updated by this pixel value. If the closest to the match happen to be the first B distribution, then the pixel value is counted as background; if not, then the pixel value is counted as foreground.

If the new pixel value is unable to find a match with any of the K distributions, the least probable distribution is replaced by a new normal distribution with mean value as the new pixel value and high variance and low weight.

The online update equations for the parameters of the Gaussian mixtures are as follows:

$$w_{k,t} = (1 - \alpha)w_{k,t-1} + \alpha M_{k,t} \tag{2.5}$$

where α is the learning rate and $M_{k,t}$ is 1 for the Gaussian that matched and 0 for the remaining Gaussians. If the pixel is a match to a component, then α is added, increasing $\boldsymbol{w_{k,t}}$ during the update, otherwise the second term of equation (2.5) disappears and the weight is reduced by the update. Note that all the weights need to be normalized to 1 after finishing updating the new pixel value to the Gaussian mixture model.

The means and covariances for the unmatched distributions remain the same while the mean and covariance for the matched distribution should be updated as follows:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \tag{2.6}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{2.7}$$

where

$$\rho = \alpha N(X_t|\mu_k, \sigma_k) \tag{2.8}$$

is the learning factor for adapting current distributions.

With the involvement of the two significant parameters: α, the learning constant and T, the proportion of the data that shall be accounted towards the background, the system is proved to be able to robustly adapt to lighting changes, slow moving objects and repetitive motions (like tree leaves blowing back and forth).

Another important feature about this system is that when a new pixel value is incorporated into the mixture, the existing model of the background is still kept. The original background remains in the Gaussian mixture until it becomes the Kth most probable and a new color is observed. Therefore, if an object stays stationary just long enough to become part of background and then it moves, the distribution describing the previous background will still exist with the same μ and σ, but with a lower weight.

2.3.2 Possibilistic C-means Algorithm (PCM)

Unsupervised learning methods are always used to reveal the organization of patterns into sensible clusters. In hard clustering, data is divided in to distinct clusters, where each data point belongs to exactly one cluster. This can be described by a partition matrix U where

$$u_{ij} \in \{0,1\} \; for \; all \; i \; and \; j$$

$$and \;\; \sum_{i=1}^{C} u_{ij} = 1 \;\; for \; all \; j. \qquad (2.9)$$

In fuzzy clustering, data points can belong to more than one cluster to some degree. The Fuzzy C-Means (FCM) is the most popular fuzzy clustering algorithm [21]. Let *U* denote a fuzzy partition matrix generated by the FCM algorithm. The elements u_{ij} of *U* satisfy the following conditions:

$$u_{ij} \in [0,1] \; for \; all \; i \; and \; j$$

$$0 < \sum_{j=1}^{N} u_{ij} \leq N \; for \; all \; i \; , and$$

$$\sum_{i=1}^{C} u_{ij} = 1 \; for \; all \; j \tag{2.10}$$

The FCM is derived by minimizing a cost function of the form

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m d_{ij}^2 \tag{2.11}$$

subject to the fuzzy partition matrix restraint in equation (2.10).

So the FCM objective function is formulated as

$$\boldsymbol{J = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m d_{ij}^2 \; + \sum_{j=1}^{N} \eta_i (\sum_{i=1}^{C} u_{ij} - 1)}, \tag{2.12}$$

where m is the fuzzifer, d_{ij} is the dissimilarity between x_i and the representative of the *j*th cluster, and η_i is the Lagrange multipler.

The Fuzzy C-means algorithm is a very useful clustering method, but requiring the probabilistic constraint that the sum of the memberships of a data point to all the classes sum up to 1 makes its memberships not always correspond to the degree of compatibility to the cluster. This also leads to its inaccuracy in a noisy environment.

To improve this weakness, Krishnapuram and Keller [22] proposed a possibilistic approach (PCM) to clustering, in which they relaxed the constraint on the sum of the memberships. It is shown that the PCM is more robust in noisy environment and its memberships can actually be interpreted as typicality of the points belonging to the clusters.

Due to the relaxation of the membership constraint, to let the membership represent degrees of compatibility, the PCM memberships must meet the following condition:

$$u_{ij} \in [0,1] for\ all\ i\ and\ j$$

$$0 < \sum_{j=1}^{N} u_{ij} \leq N\ for\ all\ i\ , and$$

$$\max_{i=1,\ldots,C} u_{ij} > 0\ for\ all\ j \tag{2.13}$$

Simply relaxing the constraint will lead to a trivial solution of the memberships: the criterion function is minimized by assigning all memberships to 0. So a constraint is added to the PCM objective function to make representative data points have high memberships while unrepresentative points have low memberships. The objective function that satisfies the requirements can be formulated as:

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m d_{ij}^2 + \sum_{i=1}^{C} \eta_i \sum_{j=1}^{N} (1 - u_{ij})^m . \tag{2.14}$$

The fuzzifier m controls the rate of the membership value decrease to 0, and η_i is a "scale" parameter, which corresponds to the size of the cluster or "zone of the influence" [23]. The first term of the objective function requires the distances from the feature vectors to the cluster center to be as small as possible, while the second term forces the membership values to be as big as possible in order to minimize J.

Differentiating the objective function with respect to u_{ij}, setting it to 0 and solving with respect to u_{ij} will give us the necessary condition on the membership matrix to minimize J:

$$u_{ij} = \frac{1}{1+(\frac{d_{ij}^2}{\eta_i})^{\frac{1}{m-1}}} \tag{2.15}$$

The value of η_i not only determines the relative significance of the two terms in the objective function, but also determines the distance at which the membership values becomes 0.5, and thus is called the size of the cluster or " zone of the influence". The value of η_i is crucial in determining the values of the typicality and the prototypes. It is suggested by Krishnapuram and Keller to run the FCM first and use the membership matrix from the FCM algorithm to get the value of η_i, using the equation:

$$\eta_i = \frac{\sum_{j=1}^{N} u_{ij}^m d_{ij}^2}{\sum_{j=1}^{N} u_{ij}^m} \tag{2.16}$$

The prototype of the cluster i, a_i, is determined by necessary condition:

$$a_i = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}^m} \tag{2.17}$$

Since equations (2.15) and (2.17) are coupled and cannot give closed form solutions, in order to obtain estimates for u_{ij} and a_i, we can use the iterative algorithmic scheme to update the membership matrix and the cluster centers alternately until there is little change made to the objective function.

Pseudo Code for PCM Algorithm:

Fix the number of clusters C, fuzzifier m and K.

Initialize the Possibilistic C-partition $U^{(0)}$ using the FCM algorithm and estimate η_i using (2.16).

Set the iteration counter t=1.

Step 1: Compute the prototypes with $U^{(t)}$ using (2.17).

Step 2: Update $U^{(t+1)}$ using (2.15).

If $|| U^{(t+1)} - U^{(t)} || < \varepsilon$:

Stop;

Else:

set t=t+1 and return to Step 1.

It is worth mentioning the PCM's unique mode-seeking property. After examining the objective function of the PCM, we can see that it can only be minimized when the prototypes of the clusters are located in dense regions. If there are N dense regions in the feature space, and assuming that the algorithm is initialized with C>=N as the number of clusters, then with the FCM as initialization, each prototype should converge to one of the N dense regions. In other words, the PCM would produce coincident clusters. On the other hand, if the algorithm is initialized with C<N as number of clusters, then, under most circumstances, at least C good clusters will be found from the data. This implies that the number of clusters need not be known a priori when using the PCM.

Another feature of the PCM is its robustness against noise and outliers. Noise points and outliers are often quite distant from the primary clusters, so according to the nature of the PCM, the further away a noise point is from the dense area, the smaller the memberships. Noise points and outliers will be assigned insignificant membership when using the PCM, which give them little influence on the estimation of the prototypes and final partition.

2.3.3 Automatic Merging Possibilistic Clustering Method (AMPCM)

Despite the PCM's mode-seeking property and its robustness towards noise and outliers, it still faces parameter-selection problems and initialization problems. Yang and Lai

created the automatic merging possibilistic clustering method (AMPCM) based on the PCM that manages to automatically find the number of clusters and solve parameter-selection problem [24].

In the AMPCM, Yang and Lai replaced the second term in the PCM's objective function (Equation 2.14) $\sum_{i=1}^{C} \eta_i \sum_{j=1}^{N} (1-u_{ij})^m$ with $\sum_{i=1}^{C} \gamma_i \sum_{j=1}^{N} u_{ij} \left(1-\mu_{ij}^{\frac{1}{k}}\right)$ and use mathematical manipulation to turn the objective function into

$$J_{AMPCM} = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij} d_{ij}^2 - \sum_{i=1}^{C} \gamma_i \sum_{j=1}^{N} u_{ij} \left(1 - \frac{k}{k+1} * \mu_{ij}^{\frac{1}{k}}\right), k > 1 \quad (2.18)$$

By differentiating the objective function with respect to the membership matrix and prototypes, we have the update equations as follows:

$$\boldsymbol{u_{ij}} = \left(\frac{\boldsymbol{\gamma_i - d_{ij}^2}}{\boldsymbol{\gamma_i}}\right)^{\boldsymbol{k}} \quad (2.19)$$

$$\boldsymbol{a_i} = \frac{\sum_{j=1}^{N} \boldsymbol{u_{ij} x_j}}{\sum_{j=1}^{N} \boldsymbol{u_{ij}}} \quad (2.20)$$

Since $u_{ij} \in [0,1]$, it is necessary to choose γ_i with $\gamma_i \geq \max_{1 \leq j \leq N} d_{ij}^2$. For simplicity, Yang and Lai consider $\gamma_i = \gamma = \max\{_{1 \leq i,j \leq N} d^2(x_i, x_j)\}$, which enables the membership between the two data points farthest apart to be 0 and the membership to be largest when the distance between two data points is 0.

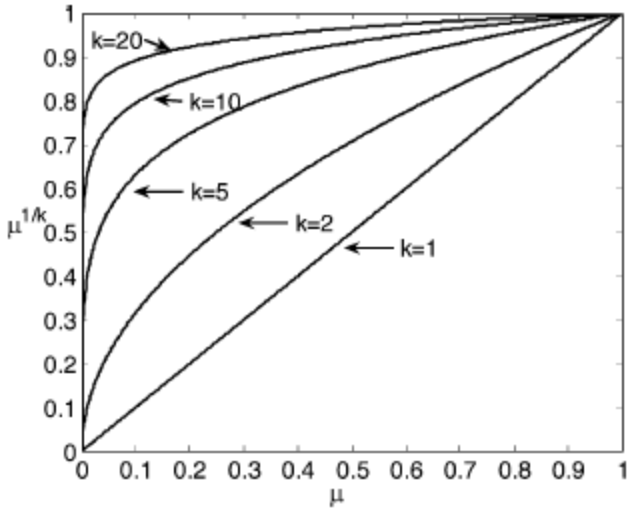


Figure 2.5 plot of functions μ(1/k) with k=1, 2, 5, 10, 20

As we can see in Figure 2.5, the parameter k is used as an impact factor of the second term in function equation (2.18). If k becomes large, the membership of $\boldsymbol{u_{ij}}$ of a data point $\boldsymbol{x_j}$ which is far away from the *i*th cluster center a_i will decrease quickly to achieve its robustness. If the value of k is known, then we can get the membership matrix and prototypes. Based on equation (2.19), we have

$$k = \frac{log(\mu_{ij})}{log((\gamma - d_{ij}^2)/\gamma} \tag{2.21}$$

Set $D = d_{ij}^2$ and $\mu_{ij} = ((\gamma - \mathrm{D})/\gamma)^k$ for given i and j. Then set $\beta = \mu_{ij}$ and have that $\beta = ((\gamma - \mathrm{D})/\gamma)^k$. Now the problem have been reduced to solving for β and D to determine the value of k

$$k = \frac{log(\beta)}{log((\gamma - D)/\gamma}. \tag{2.22}$$

Suppose that the ith cluster has n_i data points, then there will be $C_2^{n_i}$ distances between any two points in this cluster. Yang assumed that if the dataset, whose size is N, can be well separated into C clusters, then the range of the biggest cluster will be the ($C_2^{n_1} + C_2^{n_2} + \cdots + C_2^{n_C}$)/ C_2^N percentile of the C_2^N distances, if the C_2^N distances are ordered from the smallest to the largest . Let $\alpha_i = n_i/N$, we have

$$(C_2^{n_1} + C_2^{n_2} + \cdots + C_2^{n_C})/ C_2^N$$

$$= \frac{n_1^2+n_2^2+\cdots+n_C^2-n_1-n_2-\cdots-n_C}{N^2-N} = \frac{1}{N-1}[N(\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_N^2) - 1]. \quad (2.22)$$

Usually the value of $n_1, n_2, \ldots, n_C$ are unknown so that $\alpha_i = \sum_{j=1}^{N} \mu_{ij} / \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}$ is used to estimate the value of α_i. Yang et al. explained that if the biggest cluster can be well classified into a cluster under some given value of β and D, the other smaller clusters will also be well classified into their respective clusters. Hence D is chosen by the C_2^N square distances of $\Omega = \{d^2(x_1, x_2), \ldots, d^2(x_1, x_N), d^2(x_2, x_3), \ldots, d^2(x_{N-1}, x_N)\}$.

Let Δ= sort$\{\Omega\} = \{D_{(1)}, D_{(2)}, \ldots, D_{(N\cdot(N-1)/2)}\}$, and

$$q = (C_2^{n_1} + C_2^{n_2} + \cdots + C_2^{n_C})/ C_2^N$$

$$= \frac{1}{N-1}[N(\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_N^2) - 1].$$

$$= \frac{1}{N-1}[N * \sum_{i=1}^{C}(\sum_{j=1}^{N} \frac{u_{ij}}{\sum_{i=1}^{C}\sum_{j=1}^{N} u_{lj}})^2 - 1] \quad (2.23)$$

If the dataset can be well separated into C clusters, then the range of the biggest cluster should be $D_{(((N-1)\cdot N/2)\cdot q)}$. This can be used to estimate the value of β and D. We know

that if $d_{ij}^2 = D$, then the membership $\mu_{ij} = \beta$. If the distance $d_{ij}^2 < D$, then the membership will be quickly increasing to 1.

According to equation (2.21), we can see that β becomes smaller when distance D becomes bigger. Thus, for a fixed p>1,

$$\beta = \frac{p-1}{p}$$

$$D = \frac{D_{(((N-1)\cdot N/2)\cdot q)}}{p^2}. \tag{2.24}$$

The parameter p controls the relationship between the membership and distance, which Yang and Lai suggested that it be between 3 and 4.

Combining equation (2.21) and (2.24), we can get k's value by using the equation

$$k = \frac{log(1-(1/p))}{log(1-(D_{((N*(N-1)/2)*q)})p^2\gamma)} \tag{2.25}$$

To avoid the bad initialization of the number of clusters, Yang also suggested that all data points are used as initial cluster centers. Let the partition matrix be $U = [\mu_{ij}]_{CXN} = [\overrightarrow{u_1}, \dots, \overrightarrow{u_C}]^T$ and the correlation coefficient matrix be $r^2 = [r_{il}^2]_{CXC}$. Clusters with close cluster centers are merged according to the correlation of the membership matrix

$$\boldsymbol{r_{il}^2 = \frac{\overrightarrow{u_i}^T \overrightarrow{u_l}}{\|\overrightarrow{u_i}\| \|\overrightarrow{u_l}\|}}. \tag{2.26}$$

Let ρ be the threshold for correlation coefficient. If $r_{il}^2 > \rho$, then the *i*th and *l*th clusters will be merged into one.

Let $\boldsymbol{R} = \{\boldsymbol{R_1}, \boldsymbol{R_2}, \dots, \boldsymbol{R_C}\}$, where

$$R_i = \sum_{l=1}^{c} r_{il}^2. \tag{2.27}$$

The parameter R_i is used to represent the density around cluster center c_i. If R_i is large, then c_i will have priority over other prototypes with smaller R_i to merge with the correlated cluster centers.

Let $P = \{P_1, P_2, \dots P_C\}$ be the partition of the dataset, where $|P_1| + |P_2| + \cdots + |P_C| = N$. Because all data points are used as initial cluster centers to avoid bad initialization, the cluster number C is equal to the sample size N. Thus, we have $P = \{P_1, P_2, \dots P_N\}$ and $P_i = \{x_i\}, i = 1,2, \dots, N$. The cluster center with the largest density is first merged with those cluster centers being very close to it into one cluster. Then the largest density in the remaining cluster centers is merged with centers that are very close to it. The merging step is continued until all cluster centers have been merged once.

$$E_m = \{l: g = argmax_{i \in G}(R_i) \ , r_{gl}^2 > \rho, l \in G\}. \tag{2.28}$$

Here $G = \{1,2, \dots, C\} \backslash (E_1 \cup E_2 \cup \dots \cup E_{m-1})$ is the set of clusters that have yet to be merged and $m = 1,2, \dots, \mathrm{C}^{new}$.

The new partition is gained and the update of the membership and prototypes are as followed:

$$P_m^{new} = \bigcup_{l \in E_m} P_l \tag{2.29}$$

$$\mu_{mj}^{new} = \frac{\sum_{l \in E_m} u_{lj}}{|E_m|}, 1 \leq m \leq C^{new} \tag{2.30}$$

$$a_{mj}^{new} = \frac{\sum_{l \in E_m} a_l}{|E_m|}, 1 \leq m \leq C^{new} \tag{2.31}$$

The merging algorithm is summarized as follows.

Pseudo Code for AMPCM Algorithm:

Set m = 1, G = {1, 2, … , C}.

Step 1: Compute r^2 with U using (2.26).

Step 2: Compute R with r^2 using (2.27).

Step 3: Compute E_m with R using (2.28).

Step 4: Set $G = G \setminus E_m$.

If | G | > 0: set m = m + 1, and return to Step 3;

Else: set C^{new} = m.

Step 5: Compute P^{new}, U^{new}, A^{new} using (2.29), (2.30), (2.31), and then stop.

The AMPCM algorithm is summarized as follows.

Pseudo Code for AMPCM Algorithm:

Fix the initial $D^{(0)}$, $p \in [3,4]$, ρ and ε.

Set $t = 0$, $A^{(0)} = X$, $C = N$, $P = \{ P_1, P_2, \ldots, P_N\}$, where $P_i = \{x_i\}, i = 1,2, \ldots, N$; Compute

$\gamma = \max\{_{1 \le i,j \le N}\, d^2(x_i, x_j)\}$ and $\Delta = \text{sort}\{\Omega\} = \{D_{(1)}, D_{(2)}, \ldots, D_{(N \cdot N - 1/2)}\}$.

Step 1: Compute k with $D^{(t)}$ by (2.25).

Step 2: Compute U with k by (2.19).

Step 3: Update $A^{(t+1)}$ by (2.20).

Step 4: If $\left\| A^{(t+1)} - A^{(t)} \right\| < \varepsilon$: stop.

Step 5: Call Merging algorithm to obtain $P = P^{new}, A^{(t+1)} = A^{new}, and\ U = U^{new}$.

Step 6: Compute q with U by (2.23).

Step 7: Compute $D^{(t+1)}$ with q by (2.24).

If $D^{(t+1)} < D^{(t)}$: $D^{(t+1)} = D^{(t)}$.

Set t = t + 1 and return to step 1.

Note that $D^{(0)}$ is the only initial in the AMPCM that might affect the merging procedure. If $D^{(0)}$ is chosen too small, the algorithm's merging strength will be too weak so that more merging steps will be needed; if $D^{(0)}$ is chosen too large, its merging strength will be too strong that the final number of clusters will become less than what might be the natural number of clusters. Since it is better to choose the initial $D^{(0)}$ with a small value than a large value, it is recommended by Yang and Lai that the initial $D^{(0)}$ to be smaller than $D_{(((N-1)\cdot N/2)/\sqrt{N})}$. Thus, a rule for the initial $D^{(0)}$ is recommended with

$$D^{(0)} = \frac{D_{(((N-1)\cdot N/2)/\sqrt{N})}}{2}. \tag{2.32}$$

Although using all the data points as initial cluster centers makes the AMPCM computationally very expensive, the AMPCM actually manage to automatically determine the "optimal" number of clusters without specifying any kind of parameter beforehand. Inheriting the good feature of the PCM, the AMPCM is also robust to different cluster volumes, different shapes without producing coincident clusters. Note that the AMPCM may be able to find the optimal number of clusters, but it will create extra separate clusters for the noise away from the dense regions of the dataset, which might not be desirable.

3 Methodology

In order to generate more meaningful alerts to show signs of health changes, temporal information of the data is put into consideration and a multi-dimensional alert is utilized in hopes of a decrease in the numbers of false alarms and to explore more potential alerts. The goal is to capture the temporal trajectories to predict and detect health changes, accommodate gradual changes, and separate outliers with models that adapt to an elder's pattern and ailments.

Since the Gaussian mixture model has this reliable ability to accommodate lighting changes, slow moving objects, and repetitive motions from clutter when it is applied to video background modeling [20], it is proven to be a powerful tool to model the normal behavior of an elder and to create predictive measures for signaling the onset of health changes.

Feature vectors from the initial time window are recorded to model the initial normal behavior using the GMM model. When a new input vector x_t comes in at time t, it will be tested against each Gaussian that comprises target's behavior pattern. If the distance of the vector to the mean of a certain Gaussian representing normal behavior is within a specified number of standard deviations, then the corresponding time interval is considered as a normal pattern, and the parameter and the weight of the winning Gaussian is updated. If the distance of the vector is not within the specified number of standard deviations to any of the existing Gaussian components, then the corresponding time interval will be marked as an anomaly and an alert will be raised.

Unlike the case of video background modeling, the abnormal pattern will not be automatically incorporated into the normal activity model at the first alert to replace the least probable Gaussian distribution. There could be a lot of reasons that might cause this abnormal pattern. It could represent system noise like sensor failure, visitor activity, an extended absence, or it could indicate a real change in the resident's health condition. It would be unreasonable to create a new Gaussian distribution for system noise. We assume that what appear to be anomalies for a short time may actually correspond to a new normal component and those patterns, unlike system noise, will form a cluster. So when those anomalies form a cluster with no clinical input indicating a health problem, a new cluster will be added into the Gaussian mixture model.

3.1 Initialization of the Gaussian Mixture Model

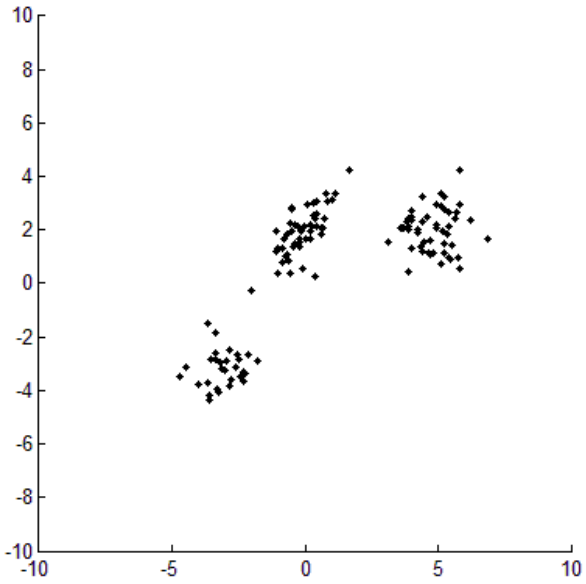
It is suggested in Stauffer's paper [20] to use a fixed number of Gaussian distributions, K, to model the video background; K is usually 3-5. But the number of Gaussians in the embedded sensor environment is different from video background modeling. Since every resident has their unique behavior pattern, the parameter K needs to be studied for each individual. We want the right number of Gaussians to generalize a normal behavior pattern: we do not want to use too many Gaussians to overfit the normal and lose generalization; we do not want to use too few Gaussians either to lose the flexibility to describe a normal state. So the problem is how to find out the right number of Gaussians to initialize the model.

The typicality-based memberships of the PCM make it robust to the effect of noise points and outliers. The mode-seeking property of the PCM makes it possible to find the k natural clusters in the dataset if we overestimate the number of clusters C>k when initializing the algorithm. Some of the m clusters will coincide with others but the number of the non-coincided clusters will be equal to k. It seems that the PCM might be the right algorithm to help us find the right number of Gaussians to initialize the normal pattern model if we manage to overestimate the number of clusters in the first place.

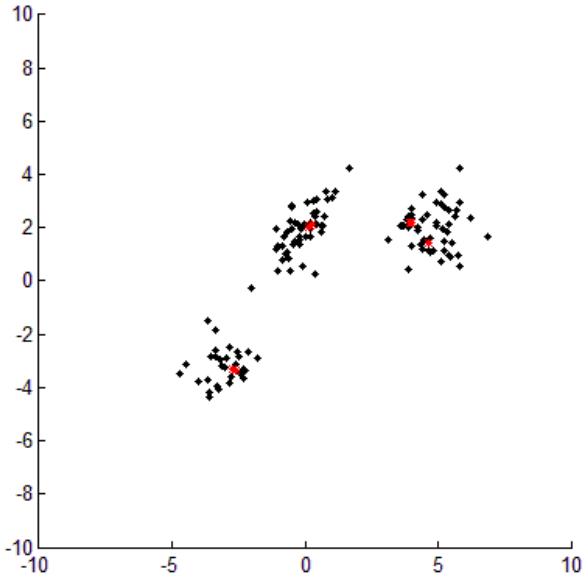
Yet the PCM is faced with initialization and parameter-selection problems. In the PCM, the prototypes are automatically attracted to dense regions in the feature space. This requires the PCM to initialize the prototypes so that each prototype will be converge to k-distinct local minima (if in fact there are C distinct clusters).Using the FCM for initialization of the prototypes is a good way to solve the initialization problem. As for the parameter selection problem, we apply the PCM on a dataset which consists of three separated clusters, as shown in Figure 3.1(a). We first consider the fuzzifier m=1.5, as recommended by Krishnapuram and Keller in [19], and overestimate the number of clusters C=10. The PCM clustering result shows that there are some clusters that become coincident clusters but are not degenerated enough. As we can see in Figure 3.1(b), some of the cluster centers are attracted to the left most cluster yet converge to two local minima. However, if a more suitable parameter of fuzzifier m is chosen, as shown in Figure 3.1(c), the algorithm seems to find the right number of clusters as all the cluster centers are attracted to three distinct dense regions of the feature space. But those coincident clusters raise a new problem of merging. We next use Figure 3.2 to further demonstrate this phenomenon. We set the number of clusters to 12 for the dataset shown

in Figure 3.2(a), which consists of 9 natural clusters. The PCM gives a good result when the fuzzifier m=1.5, as shown in Figure 3.2(b), where the algorithm degenerate some of the cluster centers to be coincident clusters such that only nine separated cluster clusters are obtained. When we adjust the fuzzifier m=2, the clustering results degenerate too much, in that one cluster is missing. Note that in both experiments, the PCM is initialized with the same prototypes. We can see that the PCM is very sensitive to the value of fuzzifier m.

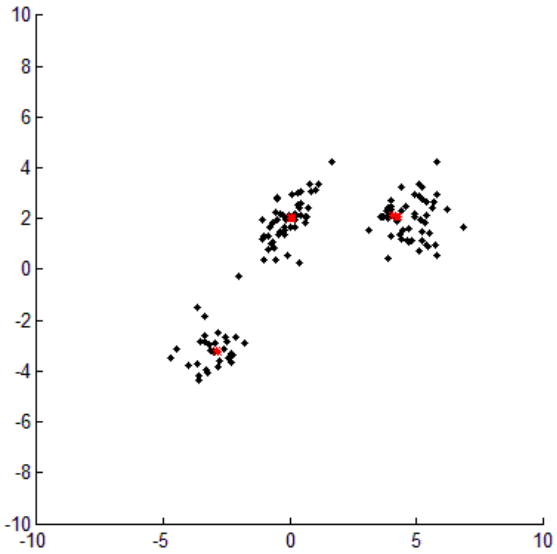
What's more, if the predefined number of clusters C happens to be less than the number of natural clusters k, then only C good clusters can be found in the data. This unique feature of the PCM, unlike most clustering algorithms which divide all the patterns into C clusters, enables the algorithm to find only C dense regions out of the k dense regions based on the typicalities while the rest of the dense regions have insignificant typicalities towards the existing C clusters. Yet this mode-seeking propoerty still will not be good enough to describe the existing normal pattern. Thus using the PCM algorithm alone will not be well-suited to initialize the GMM model.



(a)

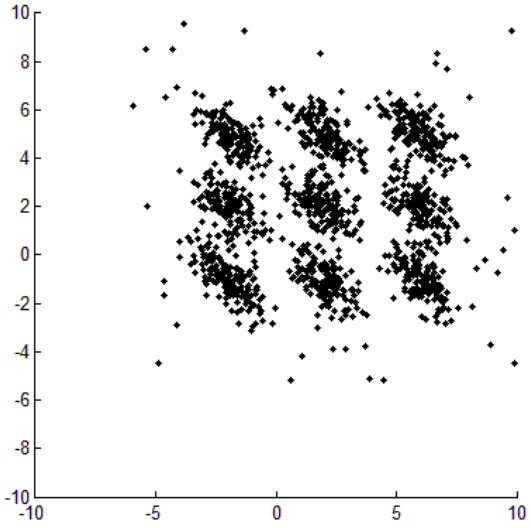


(b)

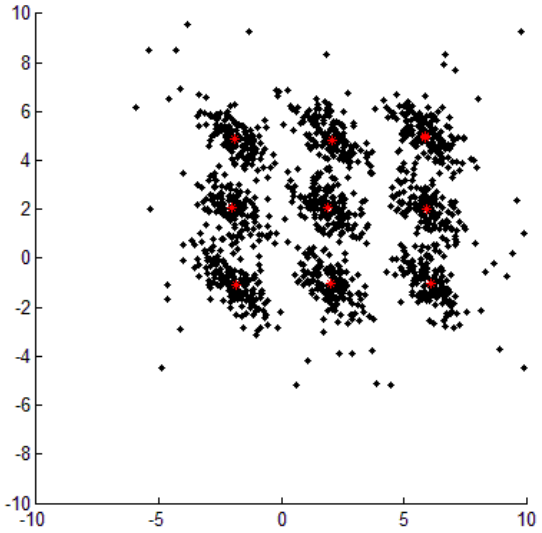


(c)

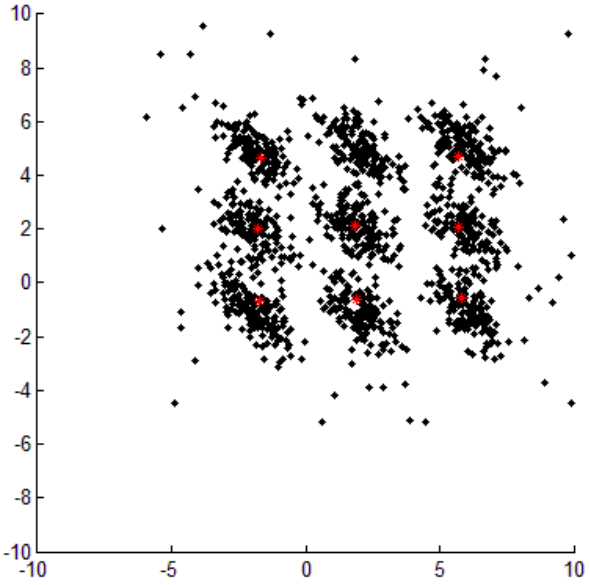
Figure 3.1 (a) original dataset. (b) PCM clustering result of data (a) with the number of clusters set as 10, and fuzzifier m=1.5. (c) PCM clustering result of data (a) with the number of clusters set as 10, and fuzzifier m=2



(a)



(b)



(c)

Figure 3.2 (a) original dataset. (b) PCM clustering result of data (a) with the number of clusters set as 12, and fuzzifier m=1.5. (c) (b) PCM clustering result of data (a) with the number of clusters set as 12, and fuzzifier m=2

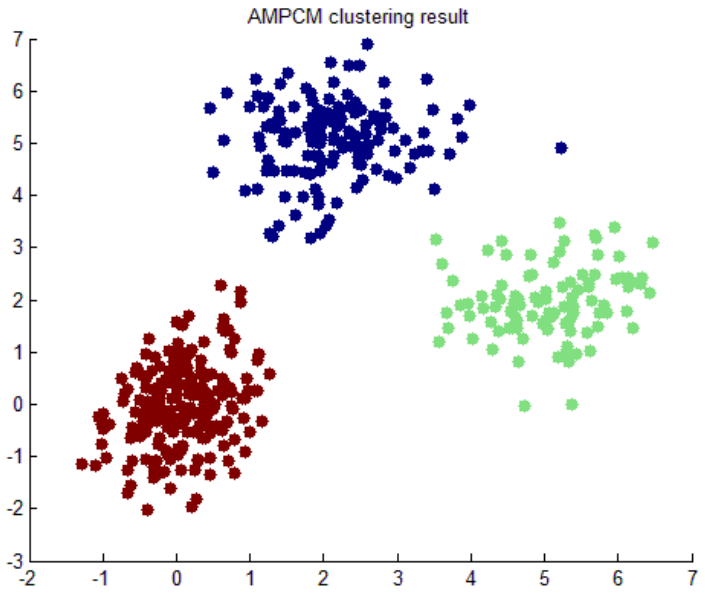


Figure 3.3 AMPCM clustering result of a dataset with no need to specify the number of clusters beforehand, in which the natural number of cluster 3 is found correctly.

The AMPCM, on the other hand, avoids the initialization and parameter selection problems and is able to automatically find the optimal number of clusters despite its computational complexity. Figure 3.3 is one demonstration of the AMPCM's robustness to initial values and number of clusters. The input to the AMPCM is the dataset itself, and with the parameters set as the algorithm's default ($p=3$, $\rho=0.9$), the natural number of clusters 3 is found automatically.

But computational complexity is not the only problem with the AMPCM. When there is noise present, it may produce separate clusters which may only have few elements for those noisy vectors away from the dense regions of the data. As shown in Figure 3.4, the AMPCM spawns four extra clusters for the noisy data. The membership values of those noisy data, like the typicalities of the PCM, are insignificant to all the other clusters. But

the noisy data's memberships toward their own clusters are still high, which makes it problematic to distinguish those clusters from those natural clusters of the data.

The AMPCM creates those separate clusters for noisy vectors because the algorithm starts with every feature vector as their own cluster centers, and since it merges the clusters by finding the most correlated clusters using membership matrix, those separate noisy clusters will stay separated. This is not desirable when we are trying to use the algorithm to find out how many components we should use for the Gaussian mixture to describe the normal behavior pattern. Incorporating the Gaussians represented by noise points is not only costly to the system memory; what's more important, it will ruin the representation of the true normal pattern and cause the system to miss important alerts. It's important to distinguish those noisy clusters from the natural clusters. Since noise might indicate changes in health condition or changes in behavior pattern, all the outliers from the initial time window should be kept in an anomaly log for future references.

