Robust Parameter Estimation and Outlier Detection in Online Streaming data using Two-Fold Incremental Gaussian Mixture Model

Bachelor Thesis Project-II (MA47202) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Integrated Masters of Science

in

Mathematics and Computing

by

Prabhav Sunil Patil

(20MA20042)

Under the supervision of

Professor Swanand Khare

Department of Mathematics

Indian Institute of Technology Kharagpur

Spring Semester, 2023-24

April 30, 2024

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 30, 2024 (Prabhav Sunil Patil)

Place: Kharagpur (20MA20042)

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Robust Parameter Estimation and Outlier Detection in Online Streaming data using Two-Fold Incremental Gaussian Mixture Model**" submitted by **Prabhav Sunil Patil** (Roll No. 20MA20042) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Integrated Masters of Science in Mathematics and Computing is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2023-24.

Date: April 30, 2024
Place: Kharagpur

Professor Swanand Khare
Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Prabhav Sunil Patil** Roll No: **20MA20042**

Degree for which submitted: **Integrated Masters of Science**

Department: **Department of Mathematics**

Thesis title: **Robust Parameter Estimation and Outlier Detection in Online Streaming data using Two-Fold Incremental Gaussian Mixture Model**

Thesis supervisor: **Professor Swanand Khare**

Month and year of thesis submission: **April 30, 2024**

With the increasing complexity of industrial production, data-driven based monitoring methods attract more attention. However, the conventional static process monitoring methods may show poor performance for the time-varying processes since they fail to track the time-varying characteristics. Incremental algorithms such as IGMM and MIGMM succumb to their inability to identify the outliers in the data-stream and the removal of the data points which can potentially form a cluster in the later-stage of online streaming.

This study proposes a two-level partition framework that can separate normal cluster-driven samples from fault samples, reducing the possibility of adding fault samples to the model. Comparative experiments are studied firstly on a synthetic periodic dataset and then finally, on a real-world financial dataset of past 10 years comprising of 5 stocks, which show that the proposed framework performs robustly compared to the classical problems.

Acknowledgements

Expressing my sincere gratitude towards my project supervisor, Professor Swanand Khare for his prompt assistance everytime I needed his support.
I deem it a blessed circumstance that I have such an immensely knowledgable and plentiful experienced person as my supervisor who has guided me to focus on important topics and has ensured that I perform my work in a goal-oriented fashion. I am also thankful towards him in helping me understand the problem statement clearly, and patiently hearing as well as sharing all the approaches which has had significant contributions in helping me develop my thesis. Providing and guiding me towards the field he has a vast experience in, he has provided me a deeper insight into thesis topics which I would not have the opportunity to study otherwise, piquing my interest into the project even further.
Furthermore, I am deeply indebted towards the Department of Mathematics, IIT Kharagpur that has provided me this opportunity to learn and educate myself on developing a proper thesis and a project.

Contents

Chapter 1

Introduction

The Gaussian mixture model, briefly termed as 'GMM' (Reynolds et al. (2009)), a statistical modeling tool, has been successfully and widely used in pattern recognition and machine learning tasks due to its powerful statistical characteristics. Typically, the parameters of GMM are learned in a weight space using the expectation maximization (EM) algorithm (Moon (1996)). However, this approach assumes that all demonstrations are available at the learning time, and the number of components M remains fixed after learning. This implies that the Gaussian Mixture Model with a fixed number of mixture components cannot cope with applications of online learning.

The incremental Gaussian mixture model (IGMM) (Engel and Heinen (2010)), an online incremental learning algorithm, is very similar to the EM algorithm for Gaussian mixture models. IGMM adopts a GMM of distribution components that can be expanded to accommodate new information from an input data point. Based on the maximization of the likelihood of the data, each data point assimilated by the model contributes to the sequential update of the model parameters. The estimated parameters are updated through the accumulation of relevant information from each data point. New points are added directly to existing Gaussian components, or new components are created when necessary, avoiding merge and split operations. However, this procedure does not take into account the smaller prior probabilities of the components generating spurious clusters, which overfit the data set. Hence, this procedure based on the accumulator variable is validated inefficient, failing to

consider overlapping rates between components which violate the update process rule of incremental Gaussian mixture models.