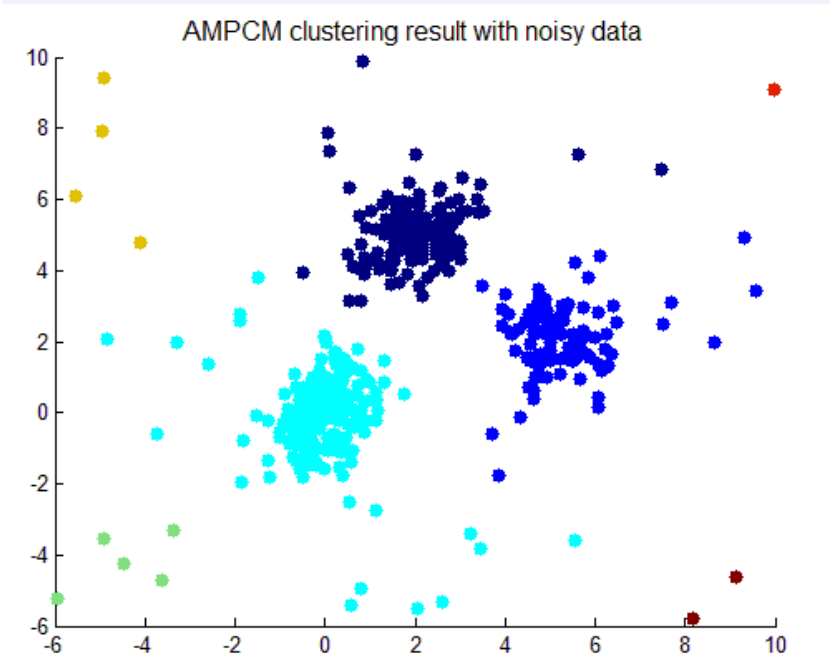


Figure 3.4 AMPCM clustering result with noisy data present: In addition to the three natural clusters, the AMPCM creates 4 extra clusters for the noises in the corners of the graph

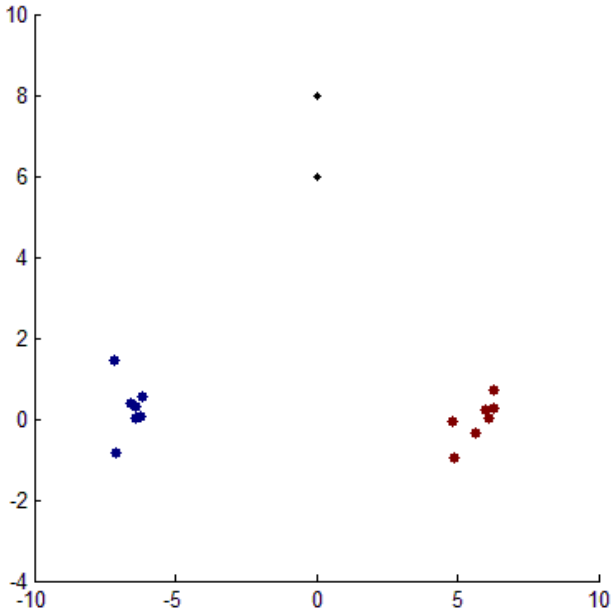


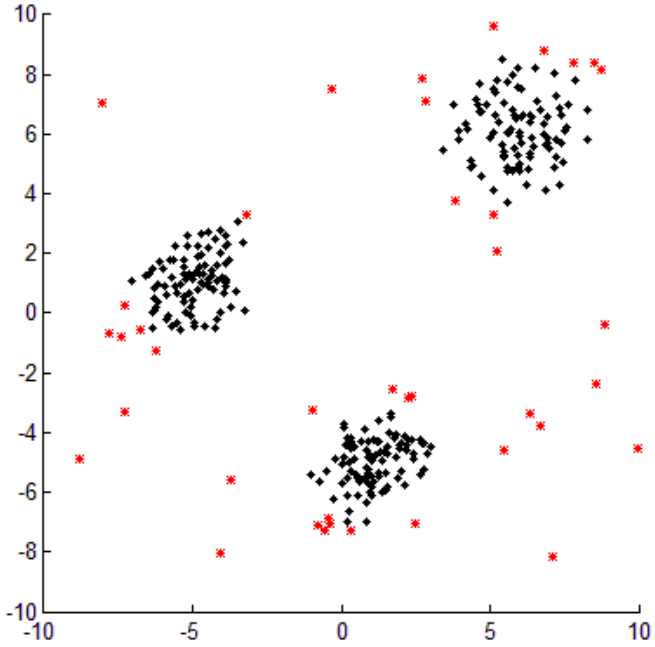
Figure 3.5 Clustering result on a simple dataset using the PCM with the number of clusters set as 2

The PCM, in this point of view is more robust against noisy data. Noise points and outliers are often quite distant from the primary clusters, so according to the nature of the PCM, the farther away the noise point is from the dense areas, the smaller the membership value, no matter how the number of clusters is assigned. Table 3.1 shows the membership values of each data point when using the PCM clustering for the dataset in Figure 3.5, we can see that the noisy data (data point 4 and data point 13) have very low memberships to both clusters, and the farther point has lower membership than the closer one.

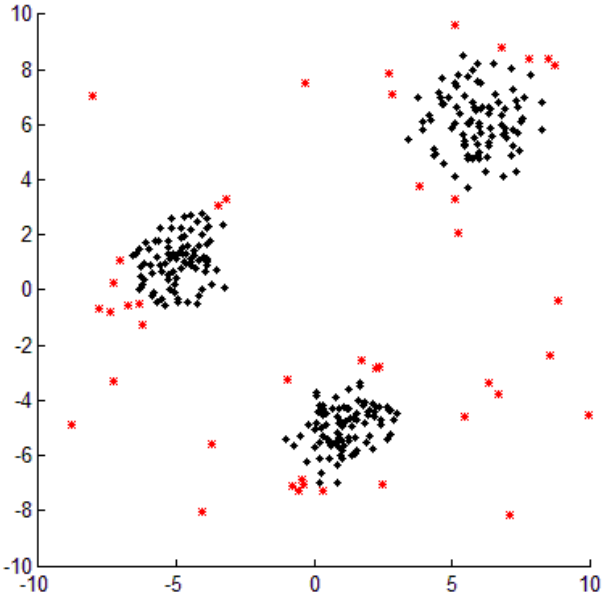
To further demonstrate that the PCM will assign low membership value to the noise and outliers far away from the dense regions of the data, we cluster a noisy dataset with different number of clusters setting; and if the data point's memberships to all the clusters are lower than a threshold $\mathbf{T_n}$, it will be recognized as an "outlier" and shown in red in Figure 3.6. Set the membership threshold $\mathbf{T_n}$ to 0.05 and cluster the data into 3 (Figure 3.6a), 5 (Figure 3.6b), 10 (Figure 3.6c) clusters, the outliers the PCM recognizes are identical. But if setting of the number of clusters is lower than the natural clusters number of the data, the PCM will recognize some of the natural clusters as outliers, as shown in Figure 3.6(d). As long as we overestimate the number of clusters, noise and outliers distant from the dense areas of the data will always have insignificant memberships to all the clusters. But specifying the number of cluster should not be an issue even if we set the number of clusters lower than the natural number of clusters. More details will be given in Section 3.3.1.

Table 3.1 Membership values and centers resulting from the PCM clustering for the noisy data set shown in Figure 3.5

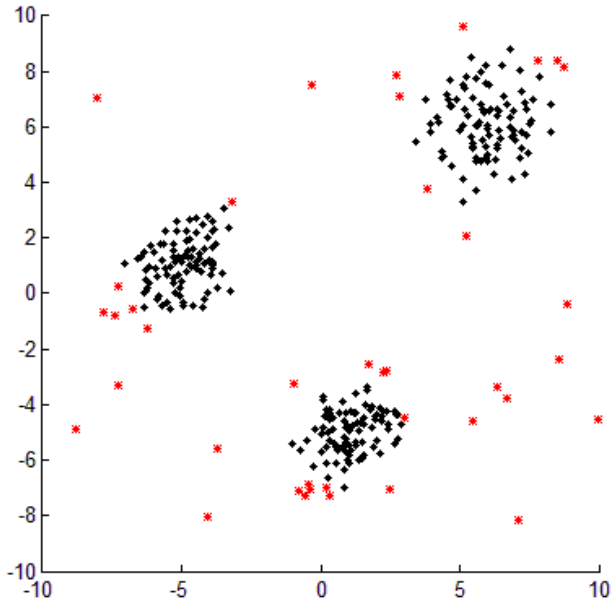
Data point	Cluster 1	Cluster 2
1	0.0094	0.9433
2	0.0103	0.4582
3	0.0095	0.9397
4	0.0177	0.0252
5	0.4636	0.0111
6	0.9577	0.0138
7	0.0101	0.9370
8	0.5407	0.0113
9	0.9244	0.0135
10	0.9189	0.0140
11	0.4150	0.0141
12	0.0089	0.7820
13	0.0126	0.0176
14	0.3544	0.0104
15	0.0075	0.2838
16	0.0084	0.3135
Cluster Center	(-5.70, 0.34)	(5.32, 0.10)



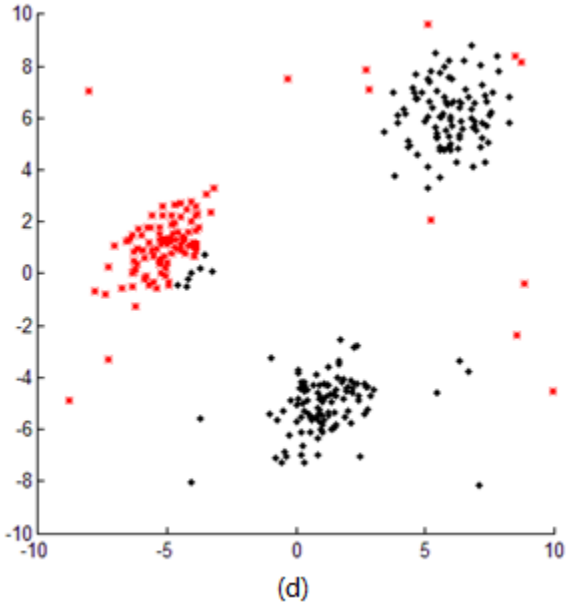
(a)



(b)



(c)

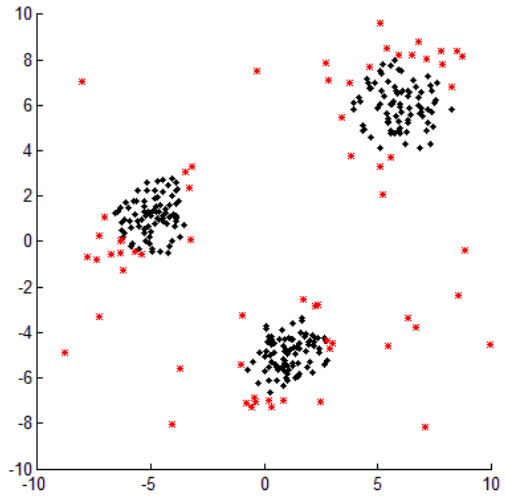


(d)

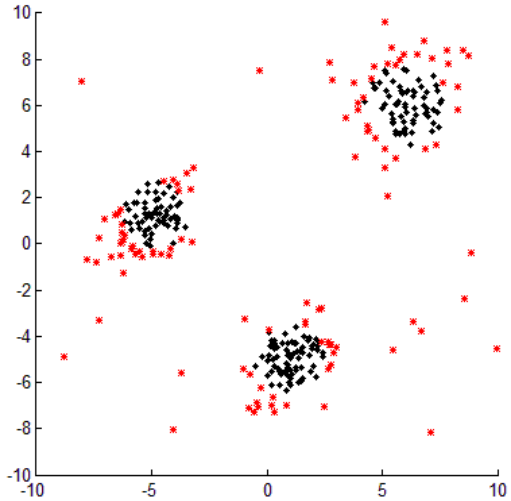
Figure 3.6 Effect of the number of clusters on the detection of noise: (a) PCM clustering result with the number of clusters set as 3, (b) PCM clustering result with the number of clusters set as 5, (c) PCM clustering result with number of clusters set as 10, (d) PCM clustering result with the number of clusters set as 2. In those graphs, the data points with insignificant membership to all the clusters are marked as red points.

Figure 3.7 and Figure 3.6(b) show the effect of the membership threshold on the recognition of the outliers. With the number of clusters set as 5, and the membership threshold $\mathbf{T_n}$ set as 0.05 (Figure 3.6b), 0.10 (Figure 3.7a), 0.20 (Figure 3.7b), we can see that the larger the value of $\mathbf{T_n}$, the more feature vectors that belong to the natural clusters will be recognized as noise. The membership threshold $\mathbf{T_n}$ is recommended to be chosen between 0.05 and 0.1.

Judging from the above, by combining the PCM's robustness to noise together with the AMPCM's ability to find the natural number of clusters with no initialization issue and parameter selection problem, we can initialize the Gaussian mixture model with the optimal number of components. To do this, the PCM is first used to "filter" out the noise in the data, i.e. the data points far away from the dense regions of the data. This can be simply done by clustering the feature vectors in the initial time window, find the feature vectors whose membership to all the clusters are insignificant, and mark them as outliers. As mentioned before, the PCM is able to recognize noise given that the number of clusters is set larger or equal to the natural number of clusters.



(a)



(b)

Figure 3.7 Effect of the membership threshold on the detection of noise (a) membership threshold is set as 0.10 and the number of clusters is set as 5. (b) membership threshold is set as 0.20 and the number of clusters as 5. The outliers are marked in red color.

After filtering out the noise, AMPCM is used to cluster those "noise-free" data with no need to specify the number of clusters beforehand. The clustering result will be used to build the Gaussian Mixture Model. The data points with the same label will be grouped together to calculate the mean and covariance of their Gaussian component (see Figure 3.8). Unlike Stauffer's case, we don't need to worry about computational complexity to use the form $\Sigma = \sigma^2 I$ for covariance matrices to enable the real-time tracking. Since our input data usually comes in multiples of hours, the computational complexity will not be an issue and computation for the full covariance model is possible. The weight of ith Gaussian component is ratio of the number of points labeled as i to the total number of data points.

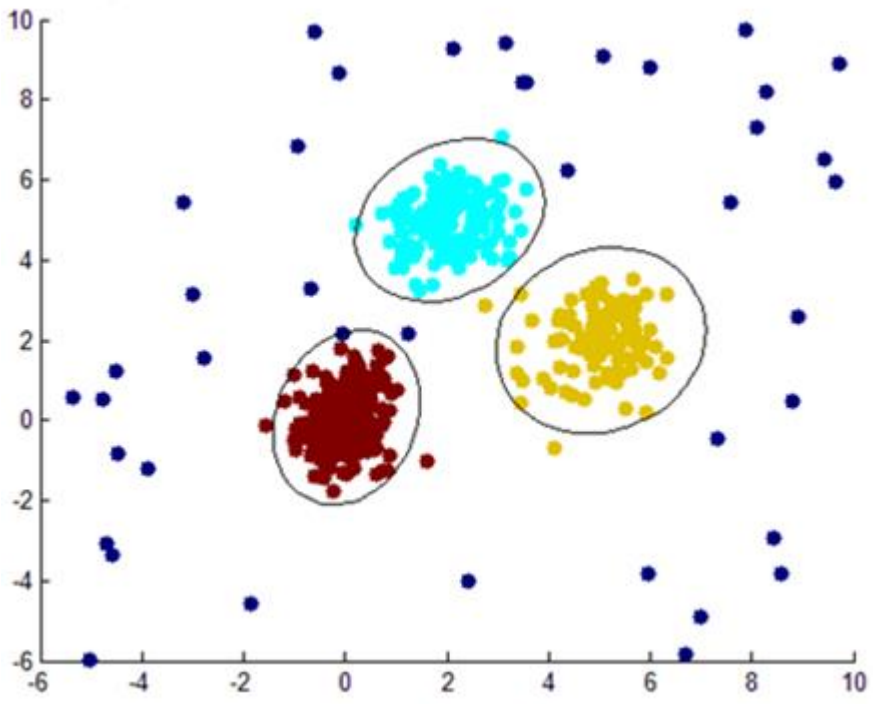


Figure 3.8 An ideal combination of the PCM and the AMPCM. The outliers are filtered using the PCM and the rest of the data is clustered using the AMPCM, the crisp partition of which is used to create the Gaussian Mixture Model.

3.2 Updating the Normal Pattern Model and Generating the Alert

A new input vector x_t at time interval t is tested against each Gaussian component comprising normal behavior. The Mahalanobis distance of the input vector to each the Gaussian components is calculated, and if the new data entrance is closest to certain Gaussian and the distance happens to fall into the pre-specified range, then this instance will be incorporated into that Gaussian; if not, then this instance will be flagged as an anomaly.

According to the equation (2.3), the loci of points of constant density are hyperellipsoids for which the quadratic form $(x-\mu)^T\Sigma^{-1}(x-\mu)$ is constant. The Mahalanobis distance is defined by equation:

$$\boldsymbol{d_{mahalanobis} = \sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}} \quad (3.1)$$

The Mahalanobis distance is a scale measure of the scatter of the samples about the mean μ. It is not only decided by the distance the point is to the mean of the distribution, but also the direction of the distribution, which is described by the covariance matrix Σ. Assume two points have the same Mahalanobis distance to the distribution, but they may have different Euclidean Distance to the mean of the distribution. Given the same mahalanobis distance, the point in the direction of the shorter eigenvector of the covariance matrix must be closer to the mean in Euclidean distance, while the point in the direction of the larger eigenvector of the covariance matrix must be farther away from the center of the distribution, as illustrated by point A and point B in Figure 3.9.

If the minimum distance of the vector to all the distributions is within a specified number of standard deviations, T_d, then that time interval is considered as part of the normal

activity pattern. The parameters associated with the winning Gaussian are then updated by x_t. Again, unlike Stauffer's real-time-sensitive system, we have enough time to update the parameters using all the data associated with the given distribution.

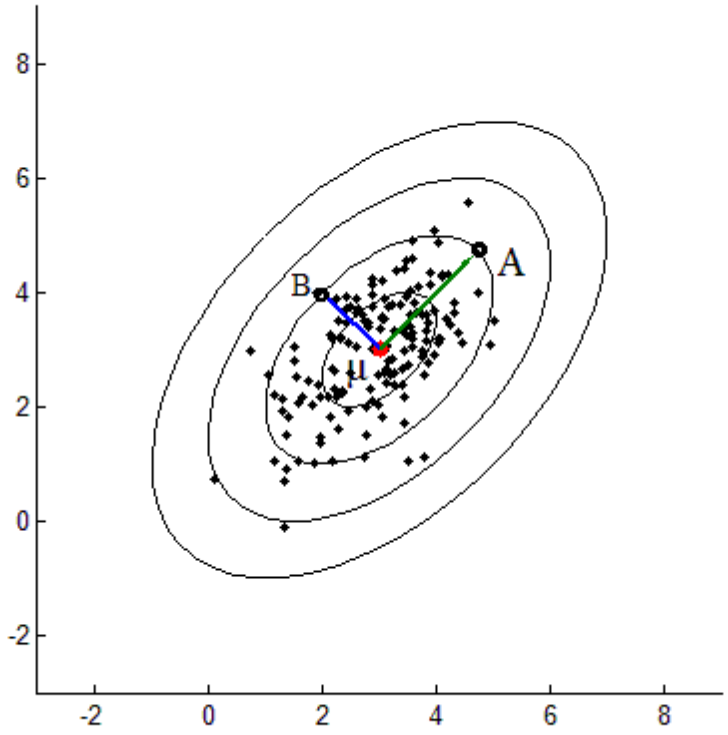


Figure 3.9 Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean μ. The ellipses show lines of equal mahalanobis distance to the center.

If the new input vector x_t does not match any of the Gaussian components, then an alert is generated. The alert index will be added into the anomaly log that records all the alert history of the elder for future references. Sometimes outliers do not necessarily indicate a health change of an elder. It may suggest a change of lifestyle, for example, starting to exercise regularly, or drinking more water to induce more frequent bathroom visits. The elder may still be healthy but due to his change of life pattern, the feature vector may

deviate from its normal. So if enough activity vectors follow the same pattern with no clinician input indicating a health problem, a new distribution will be created to describe the new normal pattern and be incorporated into the normal behavior model. Figure 3.10 illustrates the situation where a potential pattern emerges from the outliers.

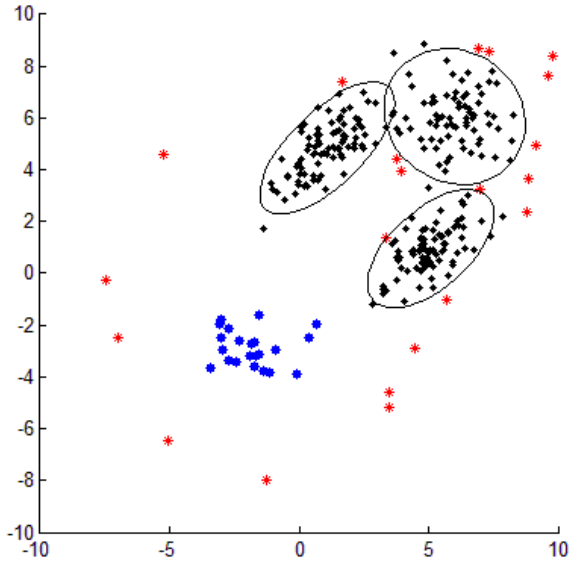


Figure 3.10 . Both red and blue data points are anomalies with blue data points showing a tendency to form a cluster. If the clinical experts determine that those anomalies are false, a new Gaussian component will be created for the blue data points to incorporate into the GMM model.

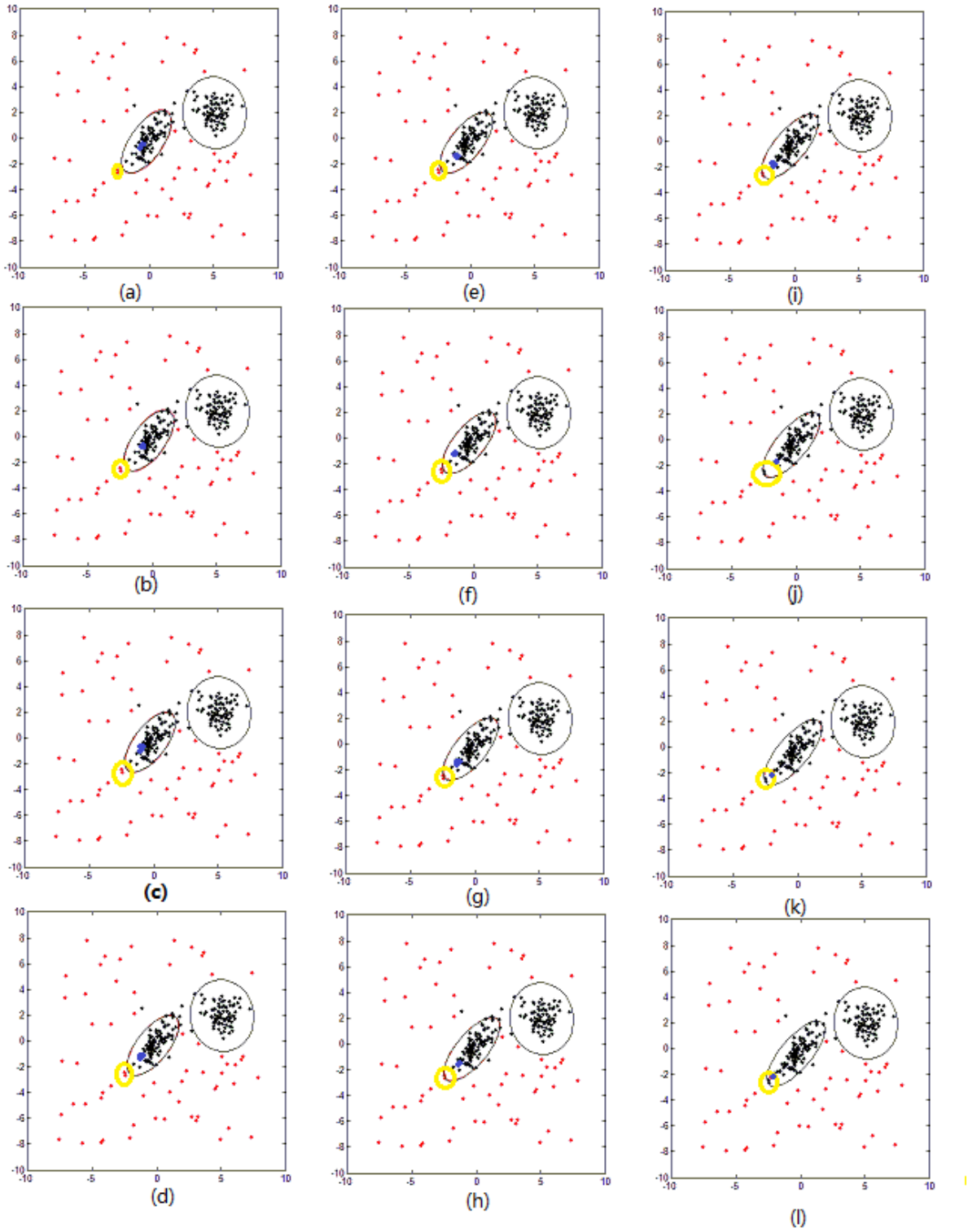


Figure 3.11 Highlighted anomalies fall into the ellipsoid cloud after several updates on the Gaussian component. (a)-(j): The new data inputs keep updating the left Gaussian ellipsoid and elongate its boundary. (k)-(l): The highlighted anomalies finally fall into the boundary and are incorporated into the normal behavior pattern.

Due to randomness of the daily behavior, using a threshold T_d to determine if the vector is an anomaly may be biased. Depending on the setting of the threshold T_d, a feature vector representing normal activity pattern may linger outside the border of the hyper-ellipsoid cloud, thus being flagged as an anomaly, but after enough updates of that Gaussian component, the flagged vector might fall into the hyperellipsoid cloud again. It is suggested to keep checking the alert history with the existing normal distributions. If some of the alerts are within T_d standard deviations to certain Gaussians after several updates to the Gaussian mixture model and the clinical experts suggest irrelevance to health change, they will be labeled as the winning Gaussian and the parameters of the winning Gaussian will be updated. Thus a better picture of the normal behavior pattern can be learned. As shown in Figure 3.11, points in black represent normal behavior, points in red indicate flagged activity while the blue point is the new data input. We can see that as the new data input keep pushing towards the boundary (the ellipsoid), the eigenvector in the corresponding direction get elongated to accommodate the changes. Thus the highlighted flagged activity fall into the ellipsoid and be labelled as normal.

3.3 Introducing a New Normal Distribution into the Normal Behavior Model

3.3.1 Discovering a potential new behavior pattern

Some of the flagged vectors may be away from the current normal activity pattern due to a change of lifestyle instead of real health change (see Figure 3.10). So after those alerts are determined by medical professions to be false alarms, they can actually be grouped

together to form a new Gaussian component to represent the new normal activity pattern. But finding a cluster from the false alarm records raises a new clustering problem. How to discover a good cluster out of a bunch of "outliers"? And what kind of standard qualifies the cluster as a good one?

First of all, how to find the clusters from the alert records? Again, the PCM can be a decent tool in this case: its robustness to noise and its mode seeking property makes it possible to find C dense regions of the alert log data (C as the number of clusters). In particular, if C is chosen to be one, then the prototype will converge to one dense part of the data, as shown in Figure 3.12. The dataset has three natural clusters and is clustered using the PCM with the number of the clusters set as 1. The prototype in Figure 3.12 is marked by 'X' and converges to one of the three clusters. All the data points belong to that cluster (marked as green) have high membership value while the rest of the data points (marked as red) have insignificant membership to the cluster. Thus using a membership threshold $\mathbf{T_o}$**,** the PCM algorithm can distinguish the dense region from the data.

Since examining the alert records for a new normal pattern is actively conducted every time a new alert is generated, if there is a cluster formed in the alert records, using the PCM with the number of clusters set as 1 will be good enough to find the cluster. If there is a legitimate cluster with no clinical input showing a health change, it will be treated as a new normal pattern and a new Gaussian component will be spawned for the GMM model. Even if the records happens to show that there are two or more "new normal patterns" emerging, which is most unlikely, the PCM mechanism will still manage to find one good cluster at a time. So defining the number of clusters to be 1 is good enough.

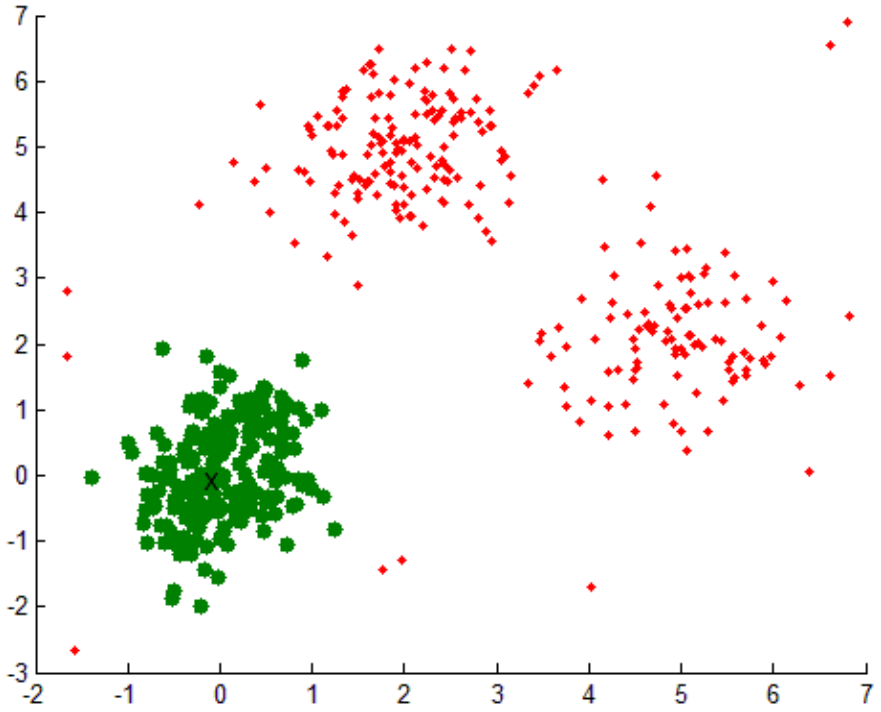


Figure 3.12 PCM's ability to find a single new cluster when setting the input number of clusters to one and using the membership threshold to manifest one of the dense regions of the data.

This procedure also compensates for the problem of specifying the number of clusters for the PCM in Section 3.1. Recall that in section 3.1, we use the PCM to filter out noise, but if we set the number of clusters lower than the natural cluster number, some of those natural clusters will be filtered out as noise and be logged into the anomaly records. When there is a new alert and the system examines the anomaly log for emerging new behavior pattern, the 'misunderstood' cluster will be rediscovered and incorporated into the normal. Thus the system is robust to the number of clusters setting of the PCM in the initialization process.

3.3.2 Cluster validation for the potential new normal behavior pattern

Now that we have a potential cluster from the PCM clustering, how do we decide if it is a good one or not? Note that the PCM will find a group of vectors as a cluster even if there is no compact cluster formed in the data (see Figure 3.13). So a good cluster validation method is crucial in recognizing the new pattern.

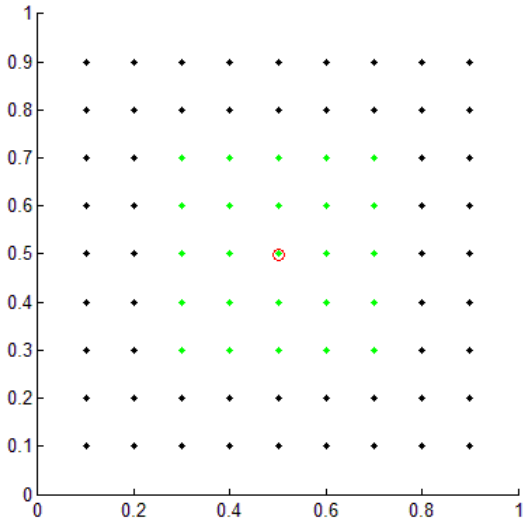


Figure 3.13 The PCM clustering result of a uniformly distributed dataset

Let cluster $X=\{x_1, x_2,\ldots, x_n\}$, the cluster validity measure we use here is a simple dispersion measure:

$$\boldsymbol{dispersion\ measure} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2}}{n} \tag{3.2}$$

The parameter n is the number of elements in the cluster, and μ is the mean vector of the cluster. The dispersion measure is a ratio of the scatter within the cluster to the size of the

cluster. The more compact of the cluster and the more elements in the cluster, the lower the value of the dispersion measure.

But the validity measure will only give us a single value suggesting the compactness of the new possible normal pattern, without any context of how confident it is that the new cluster is a real good one. To get more information about how well-formed the new potential cluster is, the dispersion measure of the existing Gaussian clouds are calculated to provide reference for the new cluster. In this system, the new possible cluster is compared to the least compact existing Gaussian components to determine if it is qualified. Tracking the compactness of the growing new cluster and determining the settling point is also a good idea to increase the confidence. Plus, in real life, whether or not this cluster could be a new normal pattern needs medical professionals' confirmation. We can send the clinical staff at TigerPlace the information of the possible new behavior pattern as well as a PCA (Principle Component Analysis) reduction plot of the data, so a better judgment can be made.

So the overall procedure and pseudo code for the whole process is as follows:

1. Cluster the existing data using PCM. Find the data points whose membership to any cluster is insignificant; mark those as outliers and keep them in the anomaly log.

2. Use the data which is not marked as outlier as input to the AMPCM. The AMPCM does not need the number of clusters as input and will output the natural number of clusters based on the correlation of the data.

3. Using the result we get from AMPCM, gather the data points with the same possibilistic labels and calculate their mean and covariance for their corresponding Gaussian component.

4. With each entrance of the new incoming data, the mahalanobis distance is calculated to determine the distance to each Gaussian component. If its minimum distance toward the Gaussian Mixture is within the pre-specified number T_d, then it is labeled as the corresponding Gaussian component; and the mean and covariance of that component is updated correspondingly. If the entrance does not fit into the Gaussian Mixture, it will be marked as an anomaly and raise an alert.

5. The mahalanobis distance between existing alert records and current Gaussian component are calculated. If some of the existing alert records fall into the range of current Gaussian component, we re-label the alert record as part of the corresponding Gaussian component and recalculate the mean and covariance of that Gaussian component.

6. Each time a new alert is raised, all the anomaly records are gathered to examine if there is a new normal pattern emerging from the data. The anomaly records are fed to the PCM with the number of clusters set as one. The data points whose memberships are larger than $\mathbf{T_o}$ are treated as one cluster and the dispersion measure of which is calculated. If the cluster is legitimate and with the clinical staff's consent, a new normal pattern will be created for the Gaussian Mixture.

Pseudo Code

Choose $\mathbf{T_d}$, membership threshold $\mathbf{T_n}$, $\mathbf{T_o}$

Initialization:

Compute the possibilistic partition U_{CXN} of the data using the PCM

Find feature vectors whose memberships to all the clusters are lower than $\mathbf{T_n}$ and log them into anomaly history

Cluster the rest of the data using the AMPCM

For each cluster, calculate their mean and covariance for their corresponding Gaussian component

Update:

When there is a new data instance x_t:

Calculate x_t's mahalanobis distances to each of the Gaussian clouds and find the minimum

If the minimum distance < $\mathbf{T_d}$ Then

Update the mean and covariance of the winning Gaussian component

If any data in the anomaly records fall into the GMM:

relabel the data as normal and update the corresponding Gaussian

End If

Else Fire an alert and log x_t into anomaly history

Examine the anomaly history for an emergent new behavior pattern:

Compute the possibilistic partition U of the anomaly record using the PCM with C=1

Find the vectors with membership > $\mathbf{T_o}$ and computer the dispersion of them as a cluster If the cluster is legitimate & clinical records show no relevance to health change

Spawn a new normal Gaussian component

End If

End If

4 Experiments and Results

4.1 Determine the Number of Clusters based on the Combination of the PCM and AMPCM

To examine the ability of the combination of the PCM and AMPCM to determine the number of clusters under a noisy environment, it is run on four sets of data. The results are compared against the results of the PCM and the results of the AMPCM alone. For all the experiments in this section, the membership threshold for noise $\mathbf{T_n}$ = 0.06. For the PCM algorithm, the fuzzifier m is 1.5 and the number of clusters is chosen as the rounded square root of the number of data instances. For the AMPCM algorithm, we give p = 3 and ρ=0.9.

Dataset 1, as shown in Figure 4.1(a), consists of three Gaussians, where one cluster has 200 data points and the other two clusters have smaller volumes (i.e., size = 100 each). Random noise data drawn from uniform distribution with size of 25 are also mixed in the data. The PCM is able to find the noise, as marked in blue in Figure 4.1(b) and degenerate the clusters to be coincident clusters. Since there are 425 feature vectors in this dataset, the number of clusters for the PCM algorithm is chosen as 20. As we can see in the enlarged part of Figure 4.1(b), although all of the cluster centers are attracted to the three natural dense areas, problem still remains in merging those coincident cluster centers. Due to the fact that the AMPCM uses all data points as initial cluster centers and the clusters are merged based on correlation in membership, noisy data may become a cluster by itself. We can see that in Figure 4.1(c), the AMPCM manages to find all the

three Gaussians, yet 13 clusters are also created for the noise, many of which have only one element. The combination of PCM and AMPCM identifies most of the noise, as marked in blue in Figure 4.1(d), and successfully finds the natural number of clusters.

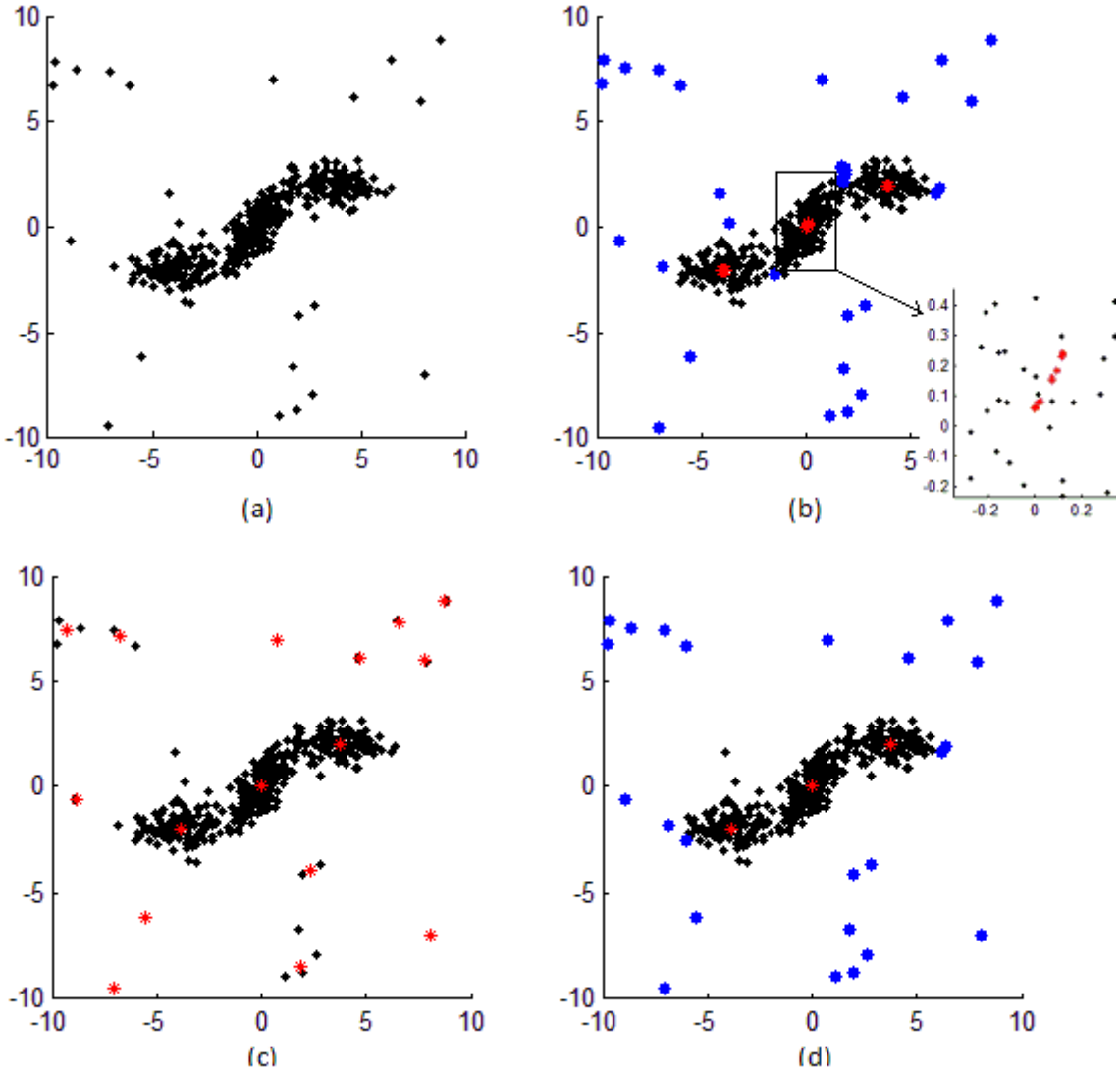


Figure 4.1 Hardened clustering in data set 1. Points in blue indicate recognized noise points, while vectors in red represent final cluster centers. (a) Original dataset. (b) Clustering result of the PCM with the number of clusters set as 20, fuzzifier m set as 1.5 and T_n is set as 0.06.(c) Clustering result of the AMPCM with p=3 and ρ=0.9. (d) Clustering result of the combination of the PCM and AMPCM, with the number of clusters set as 20, fuzzifier m set as 1.5 for the PCM, and p=3, ρ=0.9 for the AMPCM. T_n is set as 0.06.

Dataset 2, shown in Figure 4.2(a), is made of 6 Gaussians, each with 100 elements and same covariance, mixed with 30 noisy data points. As we can see in figure 4.2(b), the PCM clustering result is affected by the parameter selection, causing one of the cluster centers to be trapped in the local minima and not coinciding with other cluster centers. Also, the membership threshold $\mathbf{T_n}$ seems a little bit high for this case as some of the data points should be part of Gaussians are recognized as noise. The AMPCM, however, still finds all the natural clusters but separate clusters with noise as cluster centers are also created (Figure 4.2c). The combination of the PCM and AMPCM finds most of less-relevant data points and successfully clusters the dataset into 6 clusters, as shown in Figure 4.2d. But it is met with the same problem as the PCM due to the value of the membership threshold $\mathbf{T_n}$, i.e., some data points that should belong to the clusters are recognized as noise.

We next consider a dataset with 16 blocks in which each block has 50 data points, along with 100 noisy data points, as shown in Figure 4.3a. Using the rounded square root of the number of data instances as the number of clusters for the PCM gives a good result. As we can see in Figure 4.3(b), there are enough cluster centers to initialize the algorithm and most of the noise is recognized. The PCM is affected by the value of fuzzifer again as two cluster centers converge to the third block on the top row but not close enough to be coincident. The AMPCM is robust in parameter selection and initialization so that all the natural clusters are found correctly, as shown in Figure 4.3(c). The combination of the PCM and AMPCM gives a good result, as shown in Figure 4.3(d), most of the noise are recognized and the natural number of clusters is found.

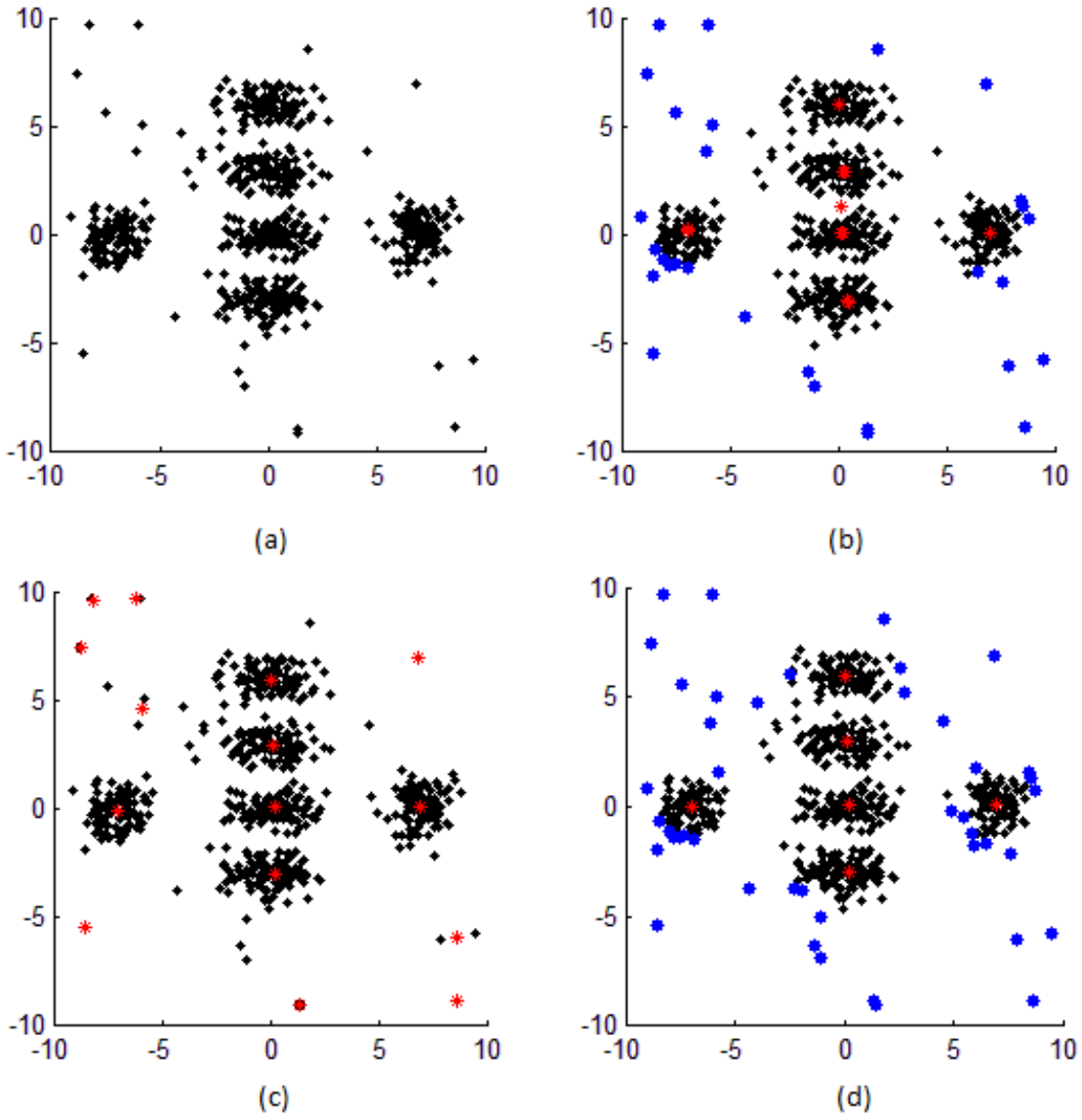


Figure 4.2 Hardened clustering in data set 2. Points in blue indicate recognized noise points, while vectors in red represent final cluster centers. (a) Original dataset. (b) Clustering result of the PCM with the number of clusters set as 20, fuzzifier m set as 1.5 and T_n is set as 0.06. (c) Clustering result of the AMPCM with p=3 and ρ=0.9. (d) Clustering result of the combination of the PCM and AMPCM, with the number of clusters set as 20, fuzzifier m set as 1.5 for the PCM, and p=3, ρ=0.9 for the AMPCM. T_n is set as 0.06.

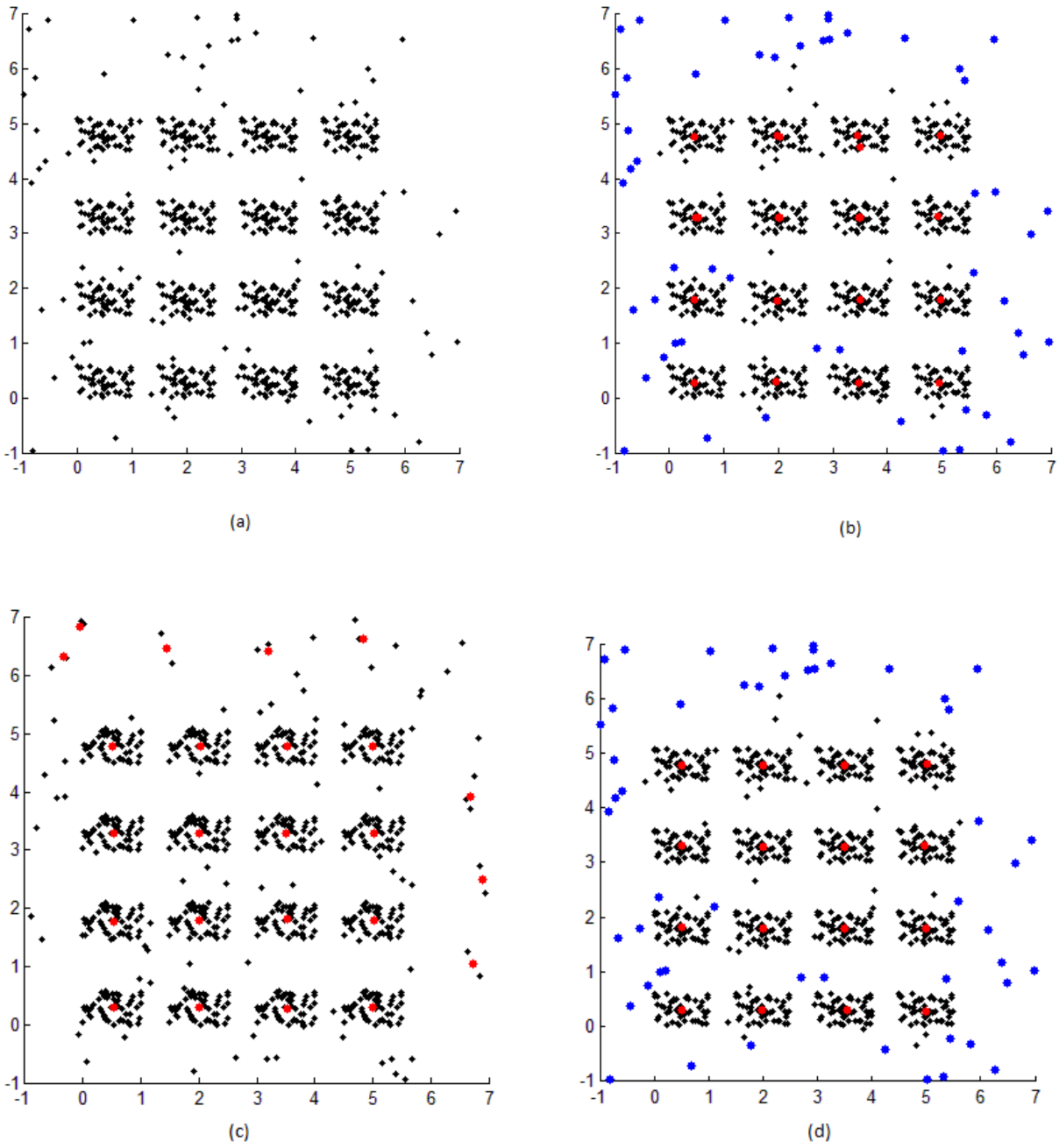


Figure 4.3 Hardened clustering in data set 3. Points in blue indicate recognized noise points, while vectors in red represent final cluster centers. (a) Original dataset. (b) Clustering result of the PCM with the number of clusters set as 30, fuzzifier m set as 1.5 and Tn is set as 0.06. (c) Clustering result of the AMPCM with p=3 and ρ=0.9. (d) Clustering result of the combination of the PCM and AMPCM, with the number of clusters set as 30, fuzzifier m set as 1.5 for the PCM, and p=3, ρ=0.9 for the AMPCM. Tn is set as 0.06.

Dataset 4 consists three Gaussians in three-dimensional space, where one cluster has 400 data points and the other two clusters have 150 data points each, and 30 noise data points. The PCM, shown in Figure 4.4(b) and the combination of the PCM and AMPCM, shown in Figure 4.4(d) give the same result in this case, while the AMPCM spawns extra clusters for the noise data, as shown in Figure 4.4(c).

As we can see, the combination of the PCM and AMPCM not only has the AMPCM's robustness to parameter selection, but it is also able to isolate the noise from the rest of the data. What's more, by using the combination of the PCM and AMPCM, we no longer need to worry about merging the coincident clusters as we need to do when we use the PCM to cluster the data. The membership threshold for noise $\mathbf{T_n}$ is the only parameter that might affect the performance of the combination of the PCM and AMPCM. The same membership threshold will have different effects on different datasets. If the value of $\mathbf{T_n}$ is big, then some of the valid data points will be recognized as noise (Figure 4.2d); if the value of $\mathbf{T_n}$ is small, then extra clusters might be created for the unrecognized noise data. Yet this problem can be compensated by choosing the right mahalanobis distance threshold **N**, which will be examined in section 4.3.

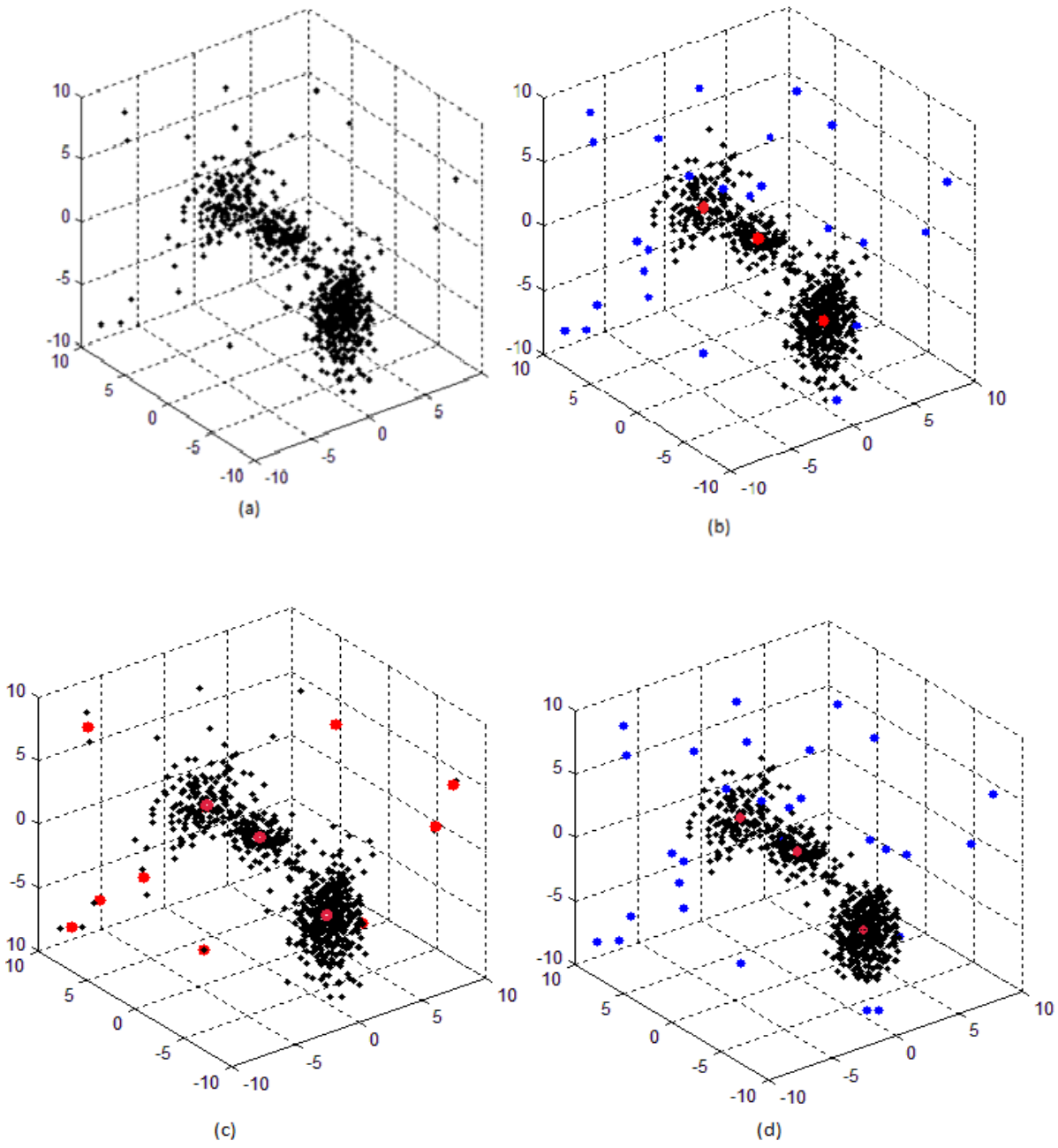


Figure 4.4 Hardened clustering in data set 4. Points in blue indicate recognized noise points, while vectors in red represent final cluster centers. (a) Original dataset. (b) Clustering result of the PCM with the number of clusters set as 20, fuzzifier m set as 1.5 and T_n is set as 0.06. (c) Clustering result of the AMPCM with p=3 and ρ=0.9. (d) Clustering result of the combination of the PCM and AMPCM, with the number of clusters set as 20, fuzzifier m set as 1.5 for the PCM, and p=3, ρ=0.9 for the AMPCM. T_n is set as 0.06.

4.2 Building the GMM using the possibilistic partition

Now that we can have a clustering result of the data from the initial time window, it's time to compute the mean and covariance matrix of each Gaussian to build the GMM. Using the possibilistic clustering algorithm enables us to get the memberships indicating typicalities. Crisp labels can be assigned to the data based on the membership values, as shown in the color of the marker in Figures 4.5, 4.6, 4.7. Computing the mean and covariance matrix based on the crisp labels is an easy choice, but the valuable typicality information in the possibilistic partition will be wasted. In this section, three different methods to compute the mean and covariance matrix for the Gaussian are experimented and compared.

The first approach is the most straightforward, where only the crisp labels of the data is used and the data with the same labels are gathered and the mean and covariance matrix are calculated. We will call it the crisp method.

The second approach gathers the data with the same labels first, then using the possibilistic partition information to compute the weighted mean and covariance matrix (equation 4.1) of each cluster. Let all the data points in cluster n: $C_n = \{x_{n1}, x_{n2, \dots,} x_{nd}\}$, assuming there are d data points belonging to cluster n according to the crisp partition of the membership, while n1, n2, …, nd are the indices of the feature vectors belong to cluster n. The equations to calculate the weighted mean $\boldsymbol{\mu}^*$ and covariance $\boldsymbol{\Sigma}^*$ are as follows:

$$\boldsymbol{\mu}^* = \frac{\sum_{i=1}^{N} \omega_i \boldsymbol{x}_i}{\sum_{i=1}^{N} \omega_i}$$

$$\boldsymbol{\Sigma}^{*} = \frac{\sum_{i=1}^{N} \omega_i \cdot \sum_{i=1}^{N} \omega_i (\boldsymbol{x}_i - \boldsymbol{\mu}^{*})^{T} (\boldsymbol{x}_i - \boldsymbol{\mu}^{*})}{\left(\sum_{i=1}^{N} \omega_i\right)^{2} - \sum_{i=1}^{N} \omega_i^{2}} \tag{4.1}$$

In the equations, ω_i is the weight assigned to each vector $\boldsymbol{x}_i$. When computing the weighted mean and covariance matrix in this case, the weight will be the typicality value of the vector towards the cluster its crisp label is assigned to and instead of using all the feature vectors, only the data points belong to the cluster in crisp partition will be used in computing the cluster's weighted mean and covariance matrix. We will call this the fuzzy method.

The crisp labels are no longer used in the third approach as all the membership values to cluster n are used to compute its weighted mean and covariance, using equation 4.1. We will call it the fuzzy-2 method.

Those three approaches are used on three datasets. Dataset 5 consists of two Gaussians with diagonal covariance matrices with random noise. The results of using the crisp, fuzzy, fuzzy-2 method are shown in Figure 4.5. The means used to generate the data are shown as blue circles, the cluster centers computed by the three methods are marked in pink, and the eigenvectors are shown to depict the covariance matrices. To have a better insight into those three methods, the difference between the mean value from the original data and the mean computed by those methods is calculated and the difference angle of the principle eigenvector is also calculated to indicate difference in the covariance matrix. The difference in span, which is the product of the eigenvalues, is also calculated for investigation. Table 4.1 shows the results averaging over 20 trials with the data randomly generated with the same parameter setting (same means and covariance matrices for the two dense regions of the data and noise data generated from uniform distribution). In the

single trial shown in Figure 4.5, the fuzzy-2 method computes the best results for the covariance matrices as the two eigenvectors are most horizontal and vertical. Yet according the results averaging over 20 trials, the crisp method and the fuzzy-2 method give the almost as good results while the fuzzy method is less than satisfactory in comparison. As shown in the table, the eigenvalues computed from the fuzzy-2 method are significantly smaller than the other due to the smaller determinant of the covariance matrix. We can also see this in the Figure 4.5, the ellipsoids drawn by the means and covariance matrices according to the fuzzy-2 method is much smaller in size with respect to the others. Yet this shouldn't be a big issue. As with some manipulation with the distance threshold **N**, the difference can be easily compensated.

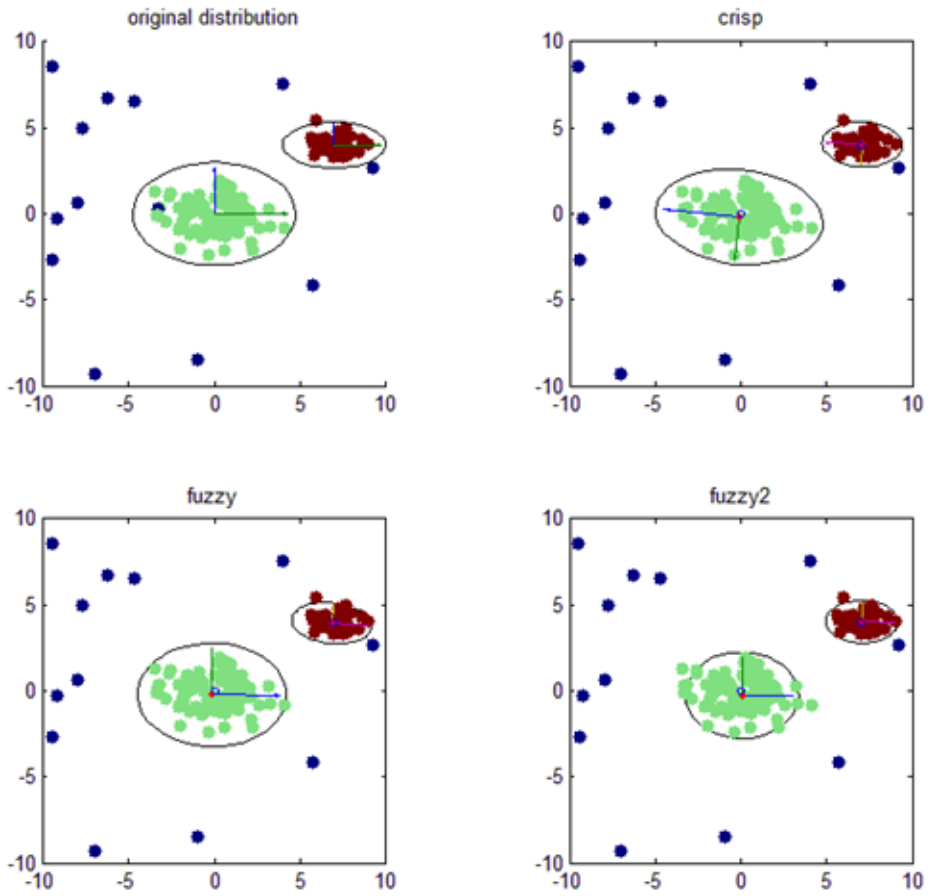


Figure 4.5 GMM built using different methods on dataset 5

Table 4.1 Numeric results of computing the Gaussian parameters on dataset 5

	crisp	fuzzy	fuzzy-2
Avg. diff. between the means w.r.t. the range of the data (%)	0.90%	1.42%	0.92%
Avg. difference angle in the Gaussian orientation (in °)	5.87	9.78	6.62
Avg. difference in span (prod. of eigenvalues)	-4.71%	-6.99 %	-32.76%

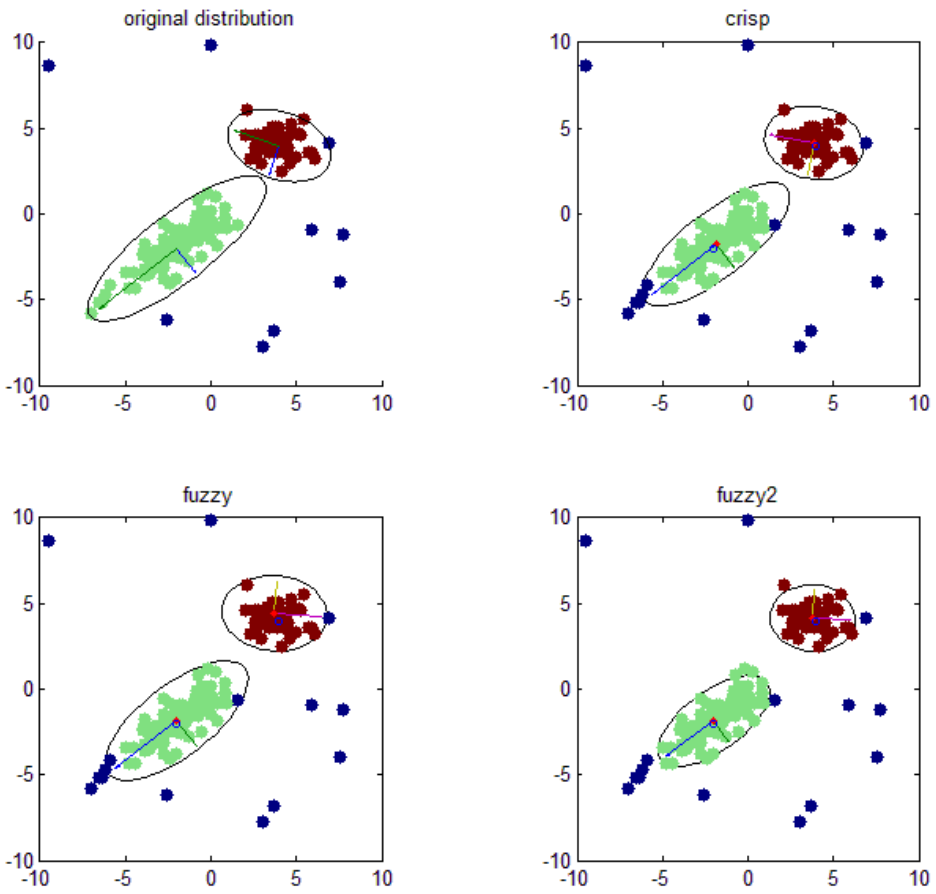


Figure 4.6 GMM built using different methods on dataset 6

Table 4.2 Numeric results of computing the Gaussian parameters on dataset 6

	crisp	fuzzy	fuzzy-2
Avg. diff. between the means w.r.t. the range of the data (%)	0.96%	1.45%	0.94%
Avg. difference angle in the Gaussian orientation (in °)	6.67	14.65	6.91
Avg. difference in span (prod. of eigenvalues)	-5.07%	-10.58 %	-27.43%

A dataset with full covariance matrices is also examined, as shown in Figure 4.6. In this case, both the crisp method and the fuzzy-2 method give the same performance except for the difference in eigenvalues, while the fuzzy method yields evident error in estimating the mean for one of the Gaussians. Table 4.2 is the results averaging over 20 trials on this set and it shows similar conclusion as what we observed from the Figure 4.6. Again, the crisp method and the fuzzy-2 method give good results while the fuzzy method creates more error.

Figure 4.7 shows another dataset with different parameter settings. We can see that the crisp labels are assigned to some of the noise, which leads to error when computing the covariance using the crisp method. By utilizing the possibilistic partition, the fuzzy method and fuzzy-2 method are more robust to the influence of the noise in this case, thus give a better estimate of the covariance matrices.

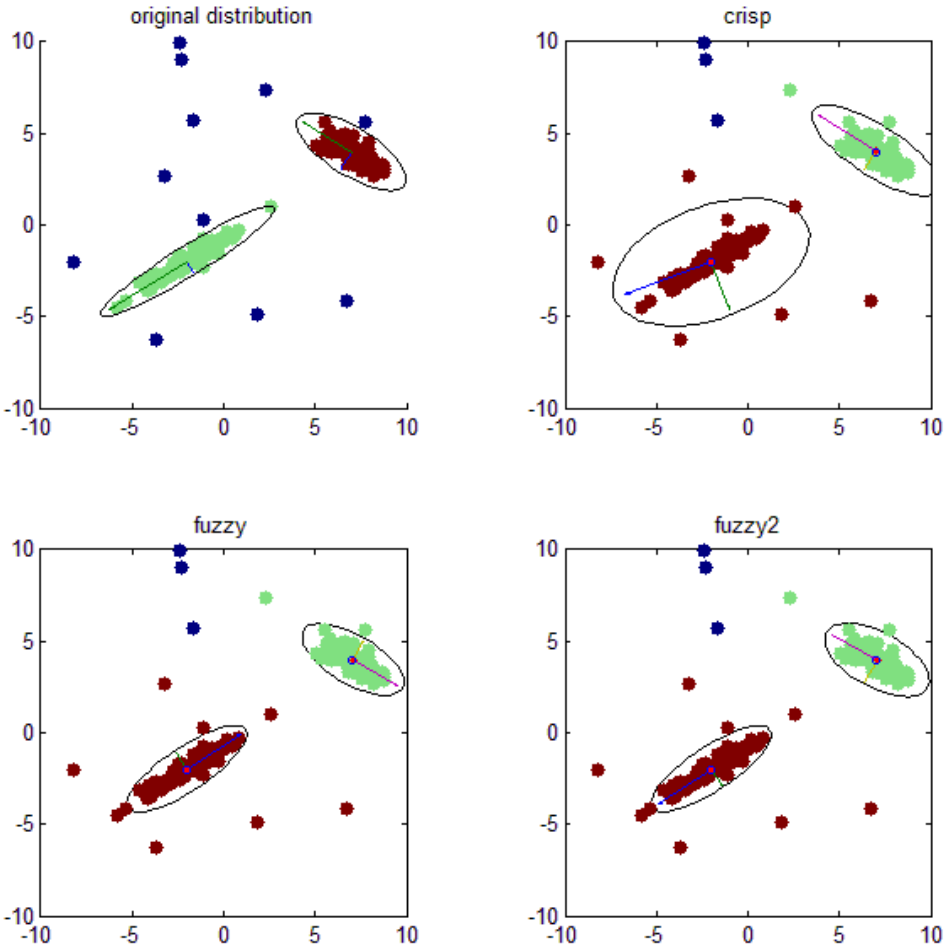


Figure 4.7 GMM built using different methods on dataset 7

Table 4.3 Numeric results of computing the Gaussian parameters on dataset 7

	crisp	fuzzy	fuzzy-2
Avg. diff. between the means w.r.t. the range of the data (%)	1.11%	1.49%	1.04%
Avg. difference angle in the Gaussian orientation (in °)	3.01	4.54	2.79
Avg. difference in span (prod. of eigenvalues)	-3.47%	-11.18 %	-25.23%

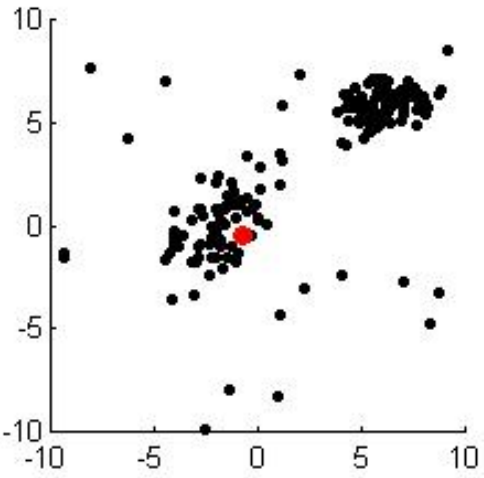
We can see that the fuzzy-2 method and the crisp method give the better results compared with the fuzzy method. Although it is hard to tell which method is better according to the statistics from the tables, the crisp method tend to underestimate the eigenvalues of the covariance matrices (the determinants of the covariance matrices), that is, given the same point, the mahalanobis distance computed using the results from the fuzzy-2 method will be much smaller than the distance estimated by using the crisp method and the fuzzy method. But this can be easily compensated by choosing the appropriate distance threshold $\mathbf{T_d}$. It is recommended to use the fuzzy-2 method instead of the crisp method to compute the means and covariance matrices due to its robustness.