Therefore, to reduce the computational complexity, the precision matrix and its determinant update procedure were developed(Pinto and Engel (2015)) to construct the modified incremental Gaussian mixture model (MIGMM)(Sun et al. (2023)); to fully consider the importance of the prior probability p_k and the overlapping rates against the update process of the MIGMM. Though the model estimates the components robustly, the data-stream has to be reset after streaming a significant data set, due to this model being a two-phase model. Hence, this model leads to the removal of the data points which can potentially form a cluster in the later-stage of online streaming, since there is a possibility of the data points in the similar region to stream into the model at a later stage.

Our Contribution: For a robust estimation of the data-stream for an irregular and continuous data-stream, a two-level iterative framework is established which adaptively distinguishes an incoming normal data-points from outliers, using recursive equations developed in IGMM(Engel and Heinen (2010)) to update the parameters of the model, which reflects twofold:

- an efficient representation of data based on normal shifting data-points taken from the online data-stream so far, resulting in more accurate model parameters
- an adaptive list of outliers taken through the online data-stream, which can then be investigated for many practical applications such as detection of anomalies, data-quality assurance, inferential statistics etc.

The remainder of this report is organized as follows. Chapter 2 presents the preliminaries of the concepts and mathematical models used throughout the report, along with the literature survey of classical incremental algorithms, namely IGMM and MIGMM. Chapter 3 presents the development of adaptive two-fold framework, including mathematically rigorous statements to justify its validity. Chapter 4 presents an experiment-based evaluation of the proposed methodology compared with other state-of-the-art unsupervised algorithms. Chapter 5 presents the experimental evaluation of the model on a periodical synthetic dataset, following the application on a financial dataset used in determining the market trends. Finally, Chapter 6 concludes the report, stating the limitations and potential future work.

Chapter 2

Theory Preliminaries and Related Work

2.1 Preliminaries

In laying the groundwork for our discussion, we begin by introducing foundational concepts that we will assume for the rest of the report.

2.1.1 Probabilistic Clustering

Gaussian mixture model

For a D-dimensional random variable x, a Gaussian mixture model (GMM), can be written as a linear superposition of Gaussian components (Reynolds et al. (2009)) in the form:

$$\sum_{g=1}^{G} \pi_g \mathcal{N}(x|\mu_g, \Sigma_g) \tag{2.1}$$

where G is the number of Gaussian components, and each component has weights (or prior probabilities) $\pi_{\mathbf{g}}$, mean $\mu_{\mathbf{g}}$ and covariance matrix $\mathbf{\Sigma_g}$, g = 1, 2, ..., G. The non-negative weights satisfy the constraints $\sum_{g=1}^{G} \pi_g = 1$. Use $\theta_{\mathbf{g}}$ to

represent the g-th Gaussian components' parameters as $\theta_g = \{\mu_g, \Sigma_g\}$ and use $\Theta = \{\theta_1, ..., \theta_g, \pi_1, ..., \pi_g\}$to represent the GMM model.

To establish GMM model based on samples, Expectation Maximization (EM) algorithm is commonly employed to estimate GMM parameters Θ (Moon (1996)) .

This approach assumes that all the samples are readily available at the estimation, and number of components G remain fixed. Furthermore, a GMM with fixed number of mixture components cannot cope with applications of online learning, since this approach fails to adapt new sets of cluster which could quite be the case during an online data-stream.

Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm is a powerful iterative method used to estimate parameters in statistical models when the data has missing or hidden variables. It is particularly popular in the context of clustering algorithms such as Gaussian Mixture Models (GMMs).

The EM algorithm consists of two main steps: the E-step (Expectation) and the M-step (Maximization). It iterates between these two steps until convergence is achieved.