4.3 Parameter for Updating the Gaussian Mixture Model

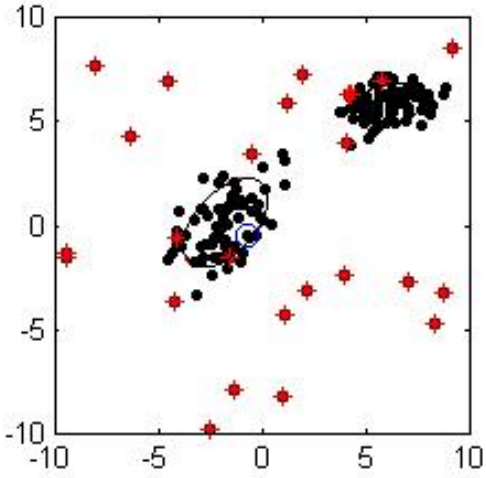
The only parameter selection problem that needs investigation when updating the Gaussian Mixture Model is the distance threshold T_d for each Gaussian component: within what range of the Gaussian qualifies the new data to be part of the normal Gaussian component?

We experiment with different values of T_d on a sequence of data to see its effect on updating the Gaussian Mixture Model. The original dataset is consisted of two clusters and a new data point is added to simulate the recording of the embedded sensor data. Figure 4.8(a1) - (a18) show a little clip of the update of the data sequence with each picture showing a new data entrance. It is worth mention that in this experiment we use the crisp method to calculate the means and covariance matrices for computational simplicity. The mahalanobis distance threshold T_d is chosen as 2 (Figure 4.8(b1) - (b18)),

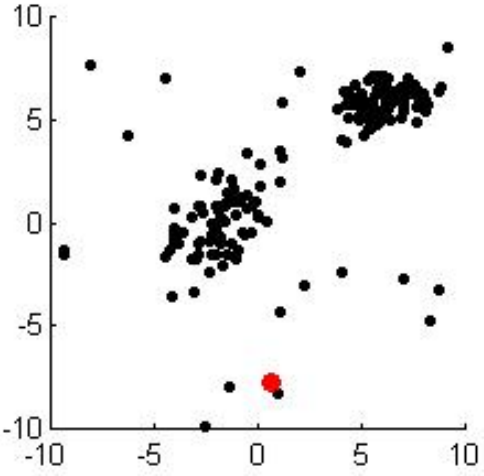
3 (Figure 4.8(c1) - (c18)) and 5(Figure 4.8(d1) - (d12)), with the black ellipsoids illustrating the boundary of the Gaussian ellipses, the red solid-lined ellipsoids illustrating the boundary of the previous ellipses and the red dashed ellipsoids illustrating the boundary of even earlier ellipses. We can see that when $T_d = 2$, the update of the GMM is not able to accommodate the gradual changes as new data points start heading away from the center of the cloud very slowly, especially in Figure 4.8(b9) - (b12). Due to the slow change in the data, unnecessary alerts are fired. When $T_d = 3$, as the data heading away from the center of the Gaussian, it can be observed that the boundary of the Gaussian is pushed in the same direction to accommodate the change. The update process is able to handle the gradual changes and achieve a good generalization. When $T_d = 5$, the GMM loses its ability in describing the data distribution and the boundary grows faster and faster while more and more noise start to fall into the range of the Gaussian. As we can see, the GMM is able to estimate the data distribution and adapt to slow changes given the mahalanobis distance threshold $T_d = 3$ in this two-dimensional case. Video version of those update processes can be viewed on http://youtu.be/-7C4YjmJxqQ.



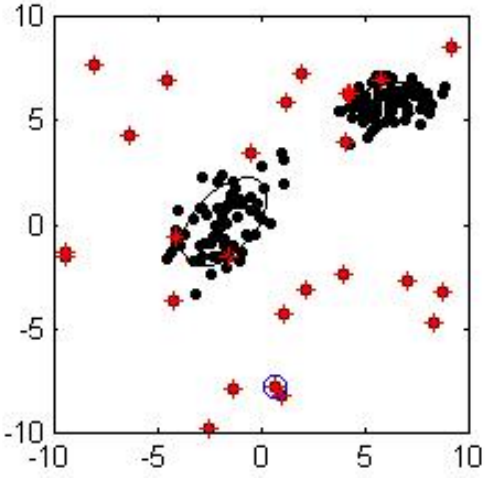
(a1)



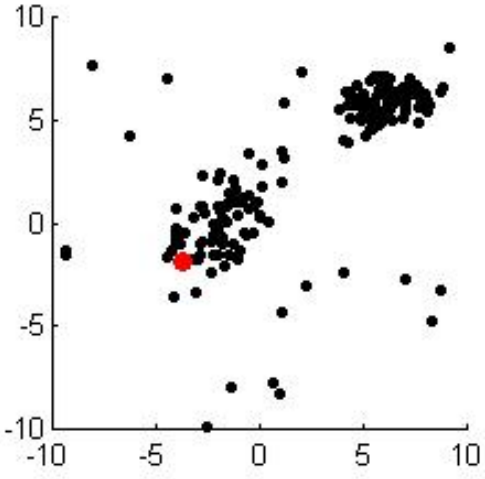
(b1)



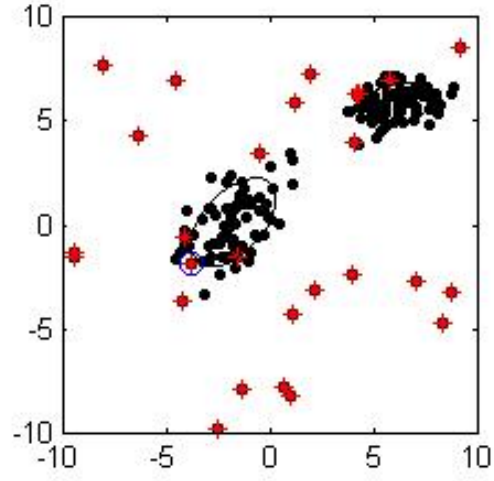
(a2)



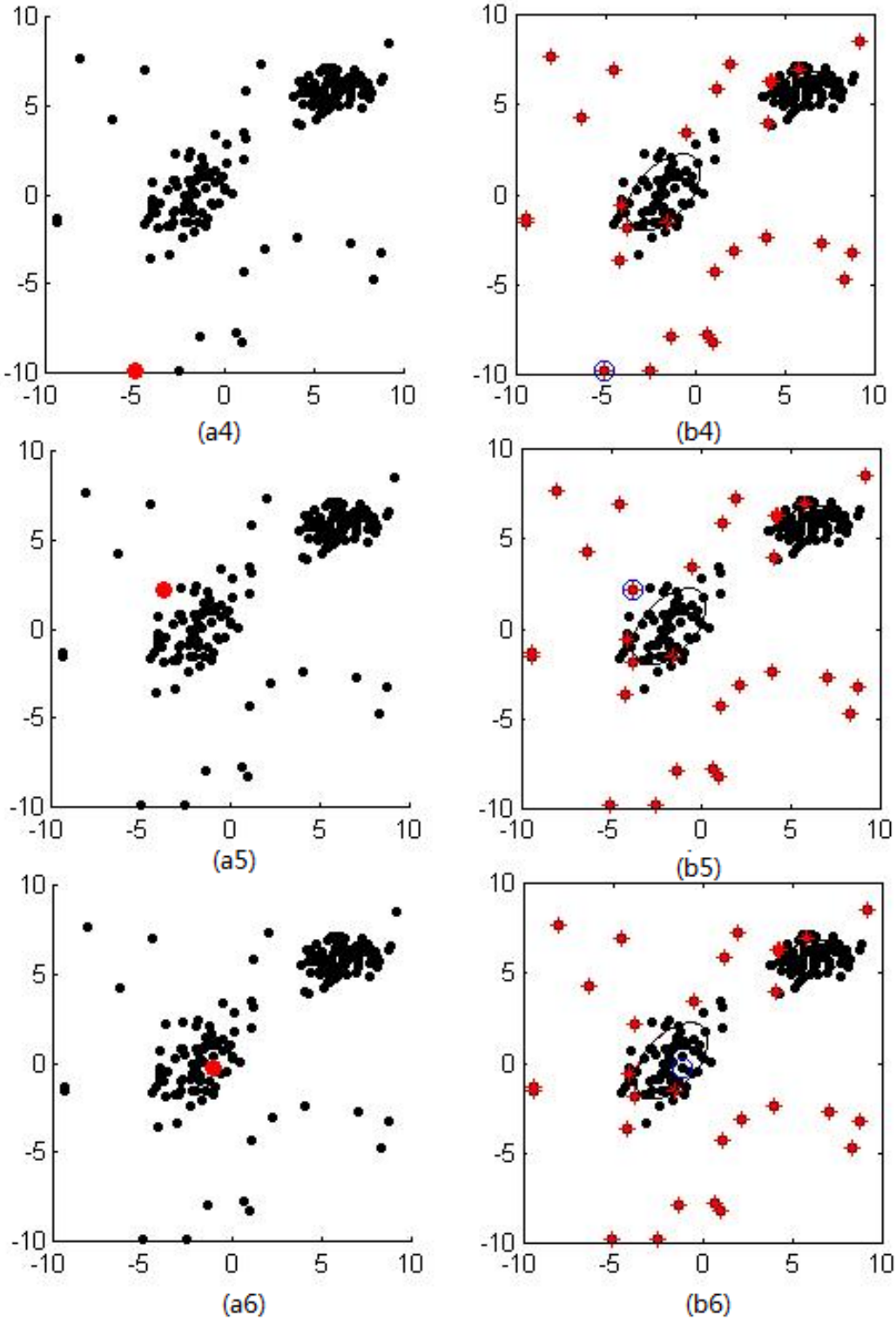
(b2)

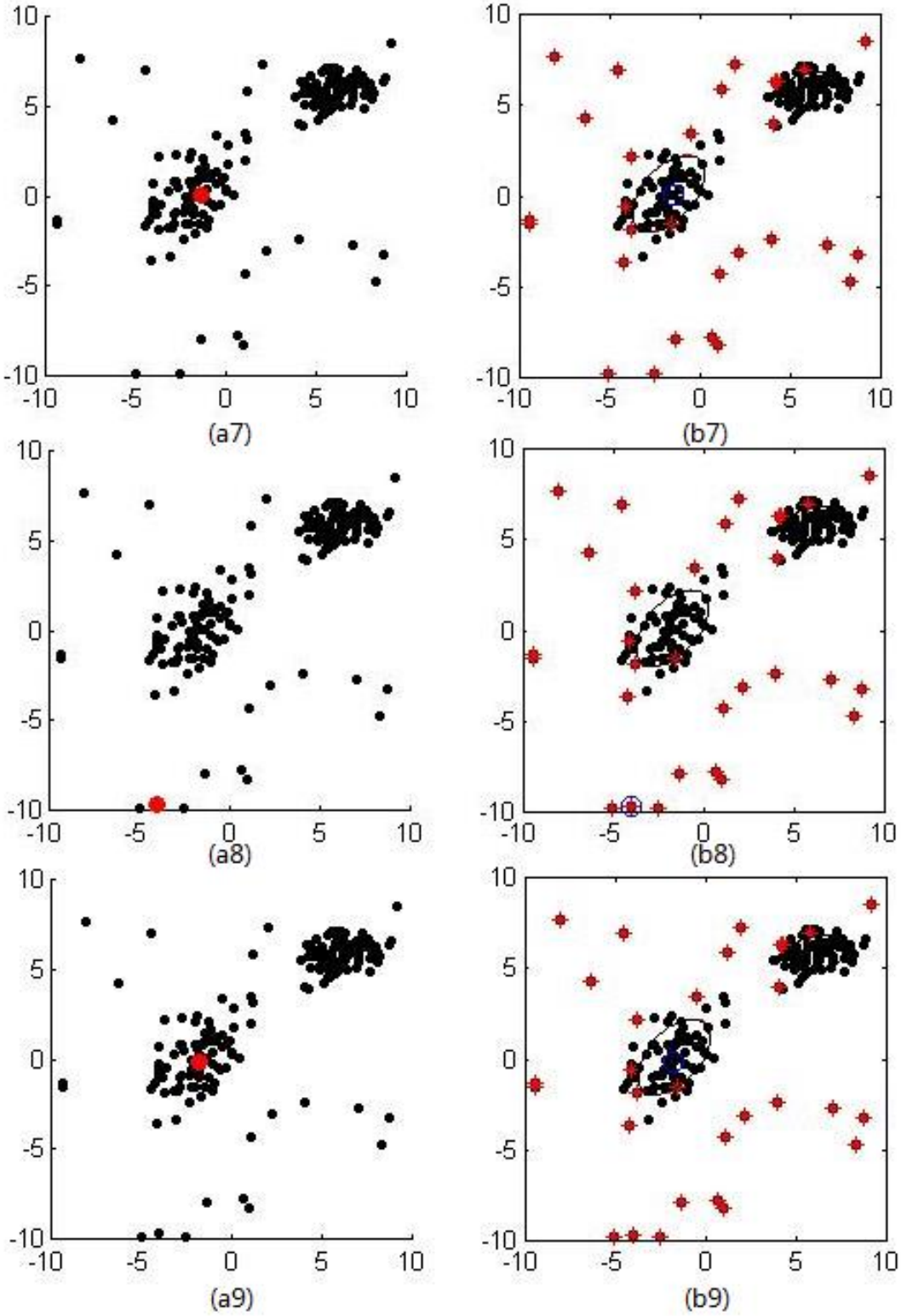


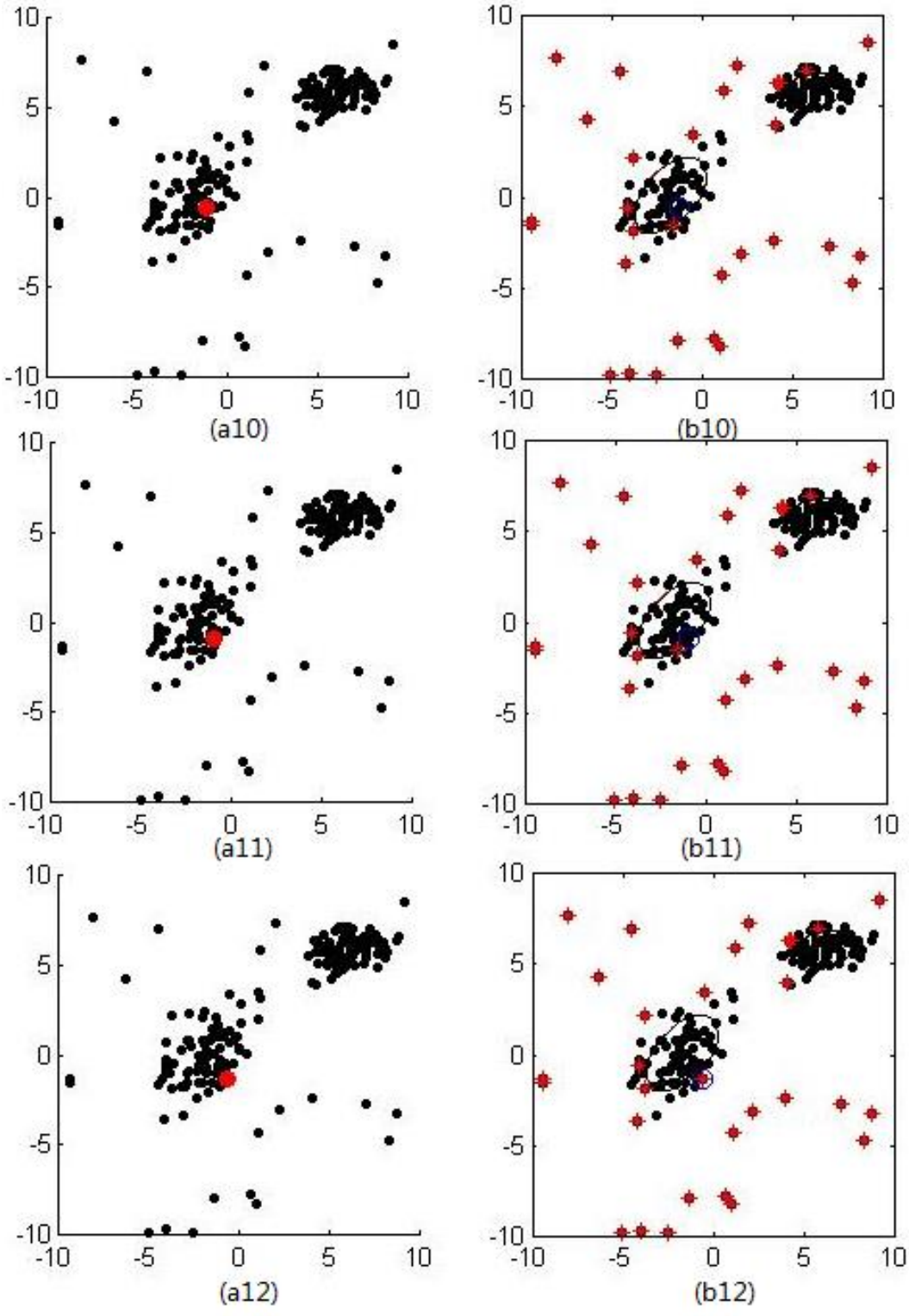
(a3)

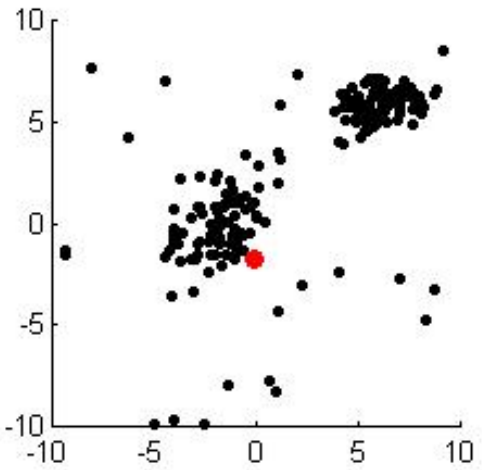


(b3)

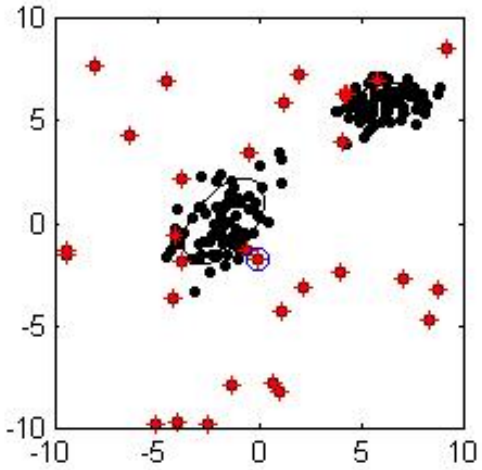

(a4)
(b4)
(a5)
(b5)
(a6)
(b6)


(a7)
(b7)
(a8)
(b8)
(a9)
(b9)

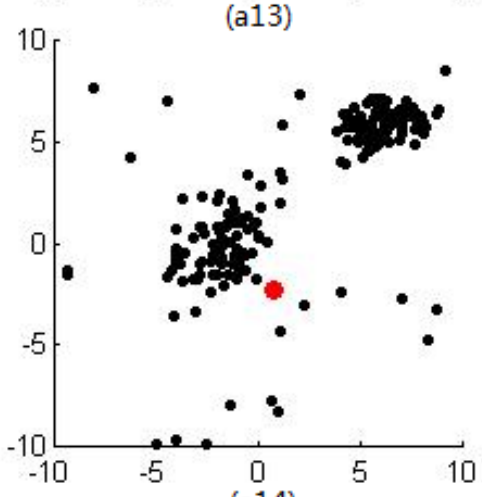

(a10)
(b10)
(a11)
(b11)
(a12)
(b12)



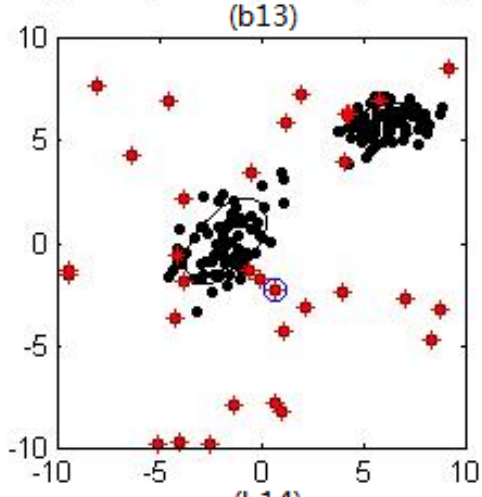
(a13)



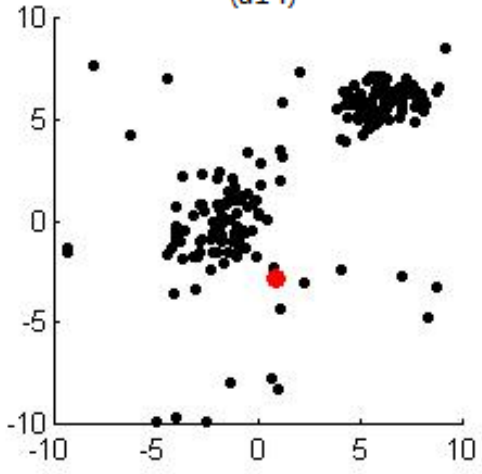
(b13)



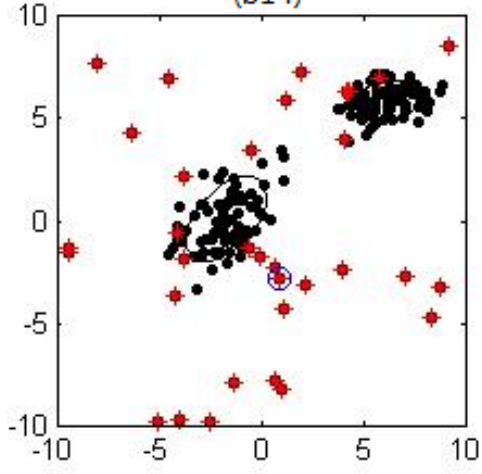
(a14)



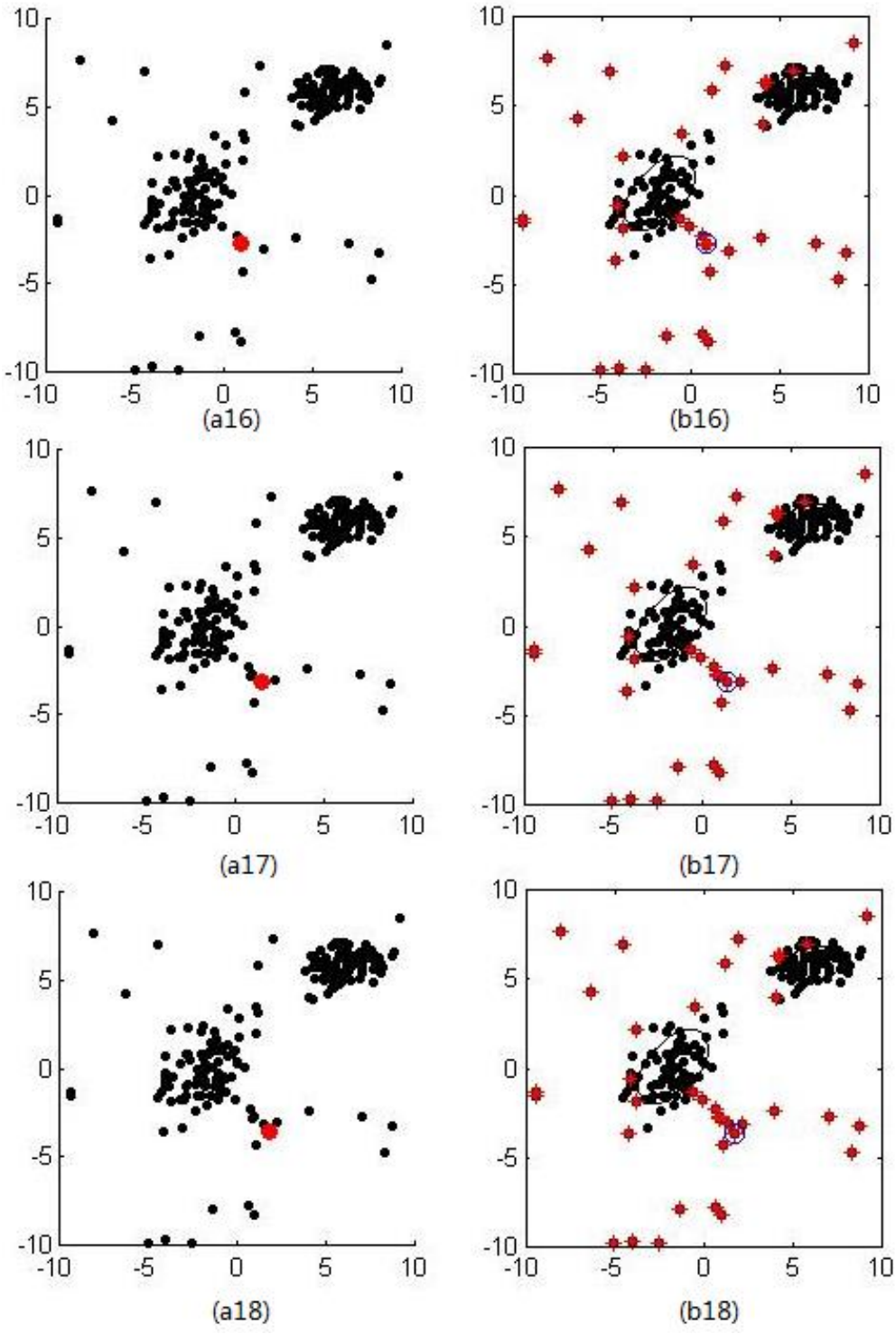
(b14)

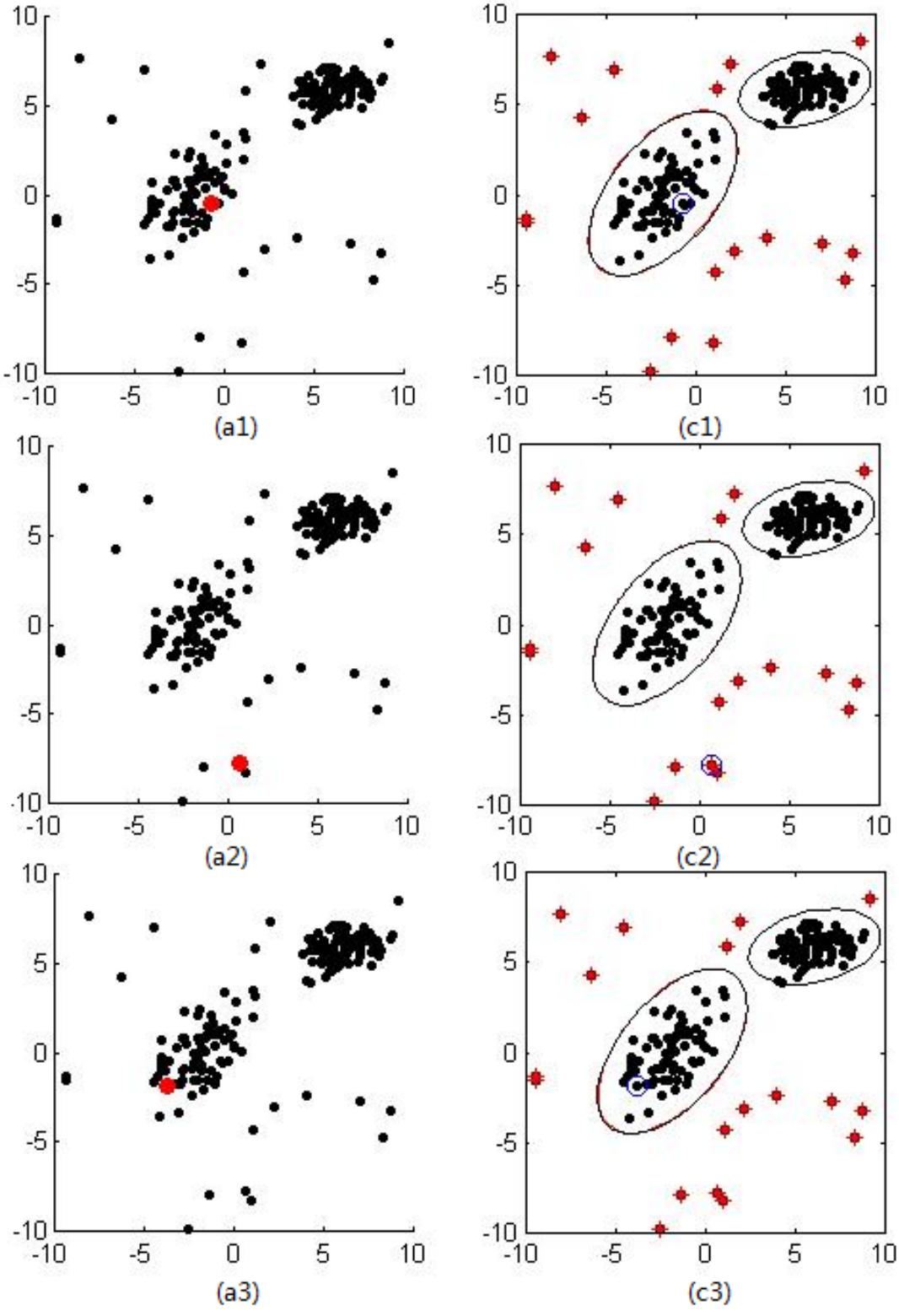


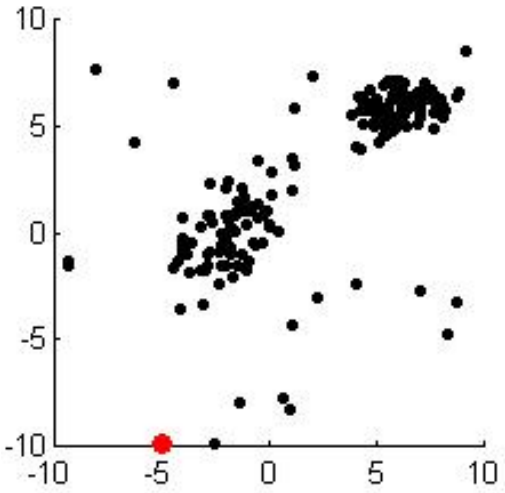
(a15)



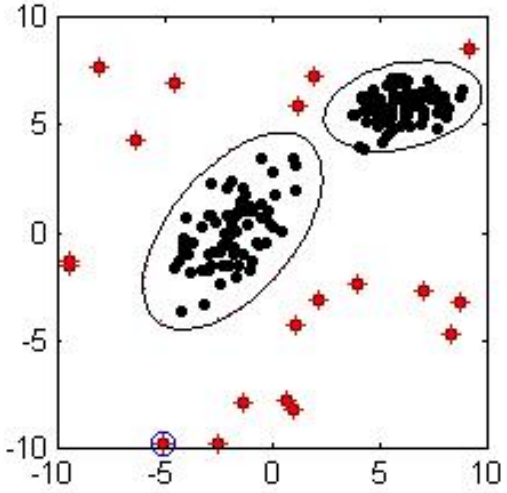
(b15)


(a16)
(b16)
(a17)
(b17)
(a18)
(b18)

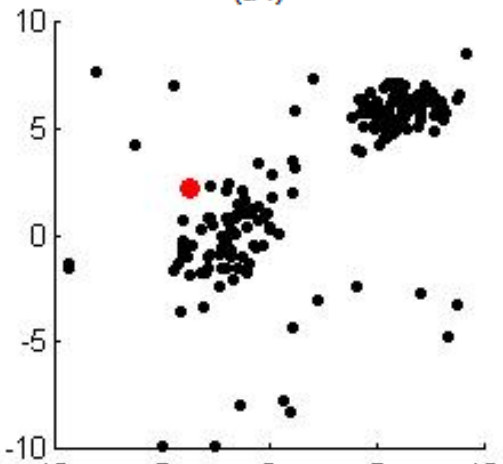

(a1)
(c1)
(a2)
(c2)
(a3)
(c3)



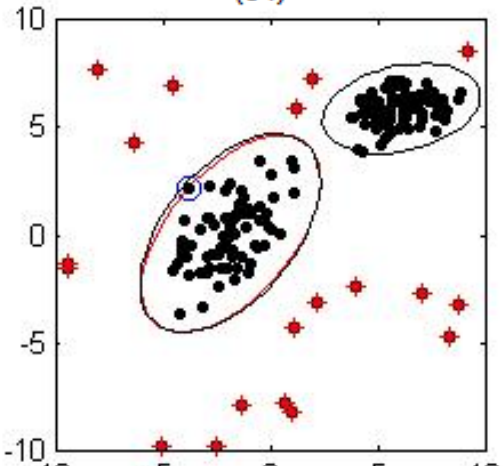
(a4)



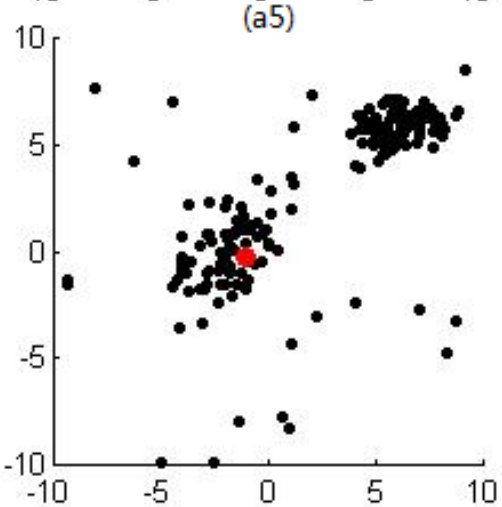
(c4)



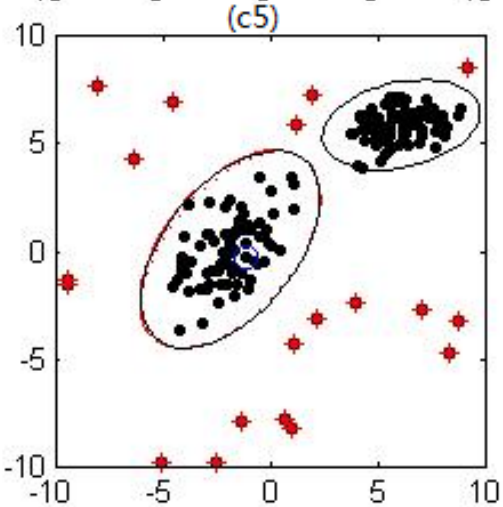
(a5)



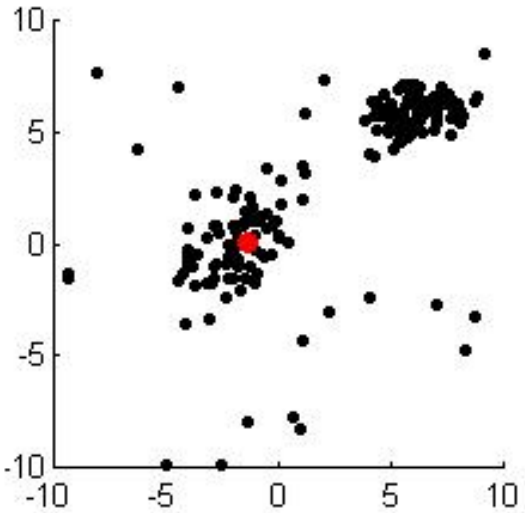
(c5)



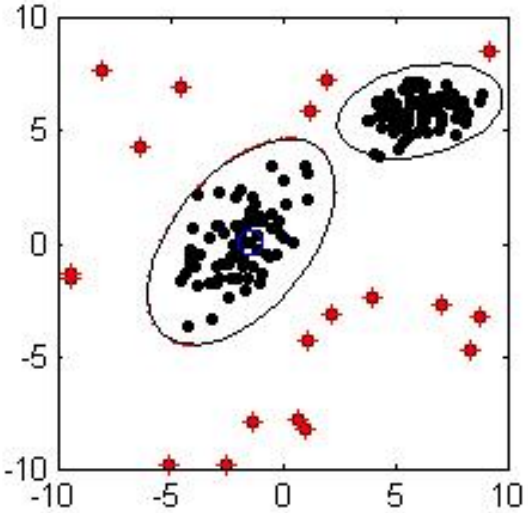
(a6)



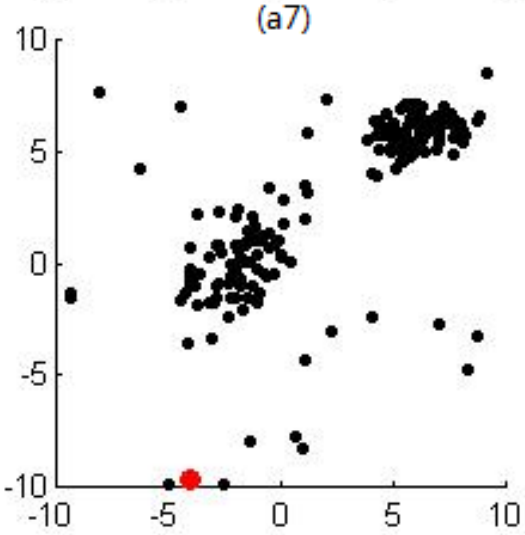
(c6)



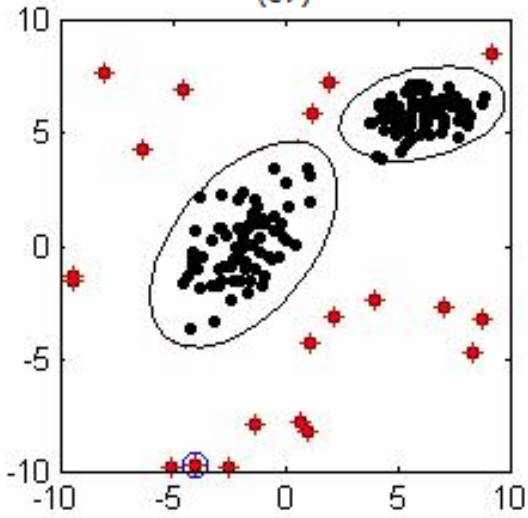
(a7)



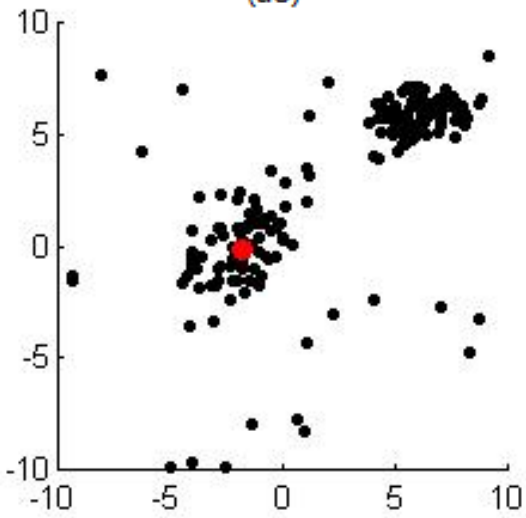
(c7)



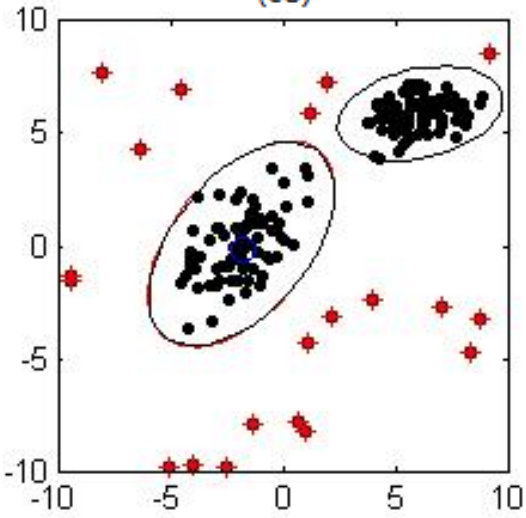
(a8)



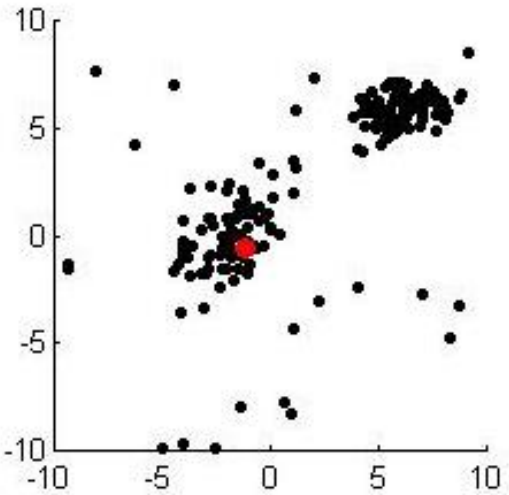
(c8)



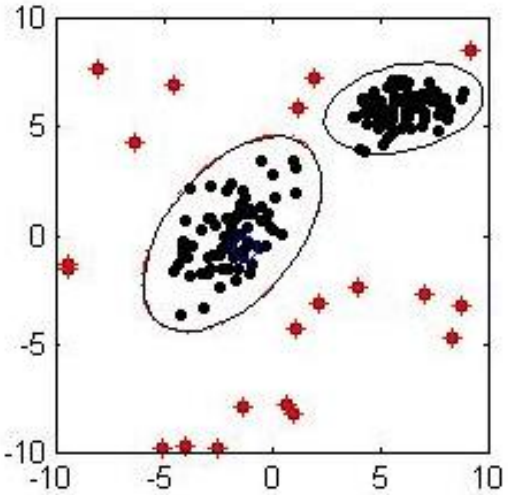
(a9)



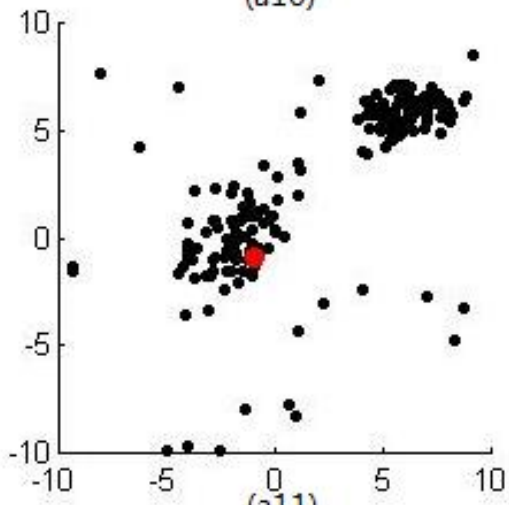
(c9)



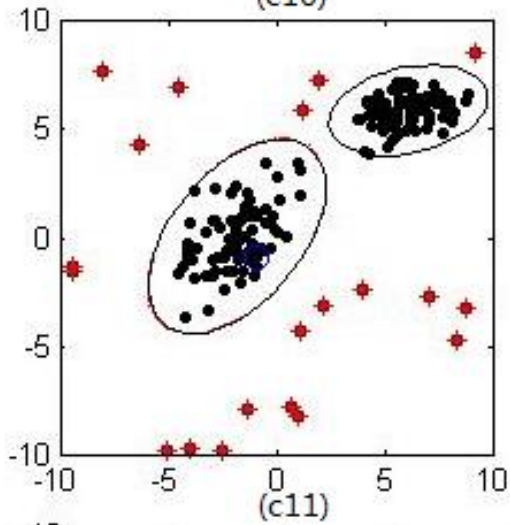
(a10)



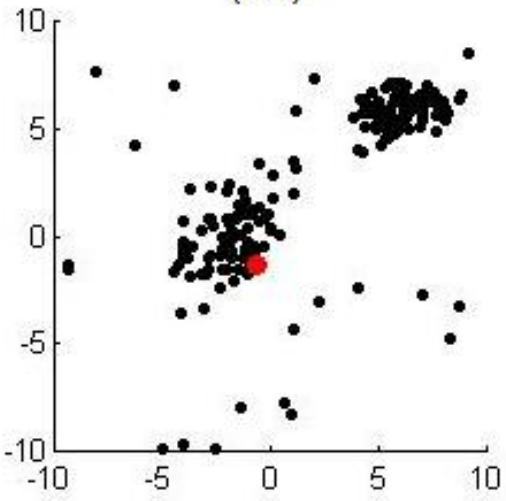
(c10)



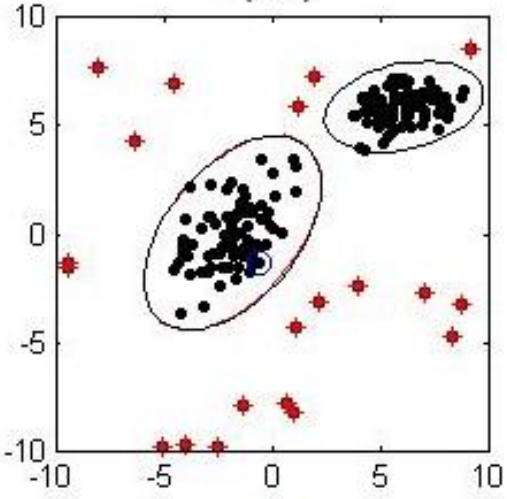
(a11)



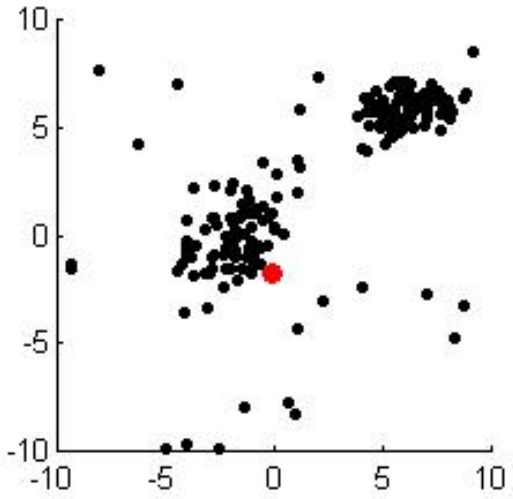
(c11)



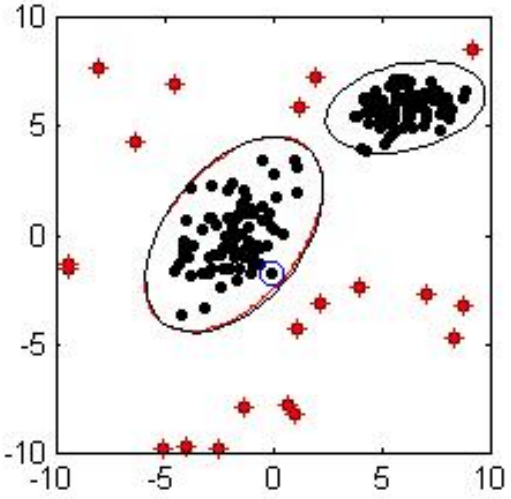
(a12)



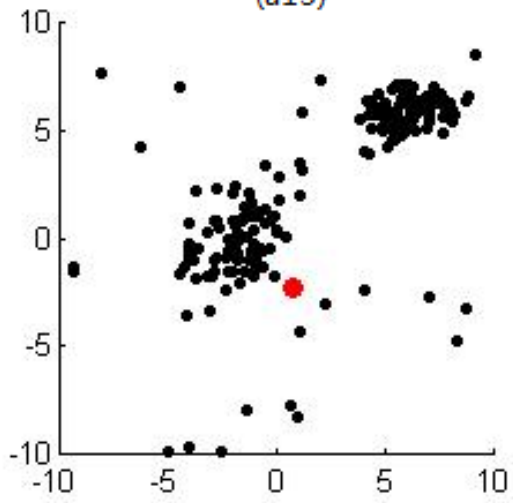
(c12)



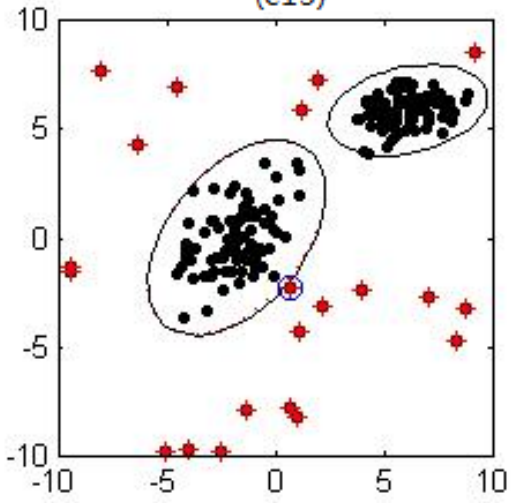
(a13)



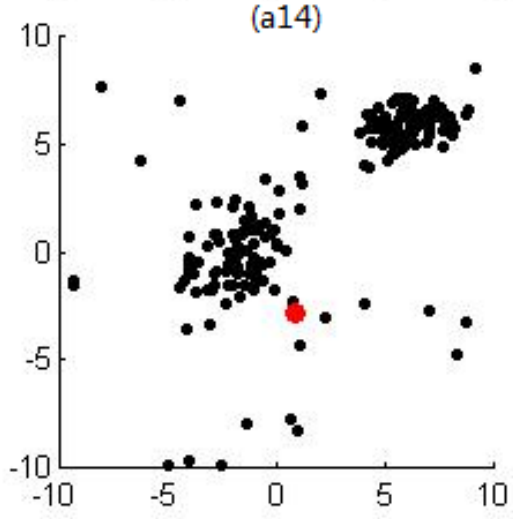
(c13)



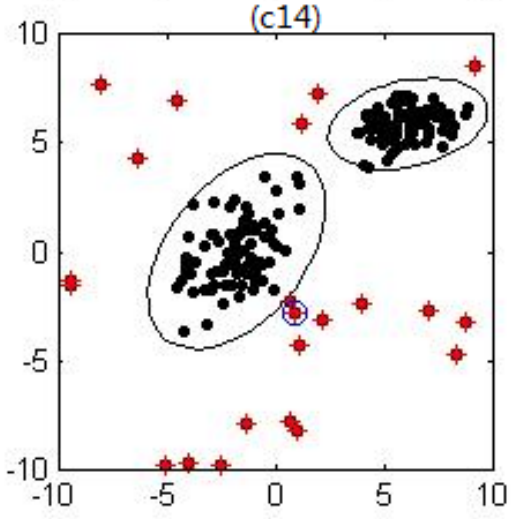
(a14)



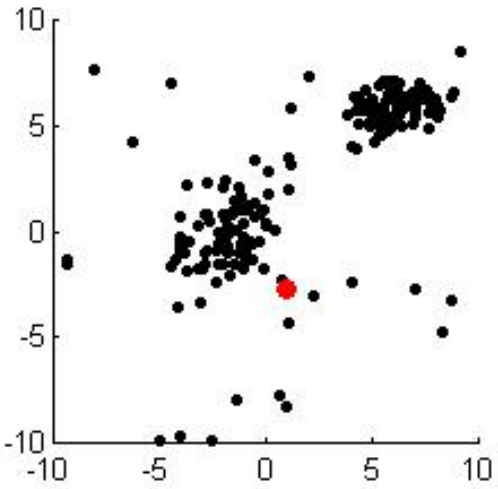
(c14)



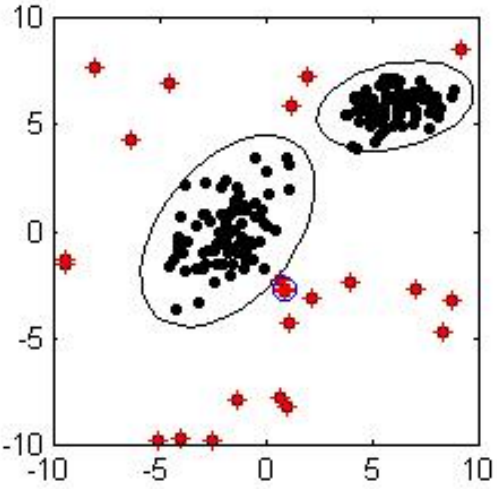
(a15)



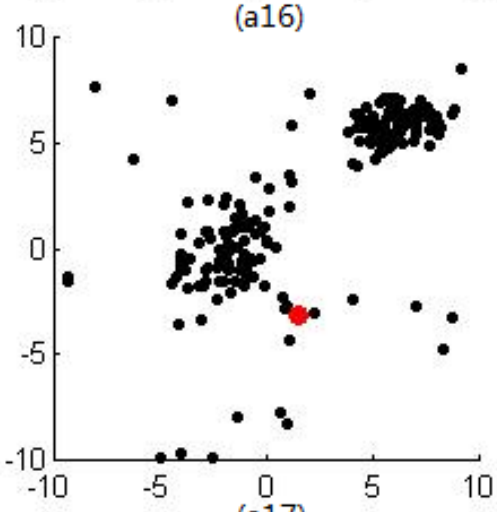
(c15)



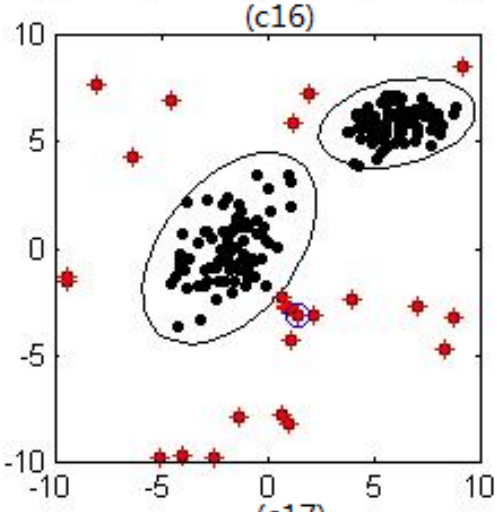
(a16)



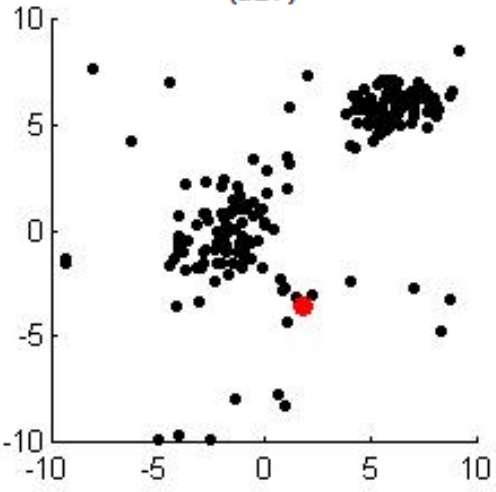
(c16)



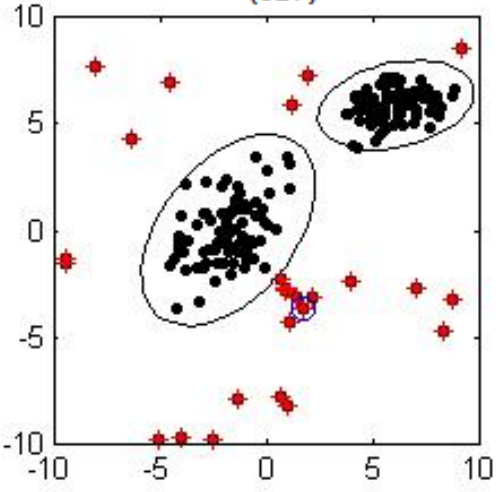
(a17)



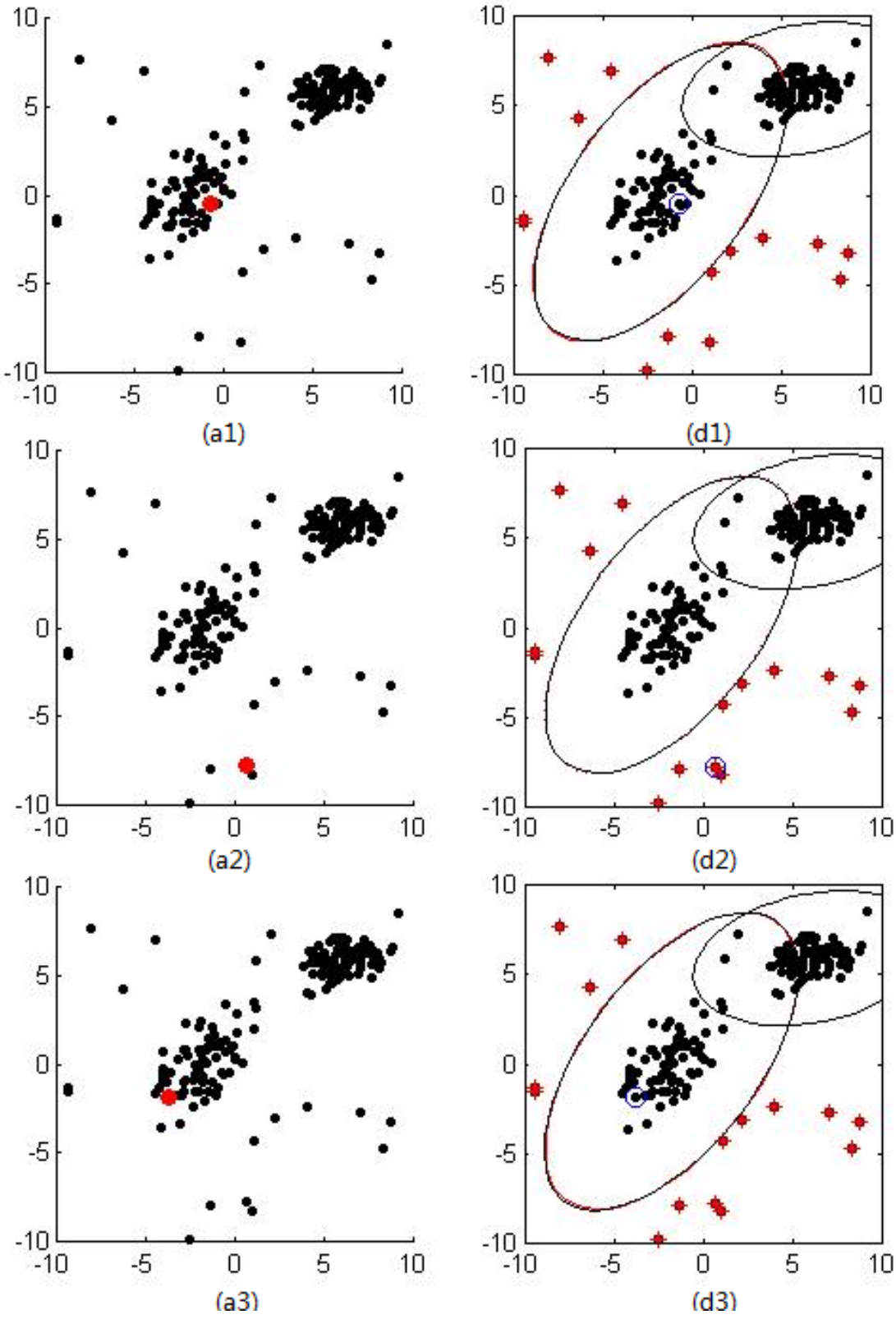
(c17)

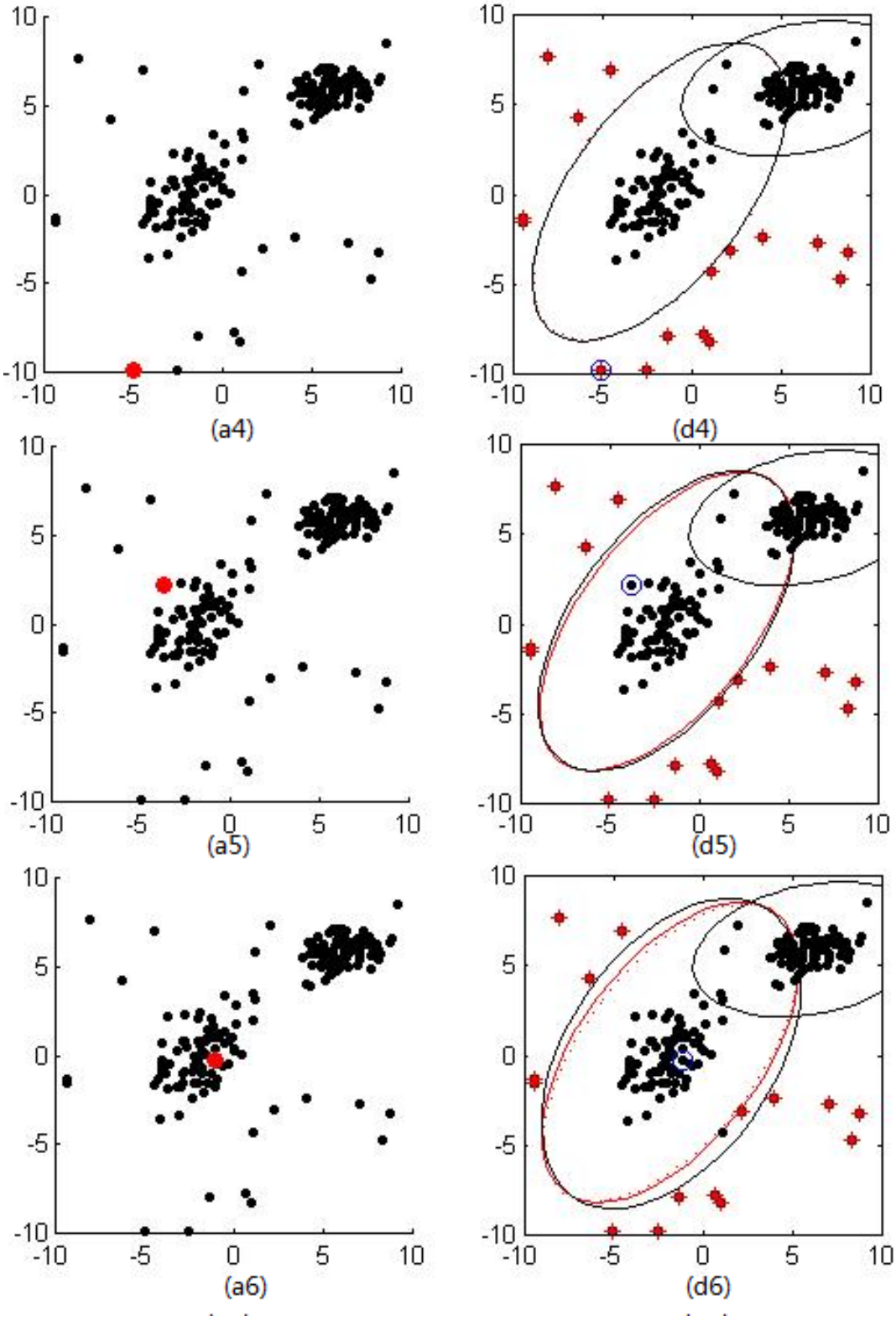


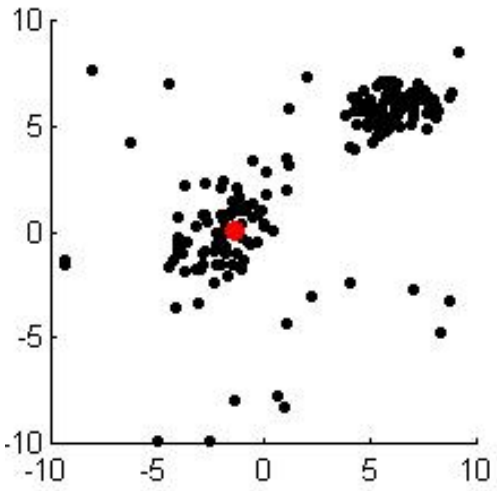
(a18)



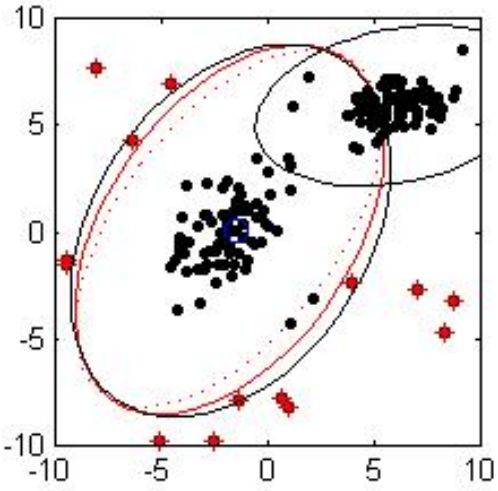
(c18)


(a1)
(d1)
(a2)
(d2)
(a3)
(d3)

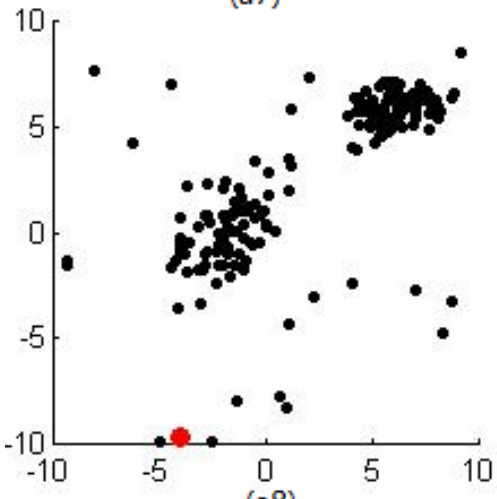

(a4)
(d4)
(a5)
(d5)
(a6)
(d6)



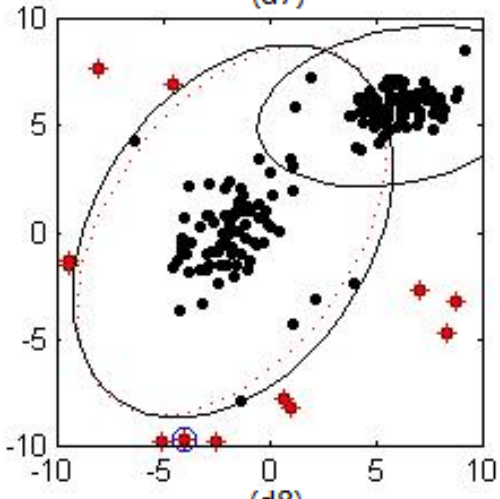
(a7)



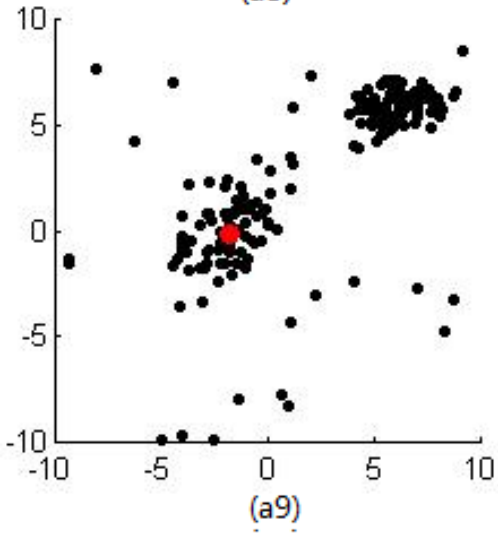
(d7)



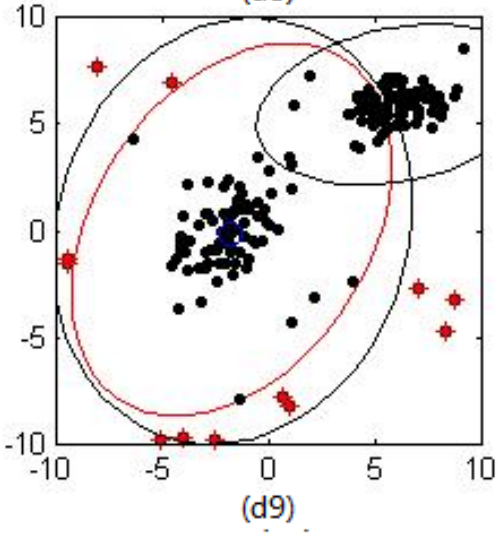
(a8)



(d8)



(a9)



(d9)

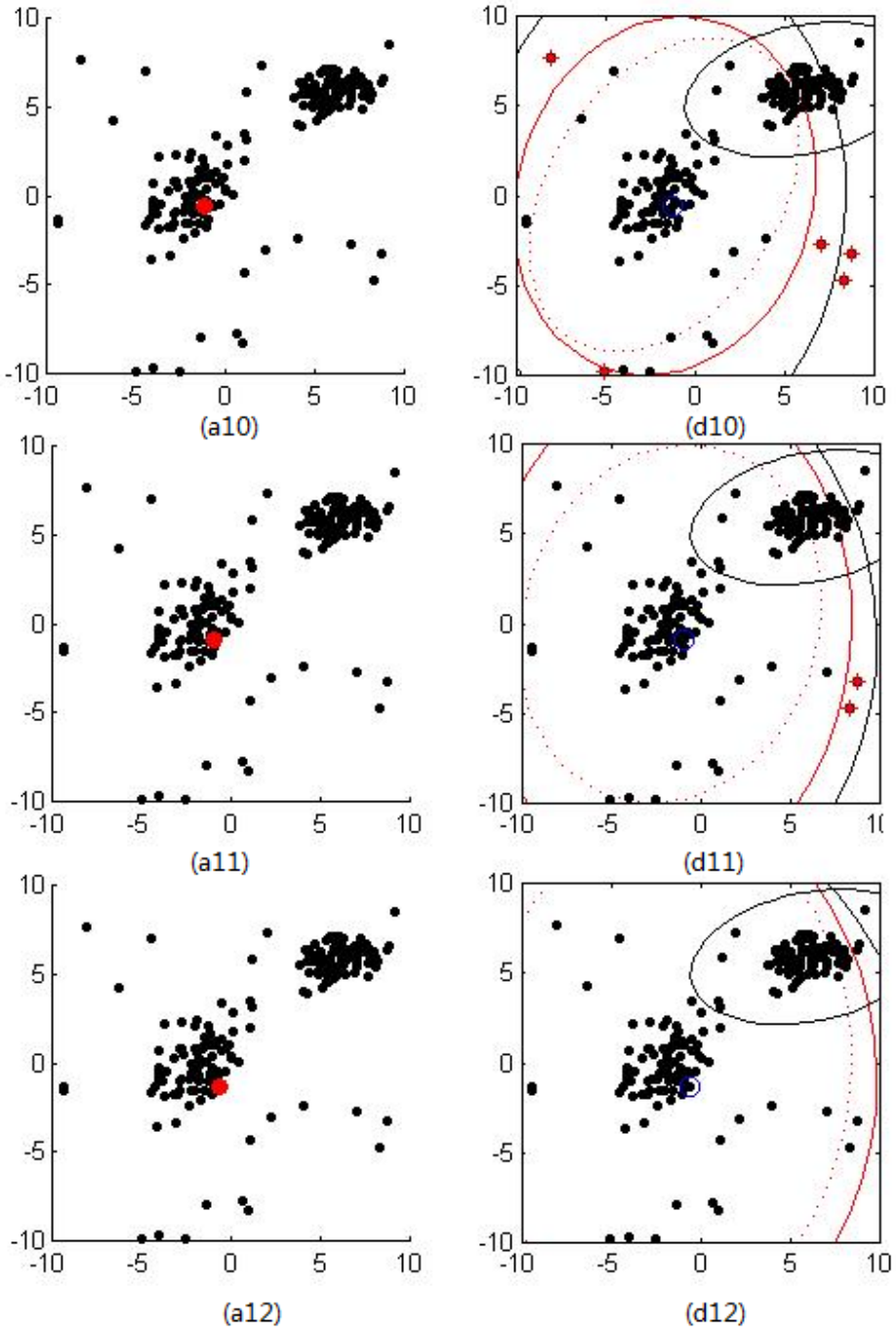


Figure 4.8 Snapshots of the GMM update. A new point (red) is added. The black ellipsoids are used to illustrate the boundary of the Gaussian ellipses, with the red solid-lined ellipsoids illustrating the boundary of the ellipses before the update and the red dashed ellipsoids illustrating the boundary of even earlier ellipses. (a1) - (a18): original dataset. (b1) - (b18): updates using T_d = 2. The system is no longer able to adjust to the gradual changes of the behavior pattern since (b4). (c1) - (c18): Updates using T_d = 3. (d1) - (d12): Updates using T_d = 5.

In the next experiment, datasets with different dimensions are updated using the GMM with different distance threshold settings. Each dataset contains 3 Gaussian components and random noise generated by uniform distribution. The GMM is initialized using the labels from generating the data, so the influence of bad initialization of the GMM can be avoided. The results of the updates are compared to the labels we get from generating the data and then the accuracies are calculated. The data generated by the Gaussian distribution are labeled as normal and the feature vectors generated by the uniform-distributed random number generator are labeled as anomaly if and only if they are away from the dense regions of the data.

Table 4.4 shows results of the GMM updates averaging over 15 trials. In each trial, the data in the initial window is randomly generated with the same parameter setting (same means and covariance matrices for the three dense regions of the data and noise data generated from uniform distribution). The new data points are added in a way that they may stay in a certain clusters for a while, travel between clusters, deviate from the clusters or just be a noise data point away from the clusters. Small random noise is also added to the new data to make its trajectory more unpredictable so that each trial the new points' behavior is totally different.

Table 4.4 Accuracy results of the GMM updates using distance threshold T_d = 2, 3, 4, 5, 6 on 2-dimensional, 3dimensional and 10-dimensional datasets averaging over 15 trials

Distance Threshold $\mathbf{T_d}$	2	3	4	5	6
2-dimension dataset	88.30%	94.58%	92.89%	92.08%	81.70%
3-dimension dataset	70.00%	96.41%	96.67%	96.03%	71.54%
10-dimension dataset	37.96%	41.97%	66.20%	99.40%	99.90%

According to the table, the distance threshold $\mathbf{T_d}$ that gives the best performance grows as the dimension of the dataset becomes higher. This is easy to comprehend as the dimension of the data goes higher, the sparser it will become, which requires a higher distance threshold $\mathbf{T_d}$.

Although using $\mathbf{T_d}$ = 3 for the two-dimension data seems to be a good fit for the GMM, in the early illness detection case, the sensor data used will be of much higher dimension in order to incorporate more information. What's more, larger distance threshold need to be used when using the fuzzy-2 method to compute the mean and covariance matrices of the GMM. Also, since different distance thresholds yield different sensitivity for the alert algorithm, more effort should be put into choosing the distance threshold when using the embedded sensor data for the alert algorithm.

4.4 Detection of emergence of a new cluster

Since the full algorithm involves determination of potential new normal patterns within the flagged data, we now examine that component. In this experiment, the PCM was used on four sets of data with the number of clusters set as 1. The membership values from the PCM output are compared against membership threshold value $\mathbf{T_o}$ = 0.1, 0.4, 0.7. Vectors with memberships larger than the threshold value are plotted as green dots to represent the cluster. Dataset 8, dataset 9 and dataset 10 are Gaussians among noisy data while dataset 11 is randomly distributed data.

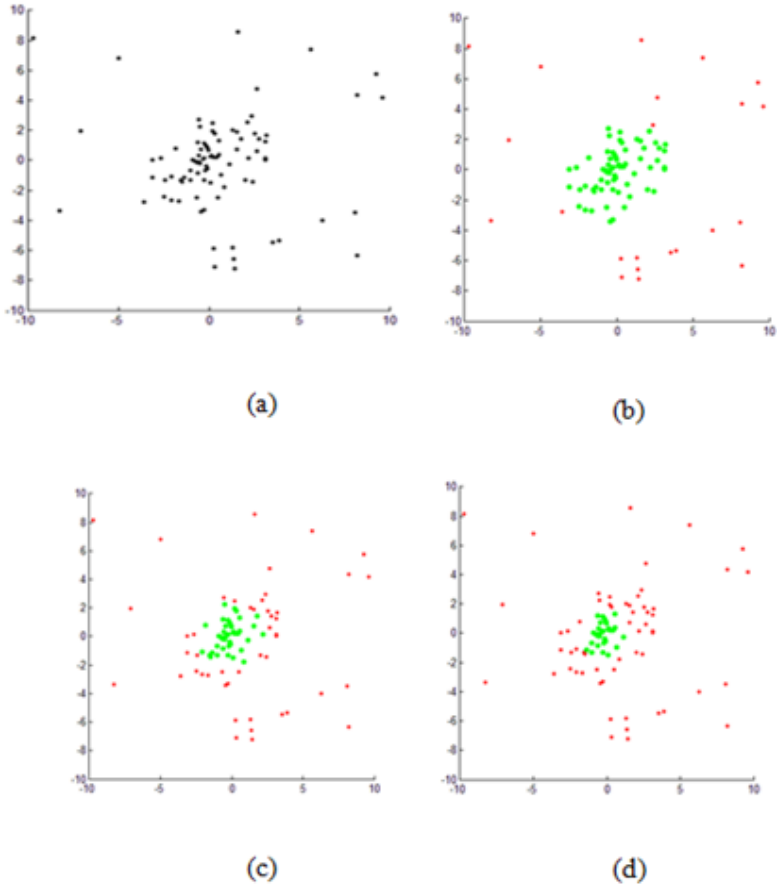
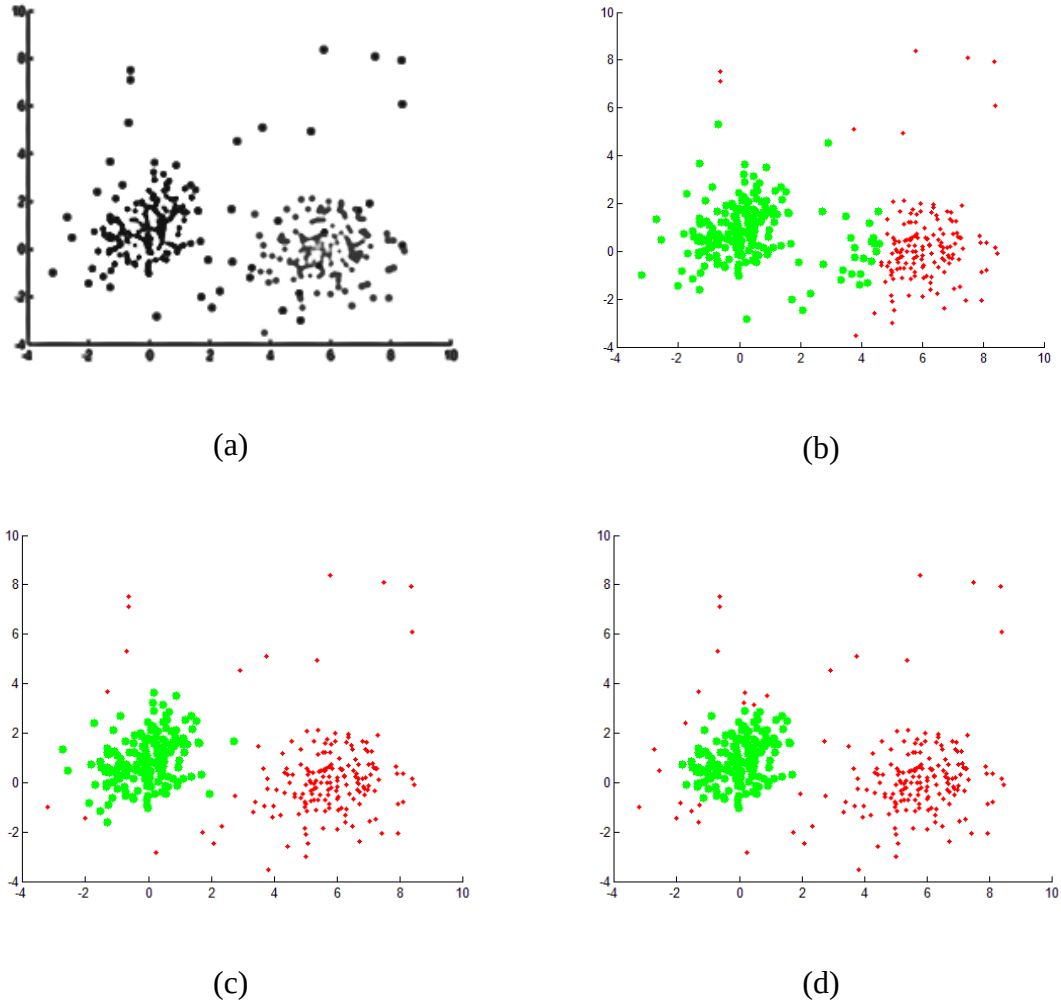


Figure 4.9 One cluster recognition using different membership threshold T_o. (a) Original dataset 8. (b) Clustering result with T_o=0.1. (c) Clustering result with T_o=0.4. (d) Clustering result with T_o=0.7.

We can see from dataset 8 in Figure 4.9, the cluster recognized by the PCM becomes smaller in size when we increase the value of membership threshold T_o. When T_o=0.1, the result is closest to the original data distribution.

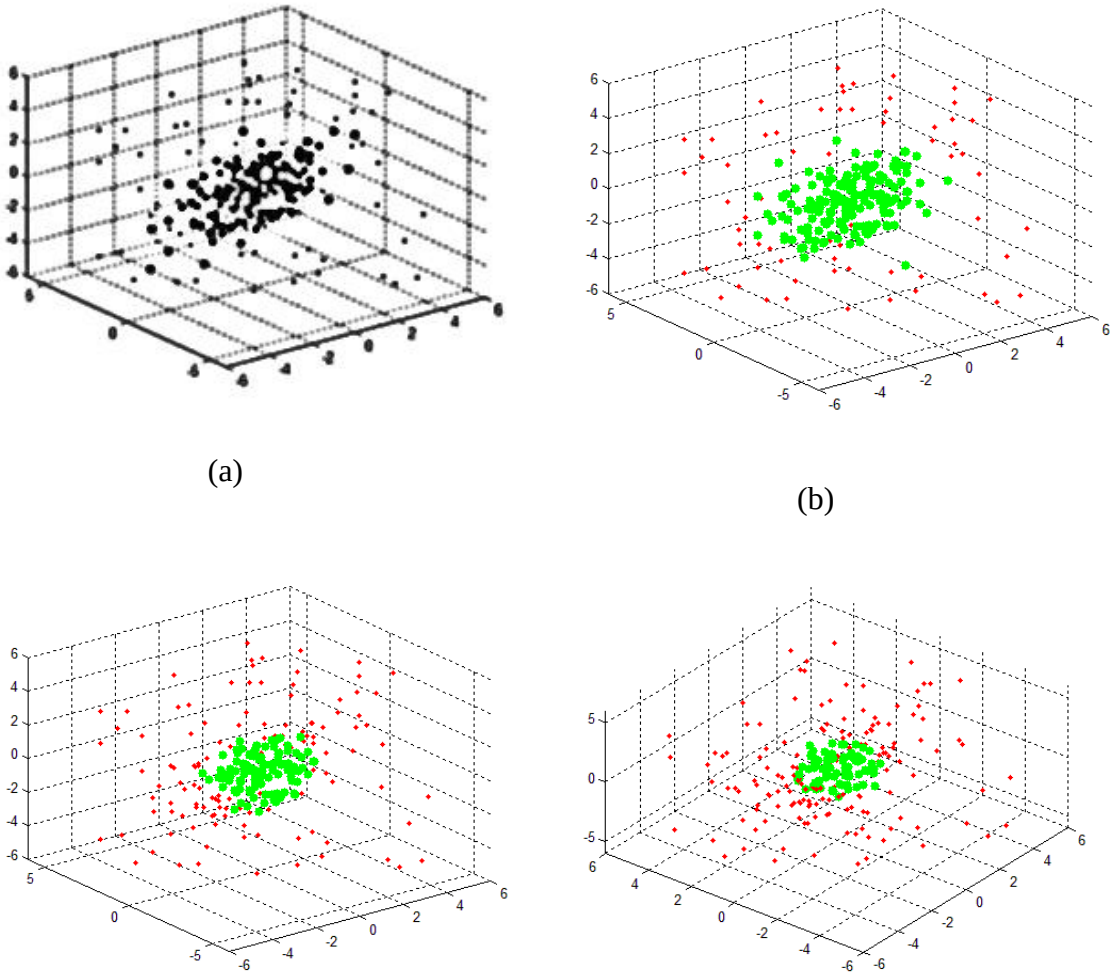


(a) (b)

(c) (d)

Figure 4.10 One cluster recognition using different membership threshold T_o. (a) Original dataset 9. (b) Clustering result with T_o=0.1. (c) Clustering result with T_o=0.4. (d) Clustering result with T_o=0.7.

In dataset 9, there are two natural clusters instead of one comparing to the previous dataset. We can see from Figure 4.10 that when T_o=0.1, the cluster recognized by the PCM involves some data points from the other natural cluster while using T_o=0.4 leads to a better clustering result. The membership threshold T_o that can best separate the one natural cluster from the data might be unique for each case.



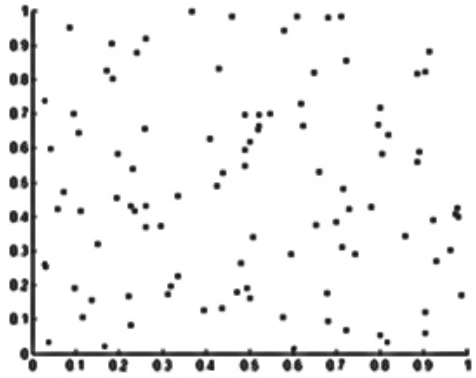
(a) (b)

(c) (d)

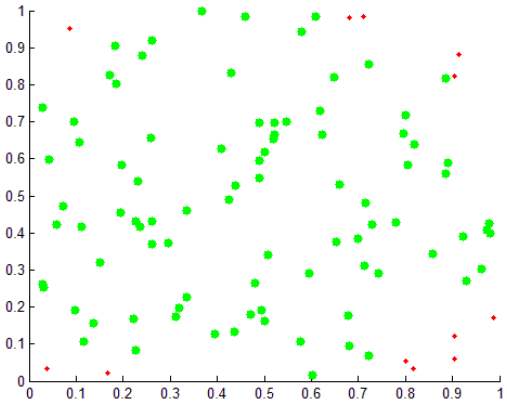
Figure 4.11 One cluster recognition using different membership threshold T_o.

(a) Original dataset 10. (b) Clustering result with T_o=0.1. (c) Clustering result with T_o=0.4.

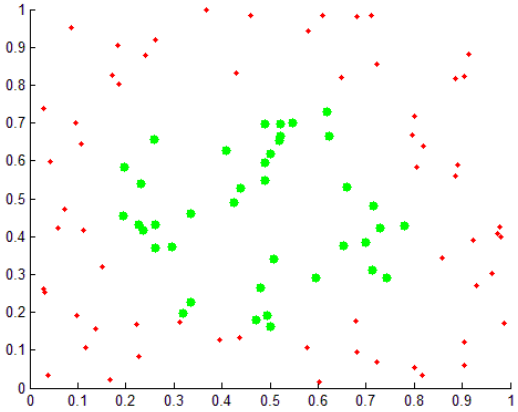
(d) Clustering result with T_o=0.7.



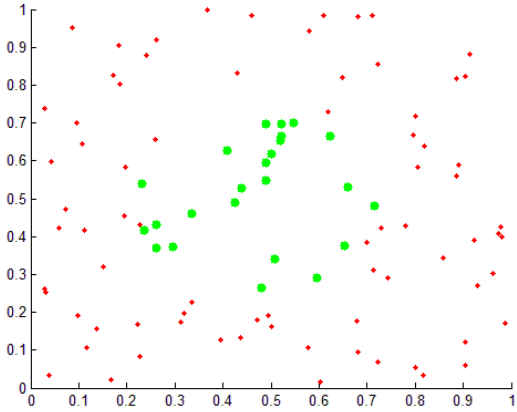
(a)



(b)



(c)



(d)

Figure 4.12 One cluster recognition using different membership threshold T_o.

(a) Original dataset 11. (b) Clustering result with T_o=0.1. (c) Clustering result with T_o=0.4.

(d) Clustering result with T_o=0.7.

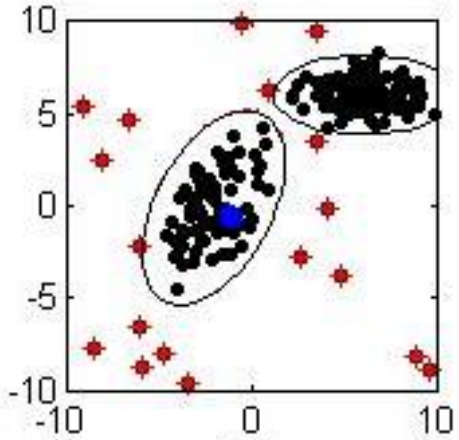
Clearly, the larger the membership threshold $\mathbf{T_o}$ is, the cluster recognized by the PCM using the membership threshold $\mathbf{T_o}$ will be smaller in size. From Figure 4.9 to Figure 4.12 we can conclude that using membership threshold $\mathbf{T_o}$ = 0.1 is good enough to distinguish the good cluster from the data most of the time. Yet this threshold value will not apply to every situation as we can see in Figure 4.10: using threshold $\mathbf{T_o}$ = 0.1 would involve

some of non-relevant vectors within the new cluster. In the eldercare situation it would mean a wrong estimation of the new normal behavior pattern and a possible miss of a real alert. It is recommended to use threshold $0.1 < \mathbf{T_o} < 0.5$, which would be a safe and more conservative choice to find the new cluster.

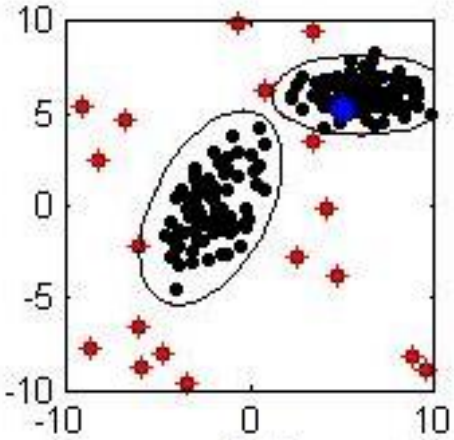
4.5 Experiments of the whole system on synthetic datasets

In this experiment the whole system is tested using synthetic datasets. The datasets are designed to simulate system noise, gradual changes in baseline, sudden changes, emergence of new normal patterns and other situations that are likely to happen in the embedded sensor data. More information regarding the generation of the synthetic datasets can be found in the appendix.

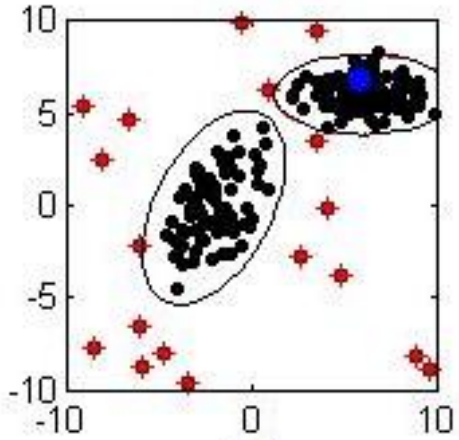
The first set of synthetic data is made of two Gaussians and random noise in two-dimension space. We choose $\mathbf{T_d} = 3$, $\mathbf{T_n} = 0.06$, $\mathbf{T_o} = 0.4$. A total of 200 feature vectors are used to initialize the model, as shown in Figure 4.13(i), with the two black ellipsoids showing the normal activity pattern, and the red asterisks are the anomalies flagged by the algorithm. The blue marker shows the most recent data entry. To better observe the changes in the Gaussian clouds, a red solid-lined ellipsoid and a red dotted-lined ellipsoid are drawn to indicate the forms of the previous Gaussians.



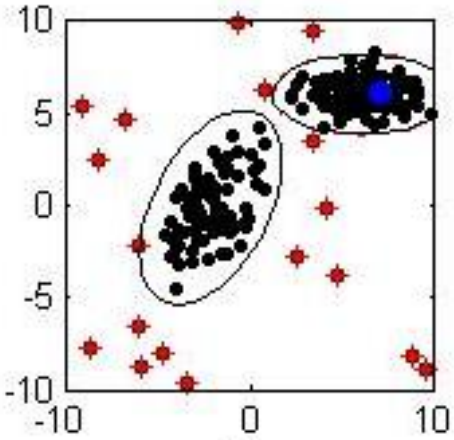
(vii)



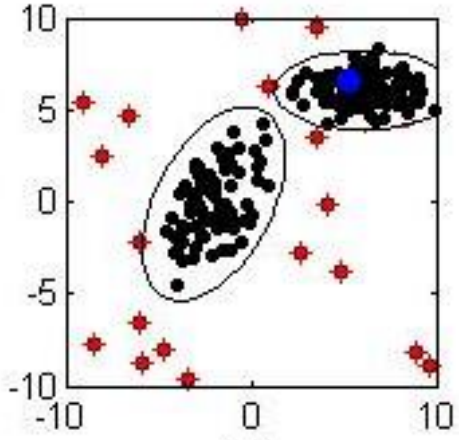
(viii)



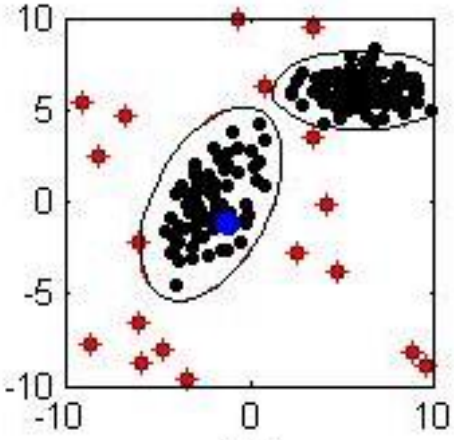
(ix)



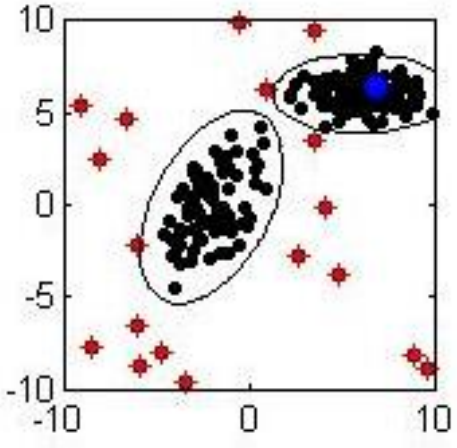
(x)



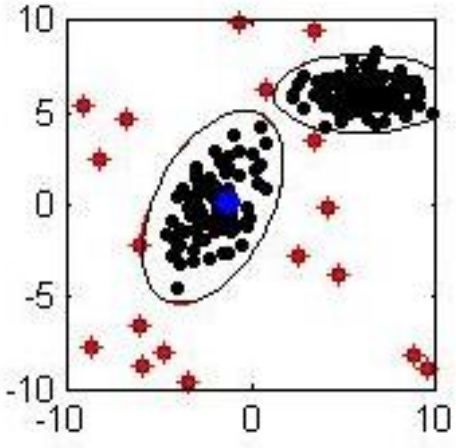
(xi)



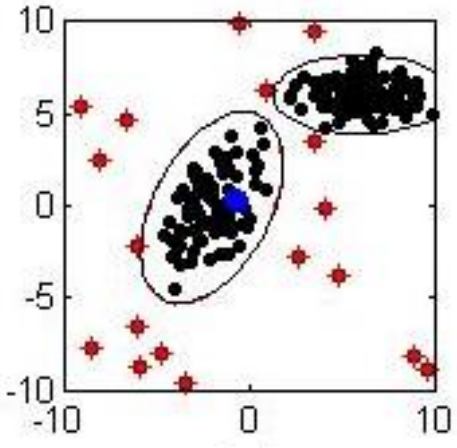
(xii)



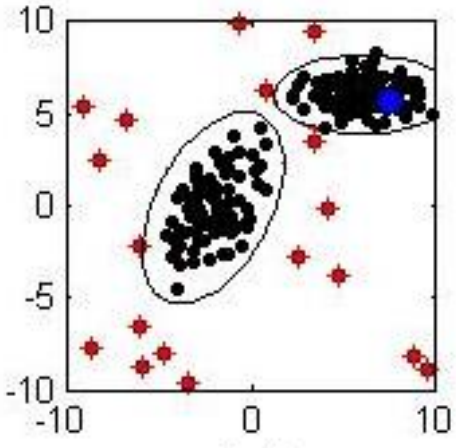
(xiii)



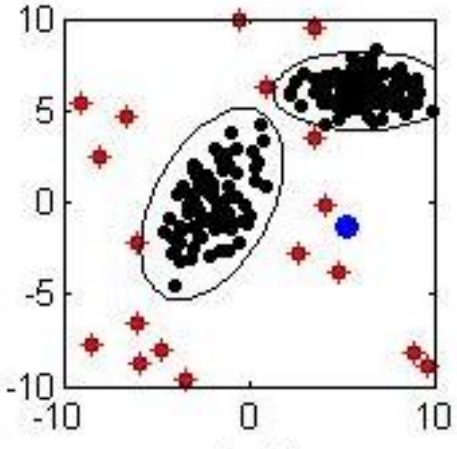
(xiv)



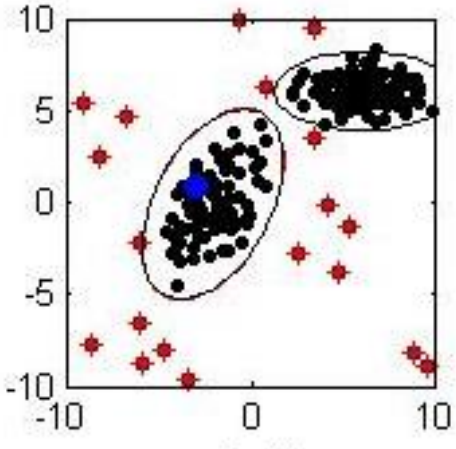
(xv)



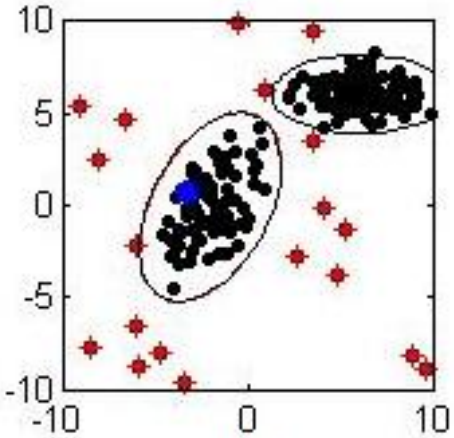
(xvi)



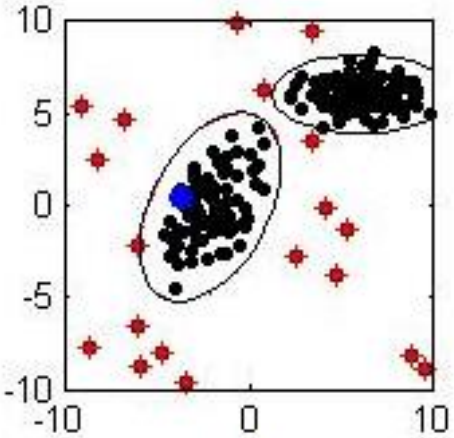
(xvii)



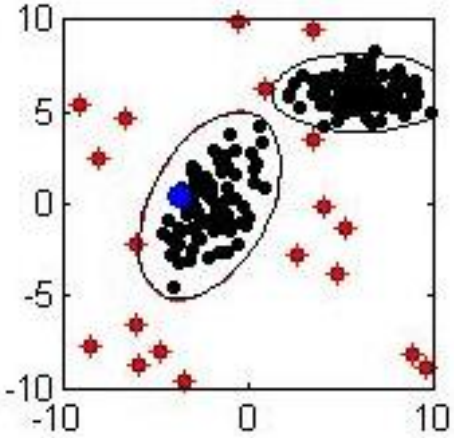
(xviii)



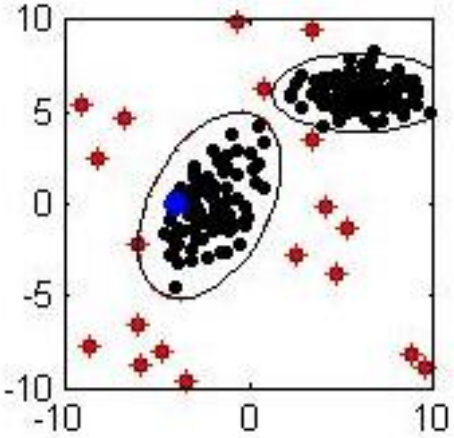
(xix)



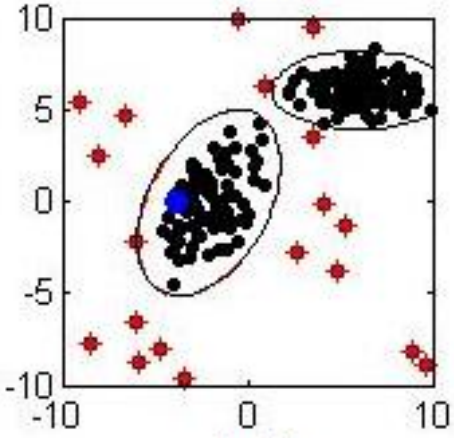
(xx)



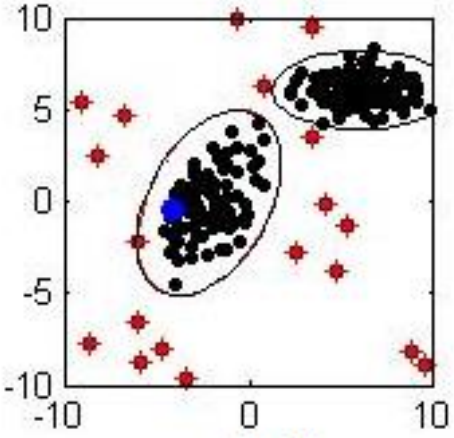
(xxi)



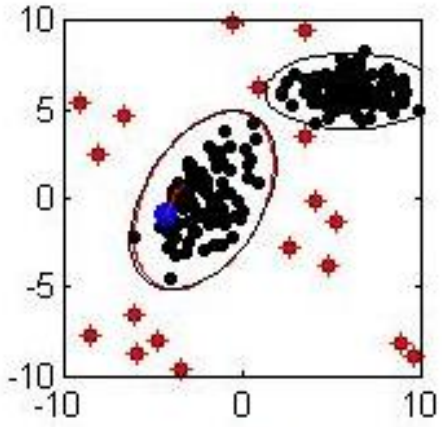
(xxii)



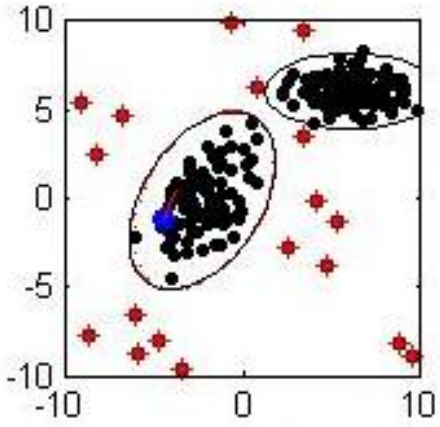
(xxiii)



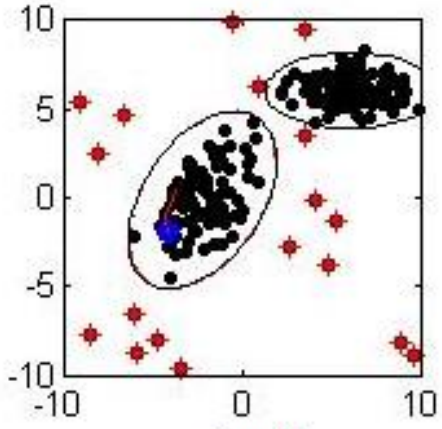
(xxiv)



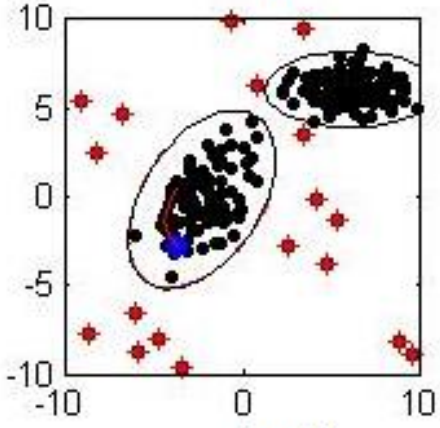
(xxv)



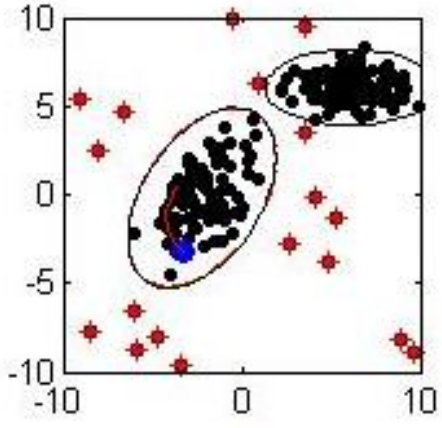
(xxvi)



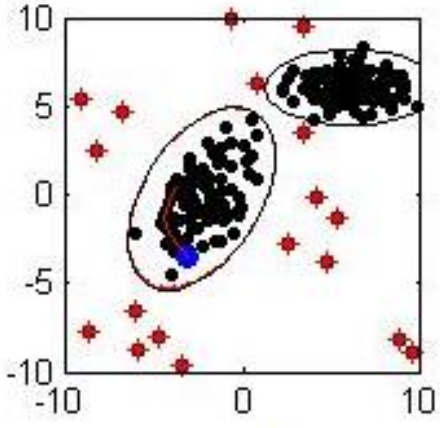
(xxvii)



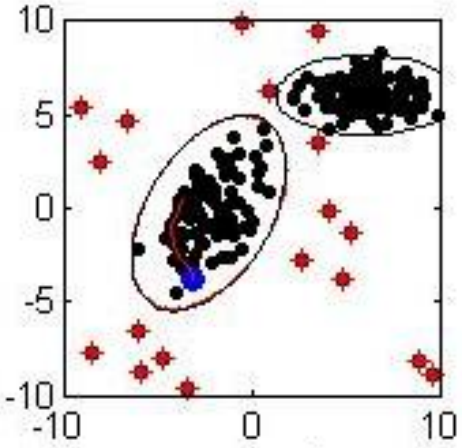
(xxviii)



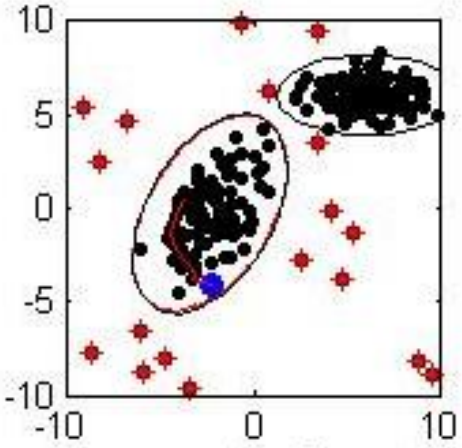
(xxix)



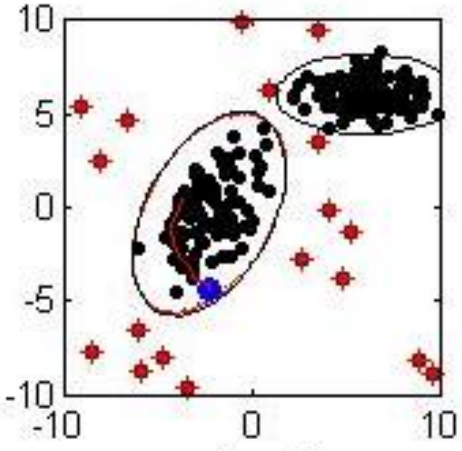
(xxx)



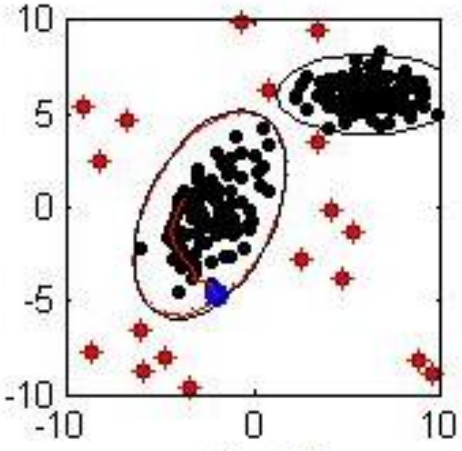
(xxxi)



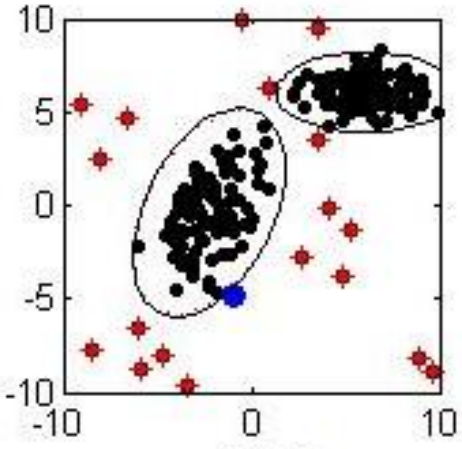
(xxxii)



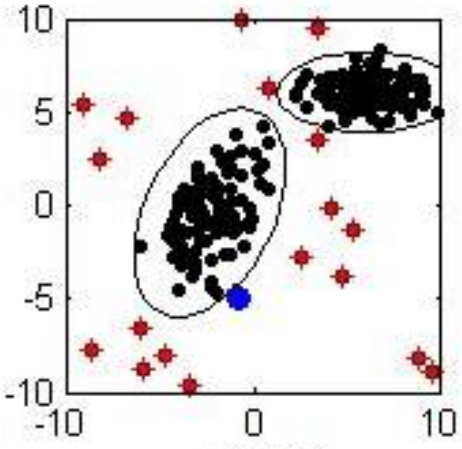
(xxxiii)



(xxxiv)



(xxxv)



(xxxvi)

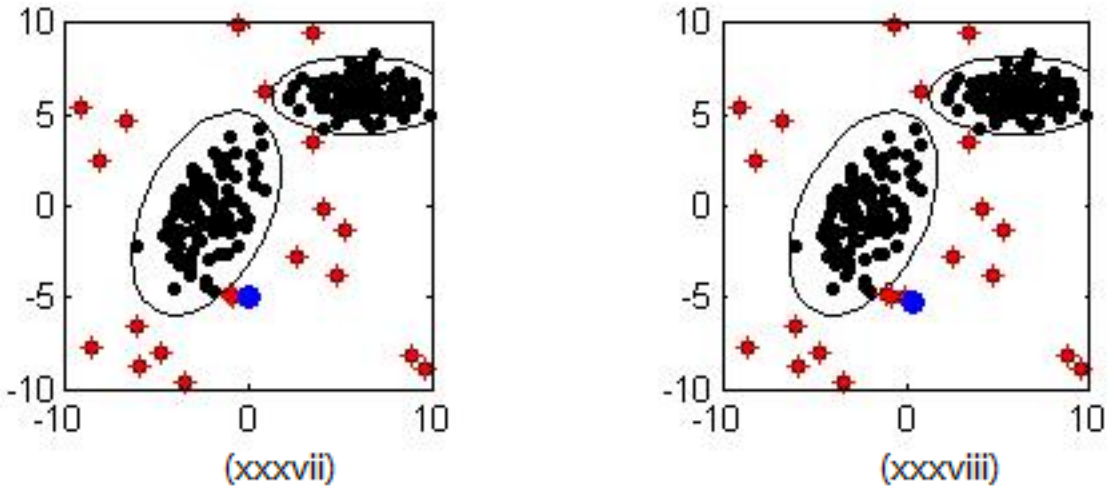
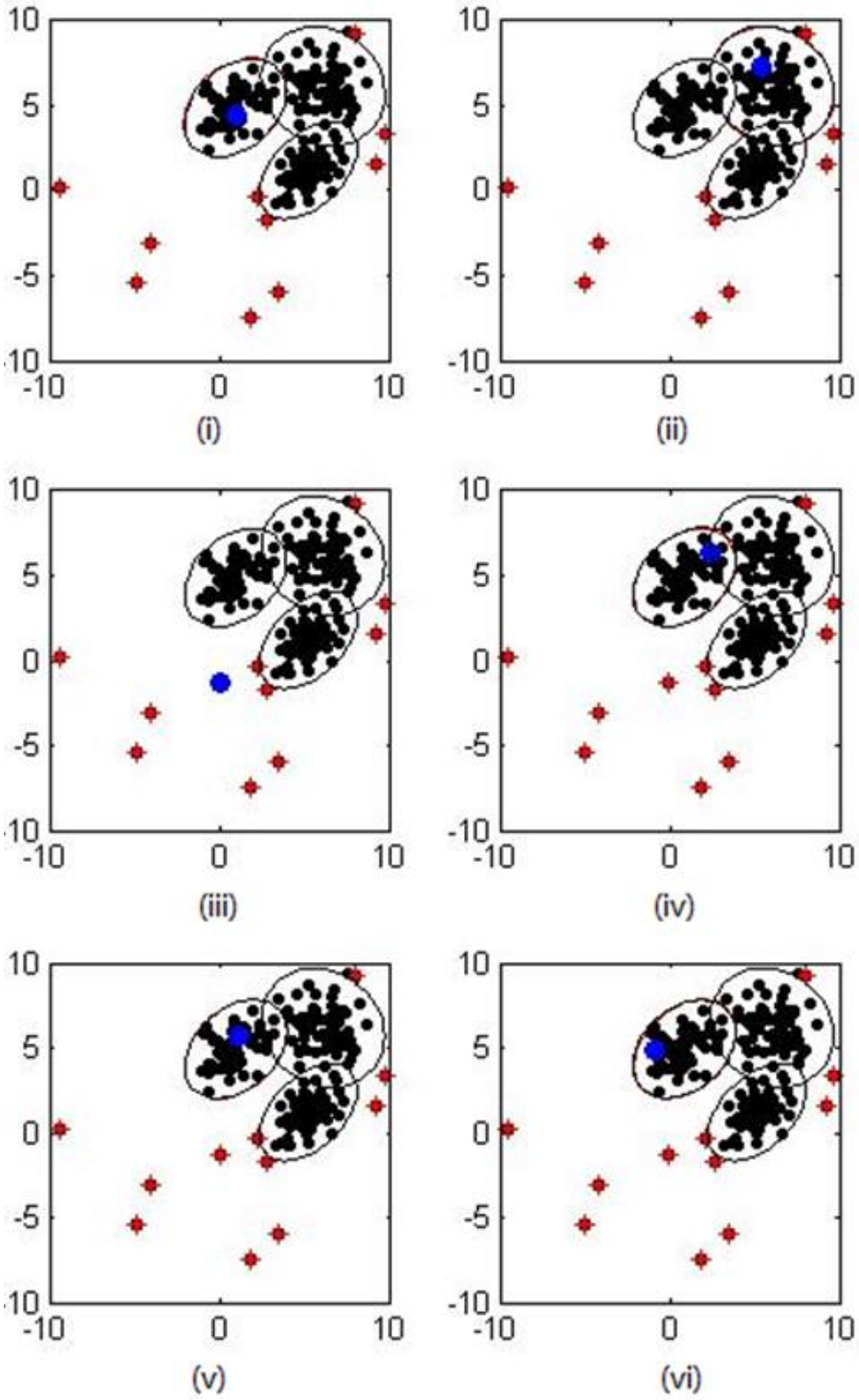


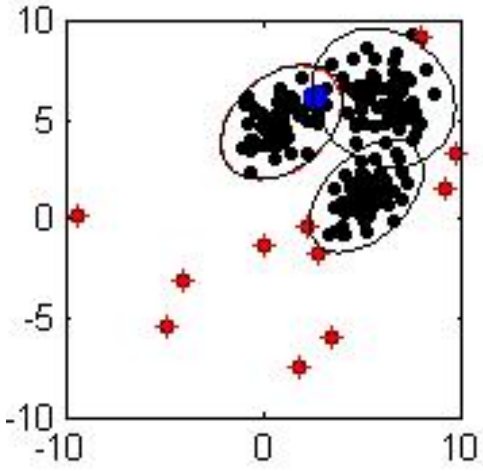
Figure 4.13 Movie clip showing the update of the GMM after initialization. The red marker shows the most recent data entry, while the blue dots are anomalies flagged by the algorithm and the black ellipsoids shows the normal pattern. From (xxv) to (xxxiv), the new data keep deviating from the center of the Gaussian, and since (xxxv), the Gaussian can no longer tolerate the changes and an alert is fired.

The algorithm first detects an outlier and an alert is fired. Then the GMM is updated with the data points indicating the normal behaviors, as shown in Figure 4.13(ii) – Figure 4.13(xxiv). Since the new data points keep representing the same behavior pattern and all the data belong to the GMM are used to update the parameters of the GMM, slight change is made to the corresponding Gaussian, as can be seen from almost indiscernible red ellipsoids.

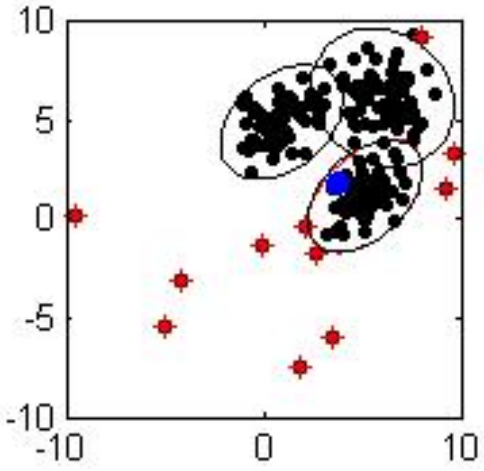
Then the new data entries show that the behavior starts to deviate from the normal pattern, as shown in Figure 4.13(xxv) – Figure 4.13(xxxiv). The new entry stays near the center of the Gaussian, and then starts heading towards the boundary of the Gaussian and "pushes" the boundary, as can be seen from the red trajectory. Alerts are fired as the GMM can no longer handle the changes in behavior, as shown in Figure 4.13 (xxxv) –

Figure 4.13(xxxx). The video version of this experiment can be viewed on http://youtu.be/4mGQYjFfTdM.

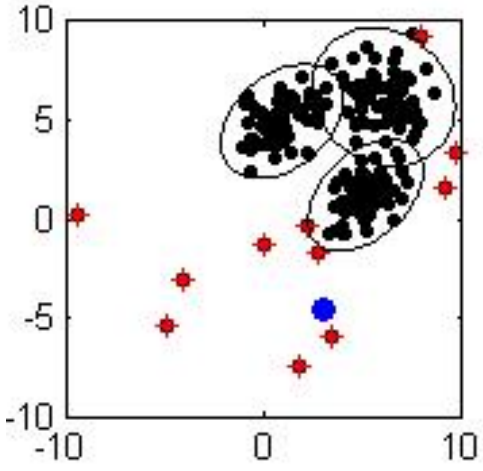




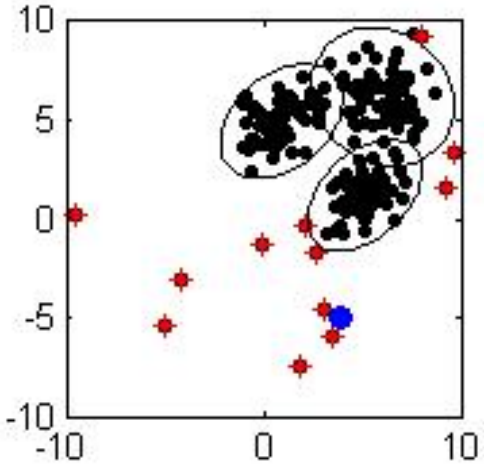
(vii)



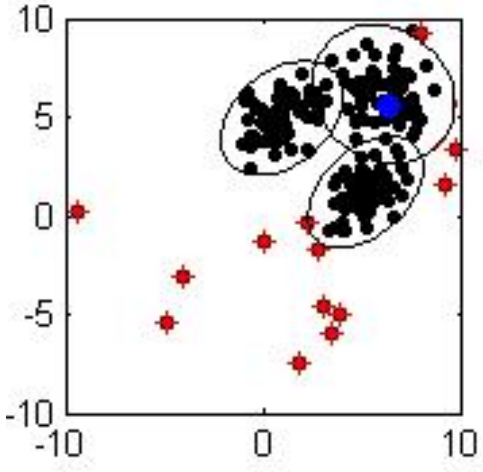
(viii)



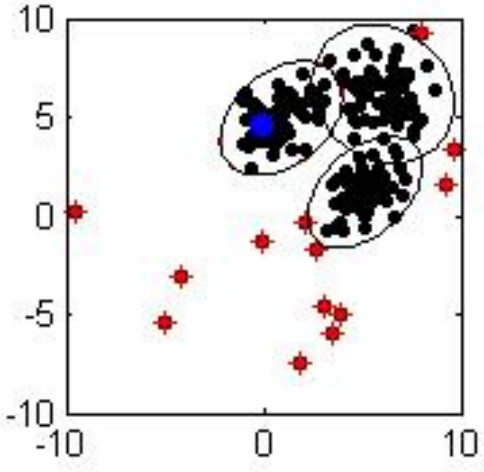
(ix)



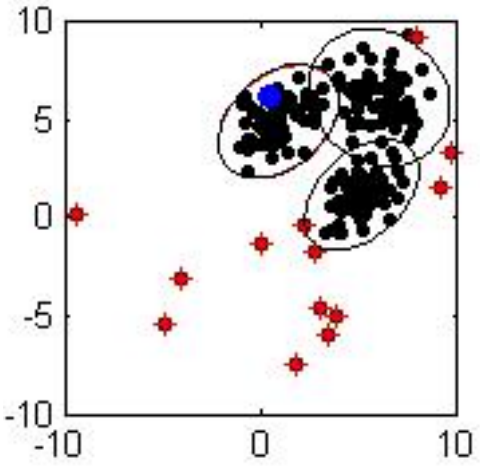
(x)



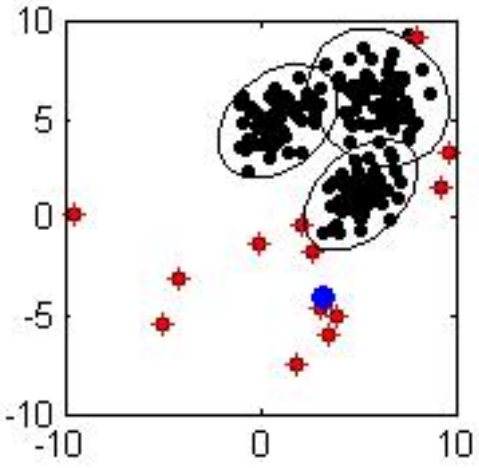
(xi)



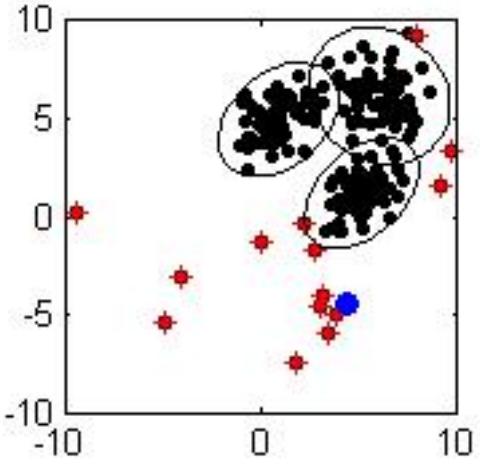
(xii)



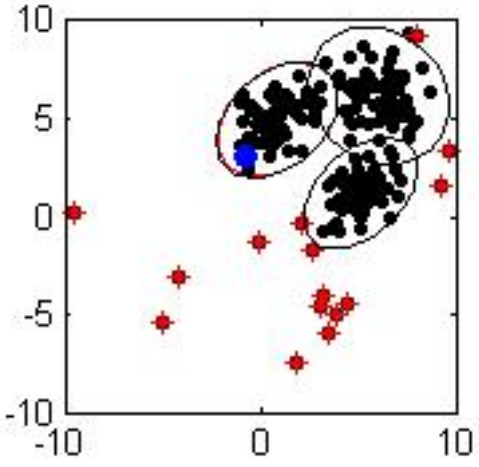
(xiii)



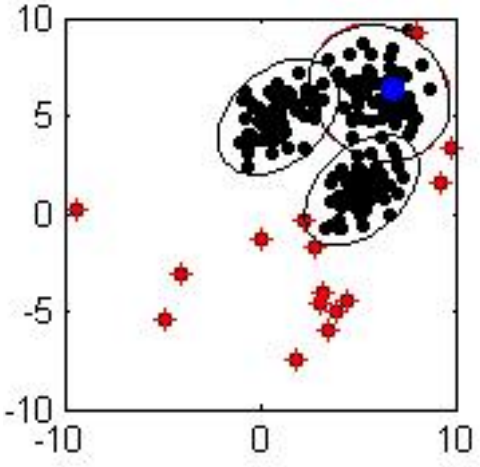
(xiv)



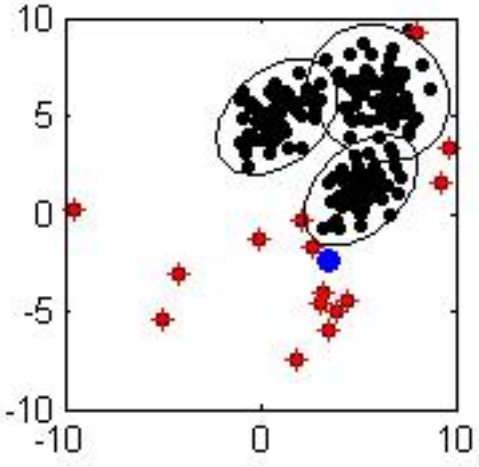
(xv)



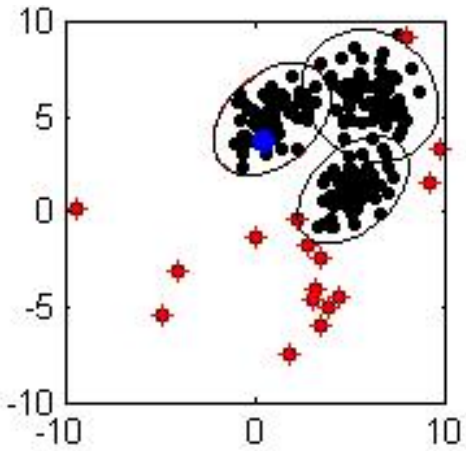
(xvi)



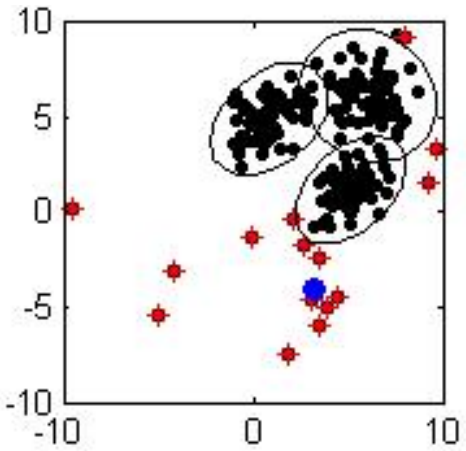
(xvii)



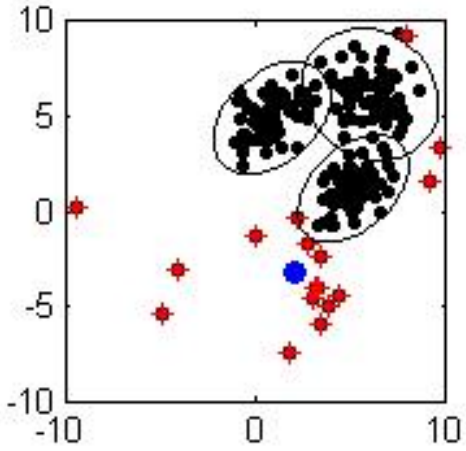
(xviii)



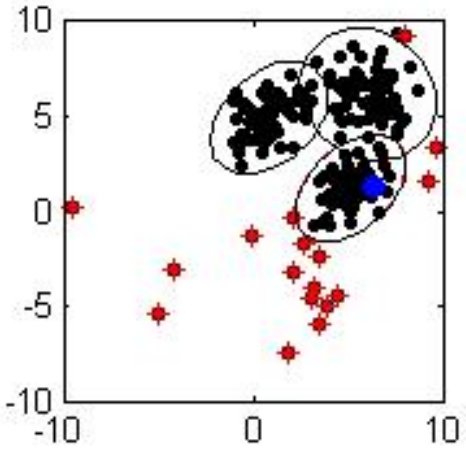
(xix)



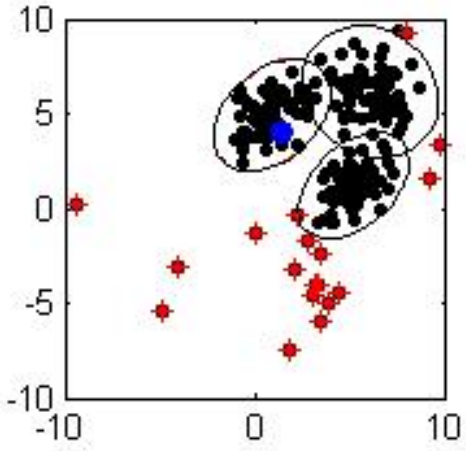
(xx)



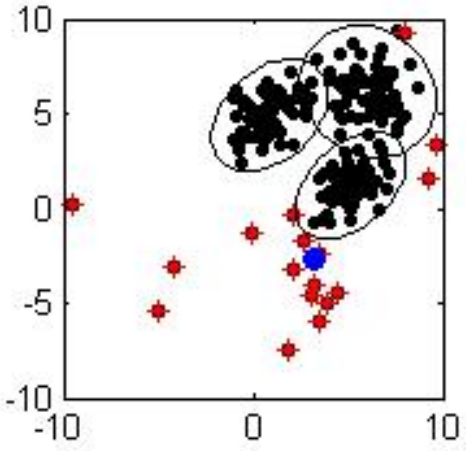
(xxi)



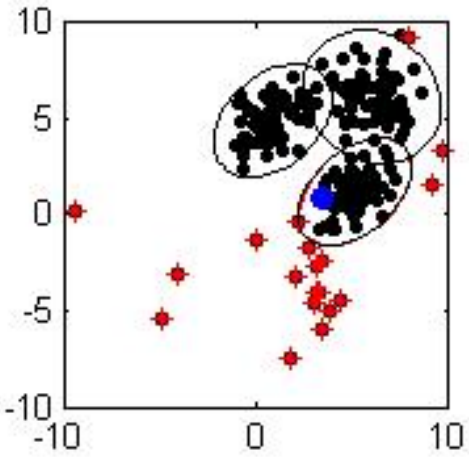
(xxii)



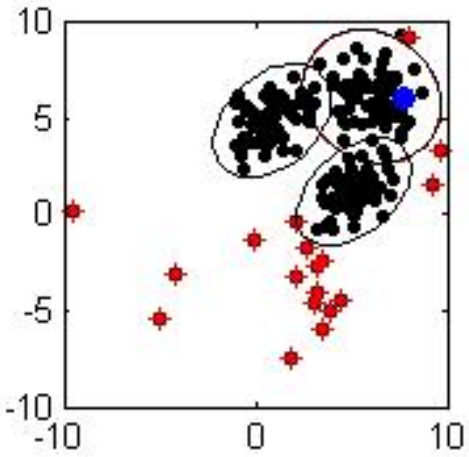
(xxiii)



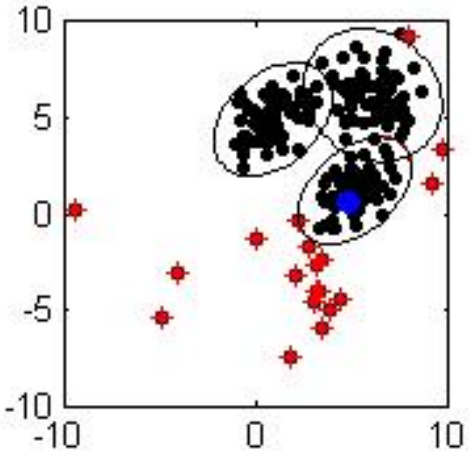
(xxiv)



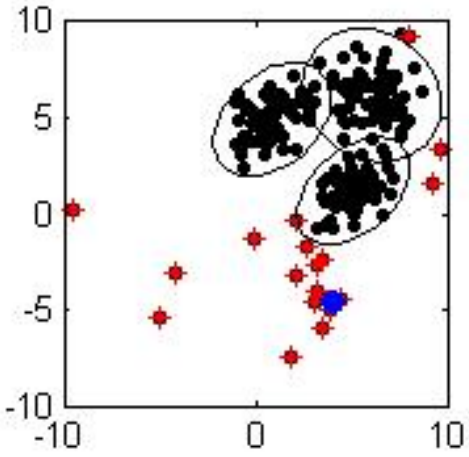
(xxv)



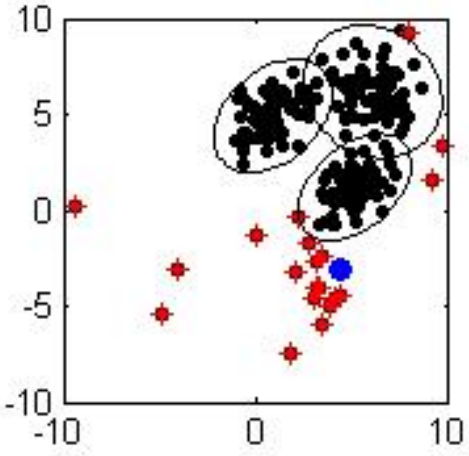
(xxvi)



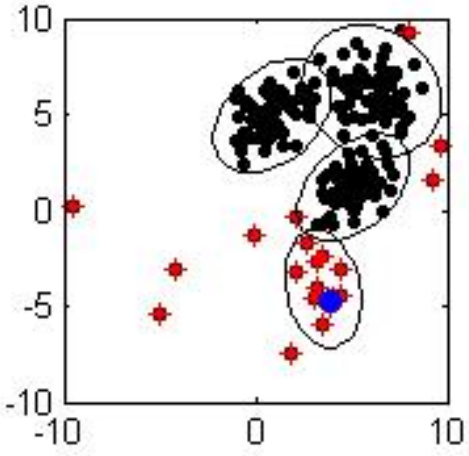
(xxvii)



(xxviii)



(xxix)



(xxx)

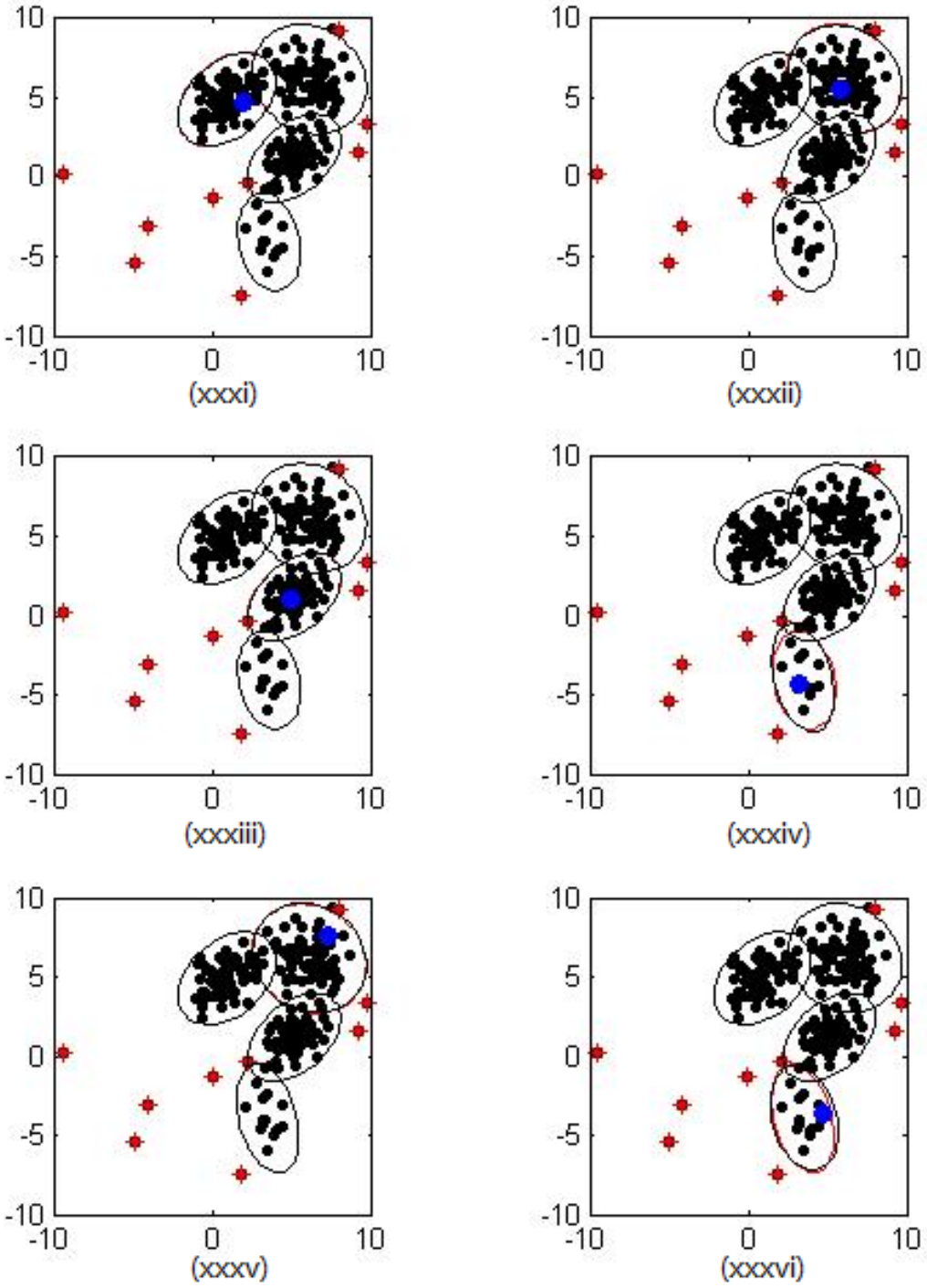
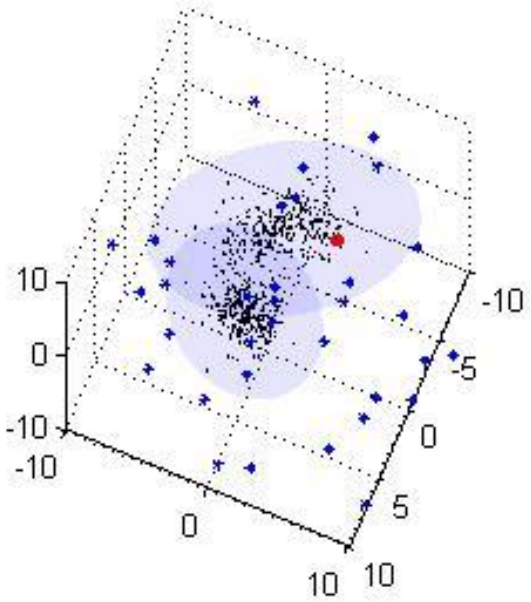


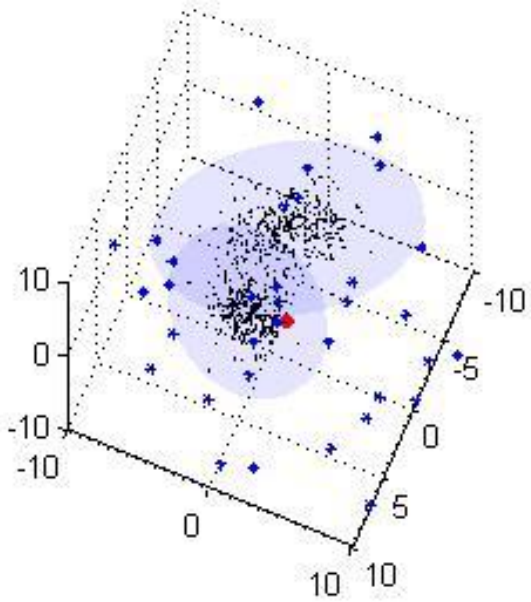
Figure 4.14 Movie clip showing the emergent of a new normal pattern. The blue marker shows the most recent data entry, while the red dots are anomalies and the black ellipsoids shows the normal patterns. Regular updating process is shown from (i) to (viii). From (ix) to (xxix), new data keeps showing similar behavior outside the GMM. The new black eclipse in (xxx) shows the new normal behavior detected by the PCM algorithm and then included into the GMM.

Figure 4.14(i) – (xxxvi) show another clip of the algorithm working on another dataset. After the update of 100 more data points, a new behavior pattern starts to show up as the new data entry keeps clustering at the lower right corner of the graph, as shown from Figure 4.14(ix) – (xxix). After several updates, the new cluster is recognized by the PCM algorithm, as shown in Figure 4.14 (xxx) and (xxxvi). A report would be sent to the medical professions and if the new cluster is proved to be a normal pattern, a new Gaussian will be spawned and incorporated into the GMM, as shown in Figure 4.14(xxxi). The video version of this experiment can be viewed on http://youtu.be/RyWQvDjY7ls.

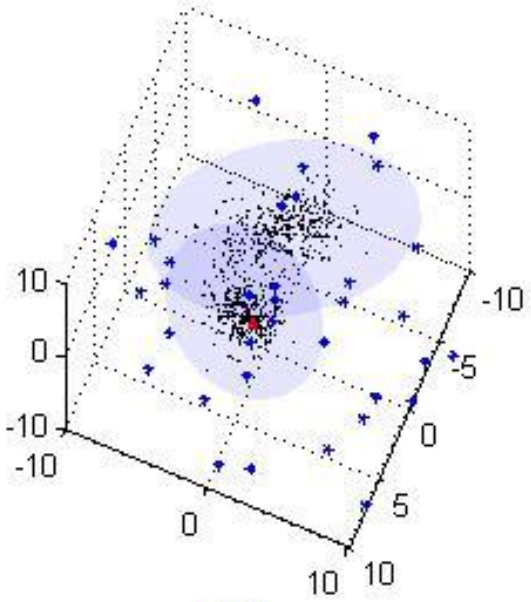
The third set of the synthetic data is a 10-dimension dataset and consists of two Gaussians. The initialization of the model uses 600 feature vectors. We give $\mathbf{T_d}$ = 6, Tn = 0.06, To = 0.4. The video version of this experiment can be viewed on http://youtu.be/3wJkmi1H4wQ. As shown in Figure 4.15, since the dataset is of 10 dimensions, the Principle Component Analysis (PCA) is used to convert the data into 3-dimensional for visualization. We can see that the new data keep updating one of the ellipsoid yet starting to deviate from the center of the cloud from Figure 4.15(ii)- (ix).


10
0
-10
-10
0
10
10
5
0
-5
-10

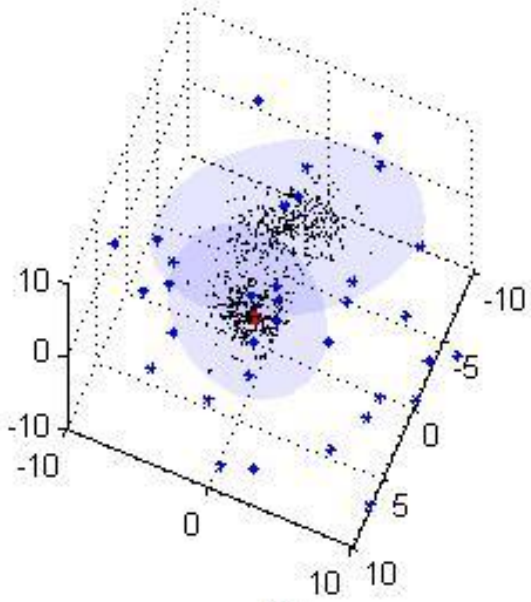
(i)


10
0
-10
-10
0
10
10
5
0
-5
-10

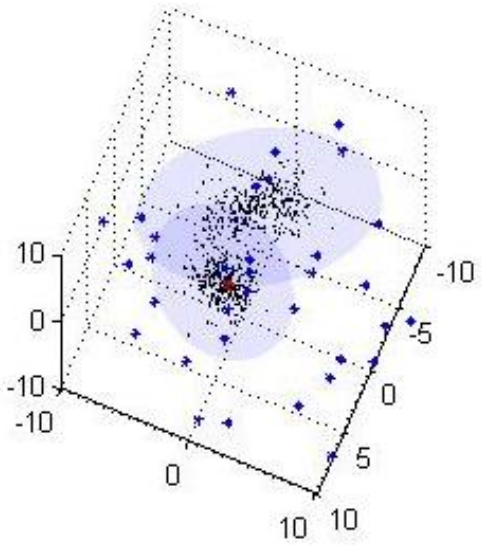
(ii)


10
0
-10
-10
0
10
10
5
0
-5
-10

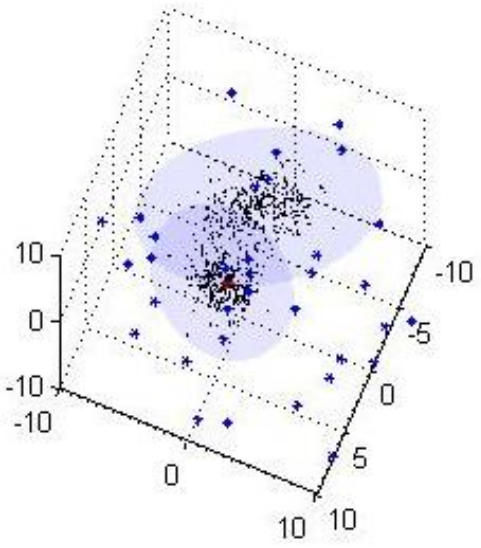
(iii)


10
0
-10
-10
0
10
10
5
0
-5
-10

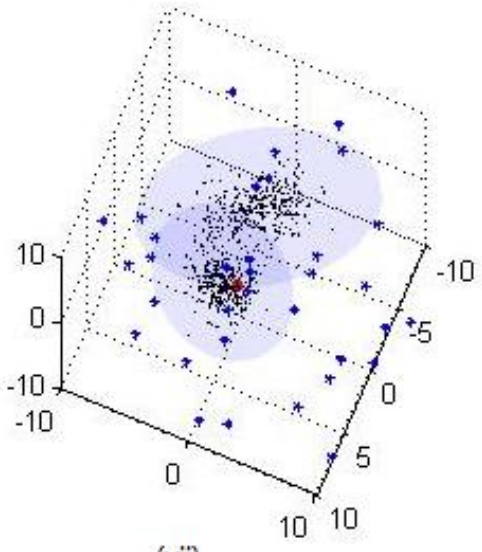
(iv)


10
0
-10
-10
0
10
10
5
0
-5
-10

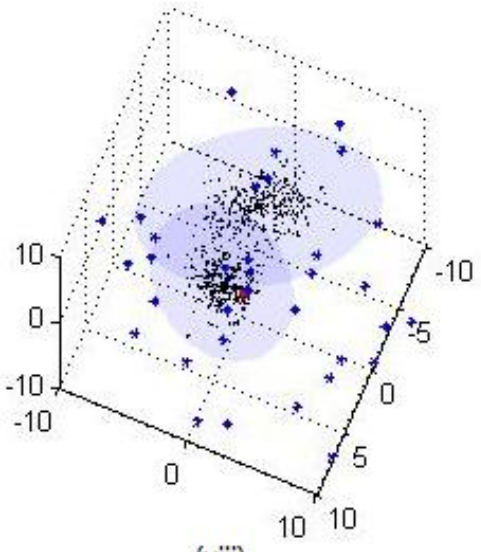
(v)


10
0
-10
-10
0
10
10
5
0
-5
-10

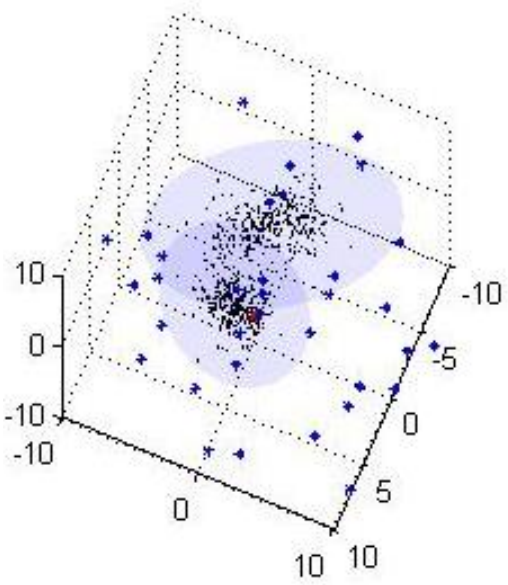
(vi)


10
0
-10
-10
0
10
10
5
0
-5
-10

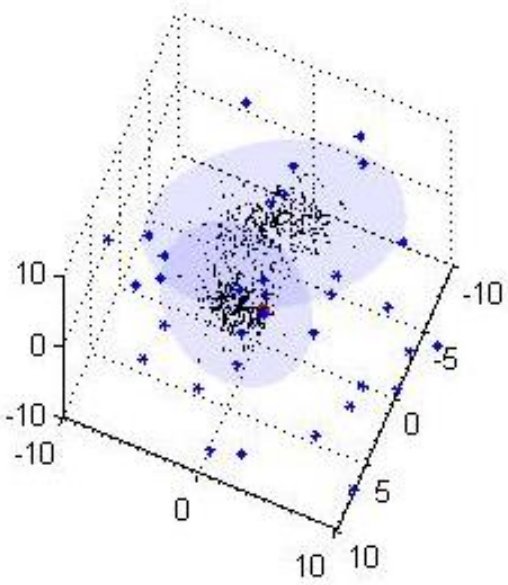
(vii)


10
0
-10
-10
0
10
10
5
0
-5
-10

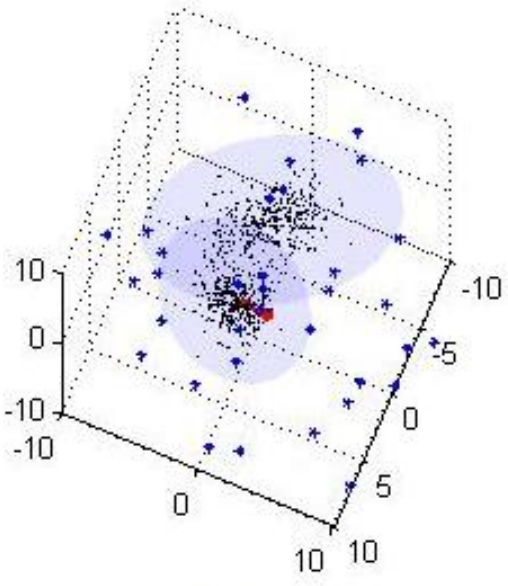
(viii)


10
0
-10
-10
0
10
10
5
0
-5
-10

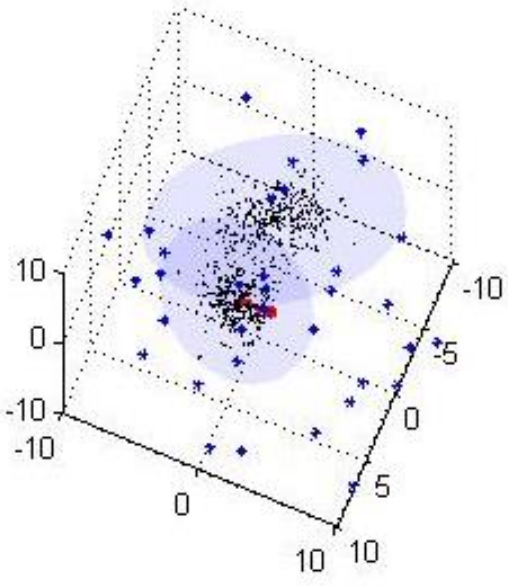
(ix)


10
0
-10
-10
0
10
10
5
0
-5
-10

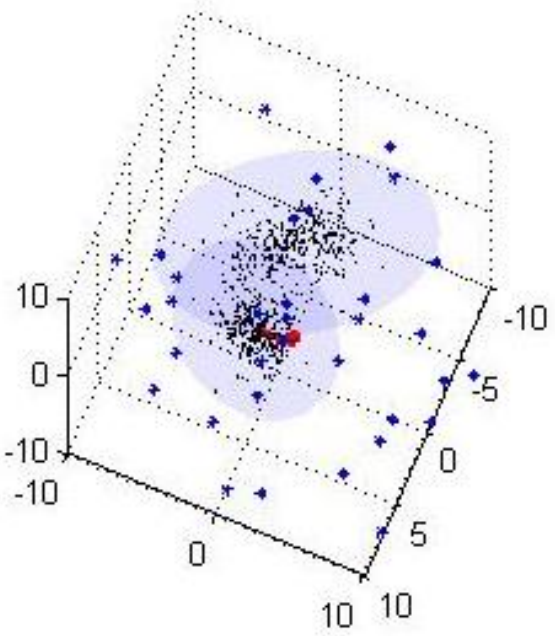
(x)


10
0
-10
-10
0
10
10
5
0
-5
-10

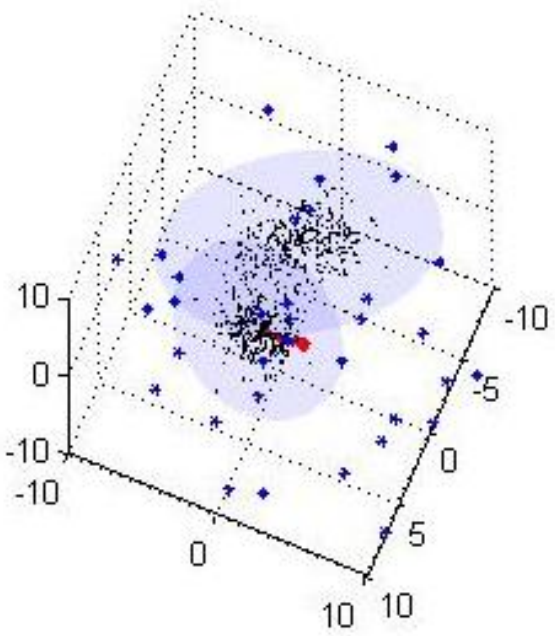
(xi)


10
0
-10
-10
0
10
10
5
0
-5
-10

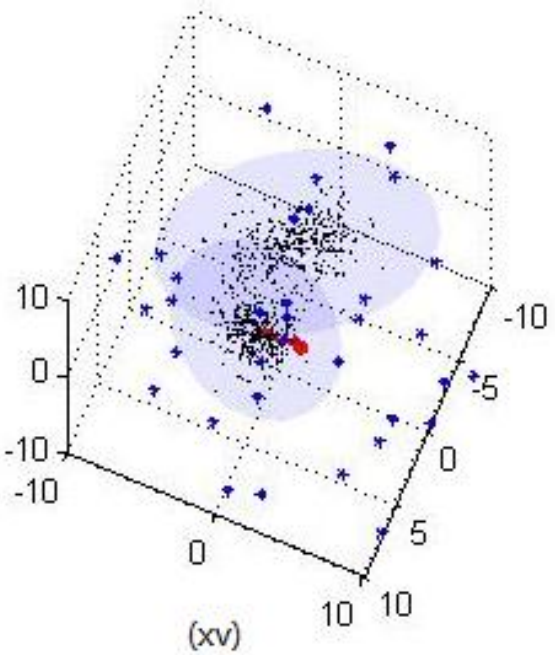
(xii)



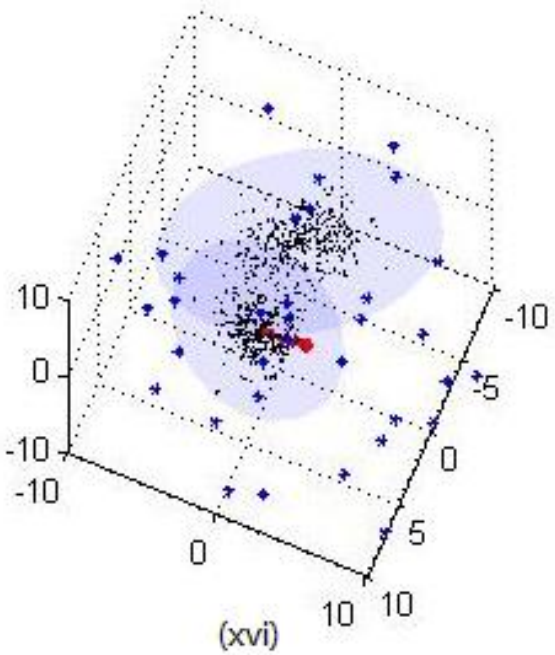
(xiii)



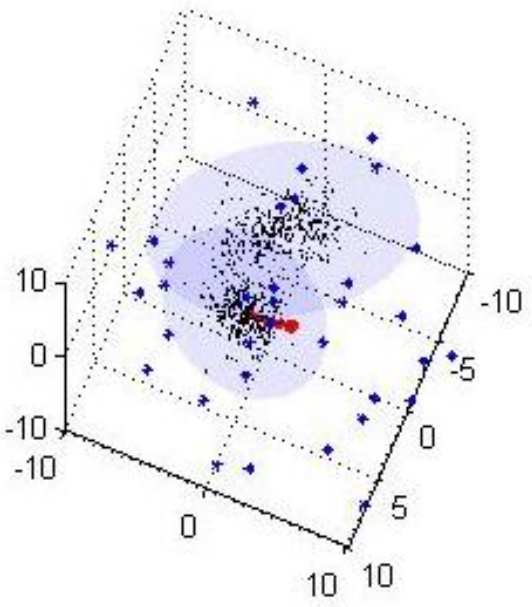
(xiv)



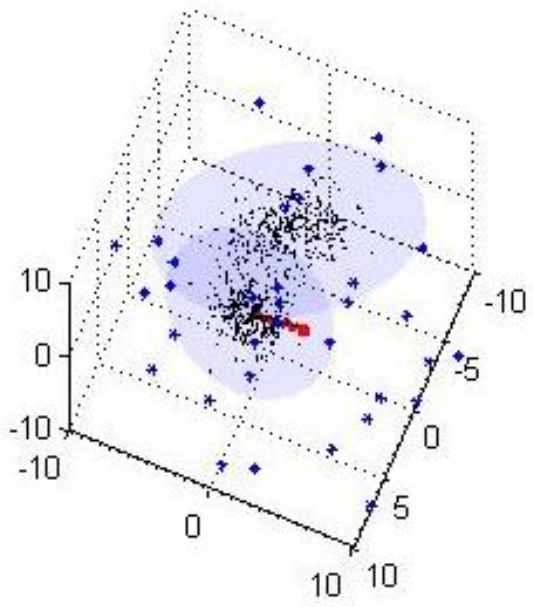
(xv)



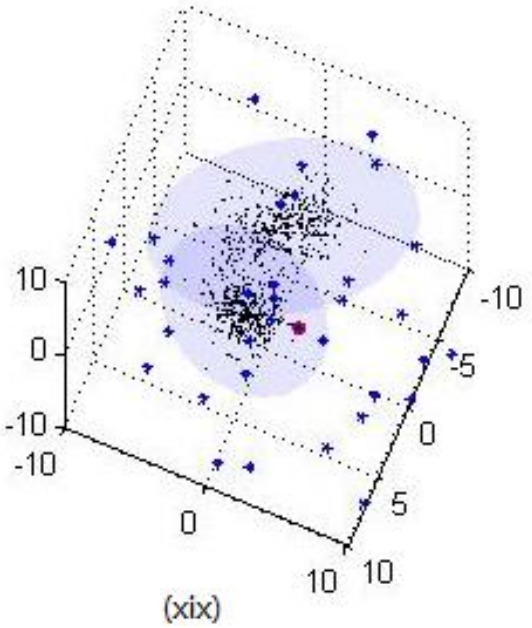
(xvi)



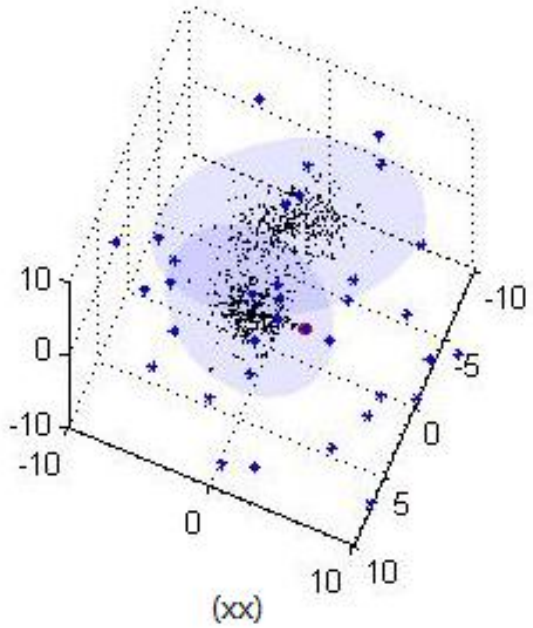
(xvii)



(xviii)



(xix)



(xx)

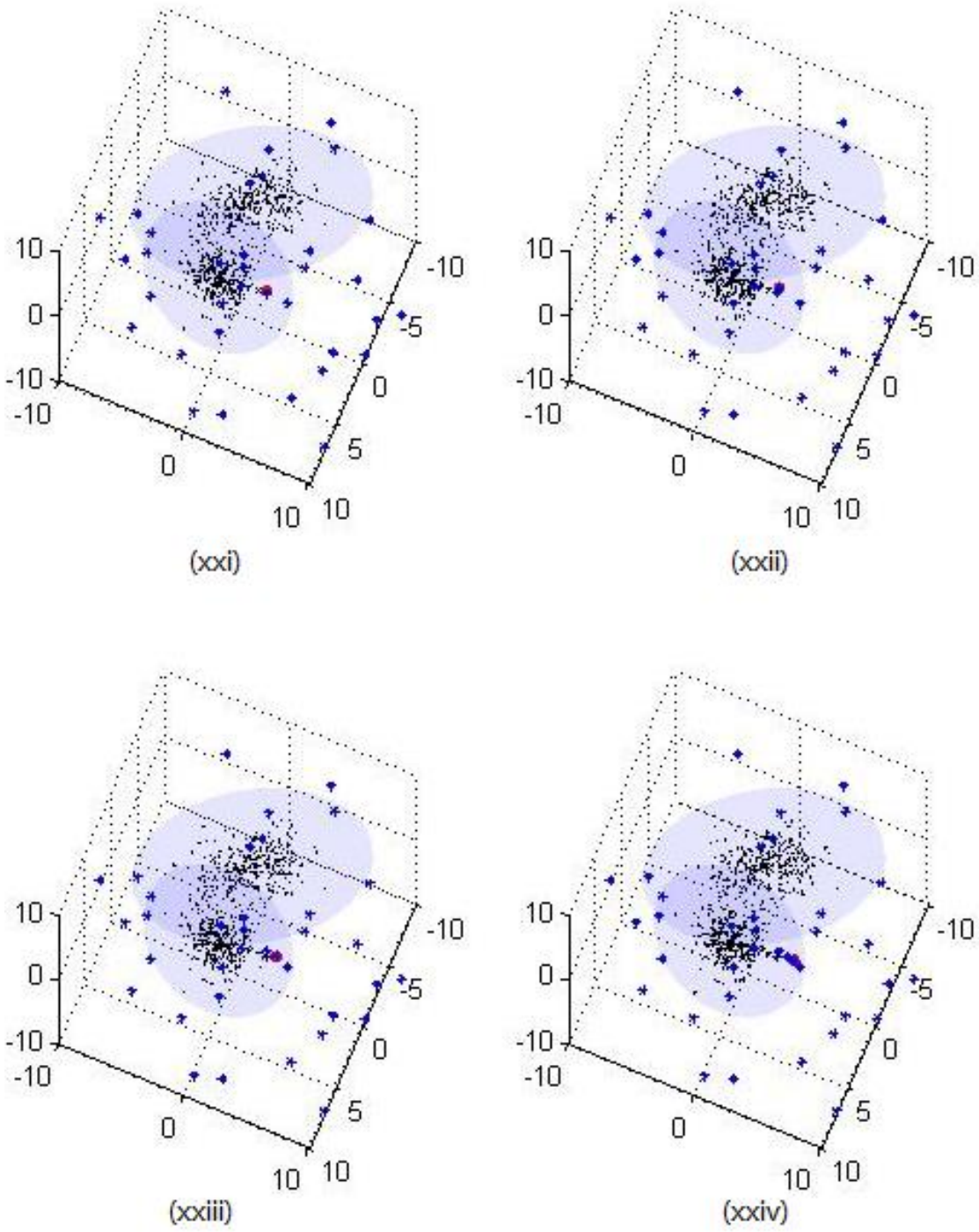


Figure 4.15 Movie clip of the updates of the 10D feature vectors, PCA reduction of the data is used for visualization. The red marker shows the most recent data entry, while the blue asterisks are anomalies flagged by the algorithm and the light-blue ellipsoids show the normal pattern. The red line (x)-(xviii) shows the trajectory as the new data entries keep pushing towards the boundary. From (xix)-(xxiv), the GMM can no longer tolerate the changes and alerts are fired.

A new type of alert is fired as the new data keep getting farther away in mahalanobis distance from the center of the Gaussian component, as can be seen from Figure 4.15(x) – (xviii) illustrated by the red trajectory. The alert is raised since the previous five data entries keep the tendency of deviating from the normal behavior despite the fact that they are still within the normal boundary. To put it simply, if the distance of the most recent data entry keeps getting bigger for five times in a row despite the fact that no anomaly alerts is fired, the prediction alert will be fired and corresponding information will be sent to the medical profession for reference. This prediction alert is designed in hope to get the attention of the health care provider as the elder starts to show sign of possible health decline. Since Figure 4.15(xviii), the GMM can no longer adapt to the changes and the anomaly alerts are fired.

To get more insights of how well the system works, multiple datasets of two dimensions and ten dimensions are tested. The results we get from the system are compared against the ground truth: the labels we get from generating the data. The data generated by the Gaussian distribution are labeled as normal and the feature vectors generated by the uniform-distributed random number generator are labeled as anomaly if and only if they are away from the dense regions of the data.

The accuracy along with the true positive rate for the anomaly alerts are computed for evaluation and the results are averaged over the datasets with the same dimension and. For each dataset with the same dimension, it starts off with the same parameter setting to randomly generate the Gaussian Mixtures of two components.

Here the same parameter setting means the same means and covariance matrices for the two dense regions of the data and noise data generated from uniform distribution. The new data points are added in a way that they may stay in a certain clusters for a while, travel between clusters, deviate from the clusters, start spawning a new cluster or just be a noise data point away from the clusters. For the two-dimensional datasets, we use 200 feature vectors to initiate the system and update the system using 400 more feature vectors. We choose $\mathbf{T_d} = 3$, $\mathbf{T_n} = 0.06$, $\mathbf{T_o} = 0.4$. For the ten-dimensional datasets, we use 400 feature vectors to initiate the system and update the system using 500 more feature vectors. We choose $\mathbf{T_d} = 6$, $\mathbf{T_n} = 0.06$, $\mathbf{T_o} = 0.4$. .

For each dimension, we create 20 datasets which are different from each other and run the m through the system. The results are compared to the manual labeling of the data. The accuracy and the True Positive rate for the results are shown in Table 4.4:

Table 4.5 Averaged Results of Accuracy and True Positive Rate of the alert algorithm run on 2-dimensional datasets and 10-dimensional datasets over 20 independent trials

	2-dimension	10-dimension
Accuracy	94.84%	99.80%
True Positive Rate for the anomaly alerts	82.49%	99.94%

We can see from the table that the system performs significantly better on 10-dimensional data than on the 2-dimensional data despite both cases give satisfying results. This could be explained by the extra information involved in the higher-dimensional data. Since in higher dimension, there is more degree of freedom and the data will seem more separate, it is much easier for the system to detect the anomaly than in the lower

dimension. Yet to achieve this, not only does it require more computational power, more data need to be gathered to ensure a better initialization of the Gaussian Mixture Model.

5 Discussion

The alert algorithm used at TigerPlace right now is a straightforward one dimensional strategy, with about half of the alerts generated are false alarms. The framework proposed in this thesis is designed in hope to improve recognition of early illness sign by the use of multiple features, thus bring more insight to the early disease recognition.

Chapter 4 shows us that by utilizing the PCM's robustness in the noisy environment and AMPCM's ability to cluster without the need to specify the number of clusters and any parameter, the system is able to isolate the noise from the rest of the data and find the natural structure of the data in the initializing window, which makes it a very suitable method to initiate the Gaussian Mixture Model. Also, by introducing the typicality information from the possibilistic partition, both the crisp method and the fuzzy-2 method are able to give accurate estimation of the Gaussian parameters while the fuzzy-2 method has more robust performance under noisy situation. With the help of Gaussian Mixture Model and appropriate distance threshold N, the system can tolerate gradual changes in the pattern; find anomalies and adapting the model to the new states. By using the PCM and the cluster validity measure, a new emerging normal behavior pattern can be discovered and incorporated into the GMM to better represent the normal behavior. Not only does the system fire an alert when the behavior pattern falls outside the GMM, another type of alert will be fired if the new pattern keeps deviating from the normal behavior, indicating possible health decline to raise the attention of the health care provider.

Table 4.4 shows us the significant advantage of using high-dimensional data. The ten-dimensional data helped us to give almost every "anomaly" and "normal pattern" their right labels, while the two-dimensional data gave less satisfying result comparatively. The reason for the significant improvement in the True Positive Rate for the anomaly alert after using the higher-dimensional data is that data will appear to be sparser in the higher dimension and the noise in the datasets will appear to be farther away from the GMM, thus are recognized much easier.

Yet we inevitably come across the curse of dimensionality when dealing with the high-dimensional data. The demand of large number of samples when initializing the Gaussian Mixture Model grows exponentially with the dimensionality of the feature space. When the AMPCM and the PCM were used to initialize the Gaussian Mixture Model in ten-dimensional space, the algorithm failed to recognize the structure of data when the size of initialization window was set as 200 and still would be at risk when the window size was set as 300. And this is just synthetic data. On the other hand, given ample amount of data, the system did work well on the synthetic data.

In Chapter 4, we utilize the typicality information from the possibilistic partition to estimate the Gaussian parameters of the GMM components, and update the GMM component by combining all the feature vectors belong to that component and computing the mean and covariance matrix. Yet instead of treating each feature vector equally when computing the mean and covariance matrix, we could actually use the technique in the Stauffer's background model [20] and bias more towards the most recent feature vector, according to equations (2.6) – (2.8). In this case, not only can we take the typicality information from the possibilistic partition into consideration by substituting the normal

probability $N(X_t|\mu_k, \sigma_k)$ with the typicality in equation (2.8); the system will put more emphasis on the most current behavior pattern when updating the GMM so that the system can have more temporal consideration of the data. The parameter α will be an interesting parameter to adjust the system‘s sensitivity towards the most recent behavior pattern, helping the health care provider to capture changes in health condition.

Though everything worked well on the synthetic data, with no doubt there would be more complications when the system is tried on the real data: choosing the meaningful features, extracting information from various kinds of sensors, signal processing the data, determine how trustworthy the feature value is. Judging by the system's behavior on the synthetic data, it is promising that feeding the system with in-home sensors' data from Tiger Place will give meaningful alerts indicating an elder's health change.

6 Conclusion

Inspired by using Gaussian mixture model to model the background proposed by Stauffer et al. [4], a system analyzing multi-dimensional activity feature vector with temporal consideration is developed in hope of identifying signs of early diseases. It involves modeling the normal behavior pattern, real-time updating the model and fire an alert when detecting unexpected behavior pattern.

Mainly using the PCM, the AMPCM and the Gaussian Mixture Model, the system has the capability to find the structure of the data without a prior, adapt to gradual changes, find anomalies, and spawn a new component for the GMM when there is an emerging new normal pattern. The system achieves our goals when tested on the synthetic datasets over extended period of time. We hope that by using the system in Tiger Place, it will help by detecting health changes before real health issue happens.

Appendix

Experiment 4.3 and Experiment 4.5 (1):

Original Gaussian Mixture:

The data are generated using the matlab function mvnrnd() with the Gaussian parameters:

```
mu1=[6;6];sigma1=[2 0;0 .5];

mu2=[-2;0];sigma2=[2 1;1 2];
```

Noise: two dimensional data with values randomly drawn from uniform distribution with range [-10,10] using matlab function rand();

The data above are mixed together using matlab function randperm().

Data starting to deviate from the center of a Gaussian component in a spiral trajectory are generated in following code:

```
r=pdist2(data(i,:),mu')+cumsum(rand(1,numnewline))/3;

temp=data(i,:)-mu';

theta=(atan(temp(2)/temp(1))+(1-sign(temp(1)))*pi/2);

theta=theta+cumsum(rand(1,numnewline))/pi;

x=r.*cos(theta)+mu(1);y=r.*sin(theta)+mu(2);

new=[x;y];
```

Experiment 4.5 (2):

Original Gaussian Mixture:

```
mu1=[6;6];sigma1=[1.5 0;0 1.5];

mu2=[5;1];sigma2=[1 0.5;0.5 1];

 mu3=[1;5];sigma3=[1 0.5;0.5 1];
```

Noise: two dimensional data with values randomly drawn from uniform distribution with range [-10,10];

The data above are mixed together using matlab function randperm().

New emergent cluster:

mu=[4,-4];sigma=[.5,0;0,.5];

Experiment 4.5 (3):

Original Gaussian Mixture:

mu1=[-4,-.5,0,0,0,0,0,0.5,0.5,0];

sigma1=diag([1,.5,1,1,2,1,.1,1,1,1]);

mu2=[-2,0.5,0,0,0,1,0,0,3,-5];

sigma2=diag([5,.2,1,1,2,1,1,1,.1,1]);

Noise: two dimensional data with values randomly drawn from uniform distribution with range [-10,10];

Data starting to deviate from the center of a Gaussian component are generated in following code:

```
        tt=1*cumsum(rand(1,numnewline))/6;
        x=mu1(1)+tt;
        new=x;
        for d=2:length(mu)
            new=[new;((x-mu1(1))*(train(i,d)-mu1(d))/(train(i,1)-
mu1(1))+randn/6)+mu1(d)];
        end

        new=new+randn(size(new))/5;
```

Reference

[1] Tuljapurkar, S. 2005. Future Mortality: A Bumpy Road to Shangri-La? Sci. Aging Knowl. Environ., Vol. 2005, Issue 14, p. 9, 6 April 2005.

[2] Skubic, M, Rantz, M, Miller, S, Guevara, RD, Koopman, R, Alexander, G, Phillips, L. (to appear). Non-Wearable In-Home Sensing for Early Detection of Health Changes. Quality of Life Technology for the Disabled and Elderly, R, Schultz, ed., CRC Press.

[3] Rantz, M. J., Marek, K. D., Aud, M. A., Johnson, R. A., Otto, D., & Porter, R. (2005). TigerPlace: A new future for older adults. Journal of Nursing Care Quality, 20(1), 1-4. PMID: 15686069.

[4] Wang, S. (2011). Change Detection for Eldercare Using Passive Sensing, PhD Dissertation, Electrical & Computer Engineering, University of Missouri, Columbia, MO, Dec

[5] Heise D, Rosales L, Sheahen M, Su BY & Skubic M, "Non-Invasive Measurement of Heartbeat with a Hydraulic Bed Sensor: Progress, Challenges, and Opportunities," Proceedings, 2013 IEEE International Instrumentation & Measurement Technical Conference, Minneapolis MN, May 6-9, 2013, pp 4356-4360..

[6] Stone E & Skubic M, "Fall Detection in Homes of Older Adults Using the Microsoft Kinect," IEEE Journal of Biomedical and Health Informatics, in press.

[7] Banerjee T, Keller J, Skubic M & Stone E, "Day or Night Activity Recognition from Video Using Fuzzy Clustering Techniques," IEEE Transactions on Fuzzy Systems, 2013, 22(2):483-493.

[8] Skubic M, Alexander G, Popescu M, Rantz M & Keller J (2009). A smart home application to eldercare: Current status and lessons learned. Technology and Health Care, 17(3): 183-201.

[9] Rantz MJ, Skubic M, Miller SJ, Galambos C, Alexander G, Keller J & Popescu M, "Sensor Technology to Support Aging in Place," Journal of the American Medical Directors Association, 2013, 14(6):386-391.

[10] Rantz MJ, Skubic M, Miller SJ & Krampe J (2008). Using technology to enhance aging in place. Proc. of the 6th Intl. Conf. on Smart Homes and Health Telematics, Ames, IA, July, pp. 169-176.

[11] Rantz MJ, Skubic M, Alexander GL, Aud MA, Wakefield BJ, Koopman RJ & Miller SJ (2010). Improving nurse care coordination. CIN: Computers Informatics Nursing, 28(6): 325-332.

[12] Galambos C, Skubic M, Wang S & Rantz M (under review). The use of density mapping for early illness detection of dementia and depression in older adults.

[13] Wang S, Skubic M & Zhu Y (2009). Activity density map dis-similarity comparison for eldercare monitoring. Proc, Conference of the IEEE Engineering in Medicine and Biology Society Conference, Minneapolis, MN, Sep 2-6, 2009, pp. 7232-7235.

[14] Alexander GL, Rantz M, Skubic M, Koopman RJ, Phillips LJ, Guevara RD & Miller SJ (2011). Evolution of an Early Illness Warning System to Monitor Frail Elders in Independent Living. Journal of Healthcare Engineering, 2(2): 259-286, 2011.

[15] Rantz MJ, Skubic M, Koopman R, Phillips L, Alexander GL , Miller SJ, & Guevara, RD. (2011). Using sensor networks to detect urinary tract infections in older adults. Proceedings, 13th IEEE International Conference on e-Health Networking, Application, & Services, Columbia, MO, June 13-15, 2011. Pp. 142-149. NIHMSID:345670.

[16] Sledge I, Keller J & Alexander G (2008). Emergent trend detection in diurnal activity, Proc. of the 30th Annual International IEEE EMBS Conference, Vancouver, BC, Canada, August 20-24, 2008.

[17] B. Fritzke, "A Growing Neural Gas Network Learns Topologies", Advances in Neural Inform. Processing Sys., vol. 7, pp. 625-632, 1994.

[18] G. Carpenter and S. Grossberg, "Adaptive Resonance Theory", The Handbook of Brain Theory and Neural Networks, MIT Press, 2003.

[19] Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[20] Stauffer, Chris; Grimson, W. E L, "Adaptive background mixture models for real-time tracking," Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. , vol.2, no., pp.,252 Vol. 2, 1999

[21] J.Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum Press, 1981.

[22] R.Krishnapuram and J.M.Keller, "A possibilistic approach to clustering," IEEE Trans. Fuzzy Syst., vol. 1, no. 2, pp. 98–110, May 1993.

[23] Krishnapuram, R.; Keller, J.M., "The possibilistic C-means algorithm: insights and recommendations," Fuzzy Systems, IEEE Transactions on , vol.4, no.3, pp.385,393, Aug 1996

[24] Miin-Shen Yang; Chien-Yo Lai, "A Robust Automatic Merging Possibilistic Clustering Method," Fuzzy Systems, IEEE Transactions on , vol.19, no.1, pp.26,41, Feb. 2011

[25] Richard O. Duda, Peter E. Hart, and David G. Stork. 2000. Pattern Classification (2nd Edition). Wiley-Interscience.