E-step (Expectation): In the E-step, we calculate the expected value of the hidden variables given the observed data and the current estimate of the parameters. For GMMs, this involves calculating the probability that each data point belongs to each of the Gaussian distributions.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{2.2}$$

Where:

- $\gamma(z_{nk})$ is the responsibility of the k-th component for the n-th data point.
- π_k is the mixing coefficient (or prior probability) of the k-th component.
- $\mathcal{N}(x_n|\mu_k, \Sigma_k)$ is the probability density function (PDF) of the multivariate Gaussian distribution with mean μ_k and covariance Σ_k evaluated at x_n.

M-step (Maximization): In the M-step, we update the parameters of the model to maximize the likelihood of the observed data, using the expected values of the hidden variables computed in the E-step. For GMMs, this involves updating the means, variances, and mixture coefficients of the Gaussian distributions.

Update for means μ_k:

$$\mu_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2.3}$$

Update for covariances Σ_k:

$$\Sigma_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2.4}$$

Update for mixing coefficients π_k:

$$\pi_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N} \tag{2.5}$$

Where:

- μ_k^{new} is the updated mean of the k-th component.
- Σ_k^{new} is the updated covariance matrix of the k-th component.
- π_k^{new} is the updated mixing coefficient of the k-th component.
- N is the total number of data points.
- $\gamma(z_{nk})$ is the responsibility of the k-th component for the n-th data point, computed in the E-step.

2.1.2 Density Based Spatial Clustering of Applications with Noise: DBSCAN Algorithm

Density-based clustering can be considered as a non parametric method, as it makes no assumptions about the number of clusters or their distribution. DBSCAN estimates the density by counting the number of points in a fixed-radius neighborhood and considers two points as connected if they lie within each other's neighborhood.

A point is called core point if the neighborhood of radius *Eps* contains at least $MinPts$ points, i.e., the density in the neighborhood has to exceed some threshold.

Algorithm 1: Expectation-Maximization Algorithm for GMMs

Initialize the parameters: means μ_k, covariances Σ_k, and mixture coefficients π_k for $k = 1, 2, ..., K$.

Repeat

E-step: Compute the responsibilities $\gamma(z_{nk})$ using the current parameter estimates:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

where $\mathcal{N}(x_n|\mu_k, \Sigma_k)$ is the probability density function of the k-th Gaussian.

M-step: Update the parameters:

$$\mu_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_n}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

$$\Sigma_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

$$\pi_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$$

Until Convergence

A point q is directly density-reachable from a core point p if q is within the *Eps*-neighborhood of p, and density-reachability is given by the transitive closure of direct density-reachability. Two points p and q are called density-connected if there is a third point o from which both p and q are density-reachable. A cluster is then a set of density-connected points which is maximal with respect to density-reachability. Noise is defined as the set of points in the database not belonging to any of its clusters. The task of density-based clustering is to find all clusters with respect to parameters *Eps* and $MinPts$ in a given database.

Formal Definitions

Let D be a set of datapoints. The definition of density-based clusters assumes a distance function $dist(p, q)$ for pair of points. The *Eps*-neighbourhood of a point p, denoted by $NEps(p)$, is defined by

$$NEps(p) = \{q \in D | dist(p, q) \leq Eps\}$$

- A point p is *directly density-reachable* from a point q with respect to Eps, $MinPts$ if $p \in NEps(q)$ and $|NEps(q)| \geq MinPts$.
- A point p is *density-reachable* from a point q with respect to Eps and $MinPts$ if there is a chain of points $p_1, ..., p_n$, $p_1 = q$,$p_n = p$ such that p_{i+1} is directly density-reachable from p_i. Density-reachability is a canonical extension of direct density reachability.
- A point p is *density connected* to a point q with respect to Eps and $MinPts$ if there is a point o such that both p and q are density-reachable from o with respect to Eps and $MinPts$

A cluster C with respect to Eps and $MinPts$ is a nonempty subset of D satisfying the following two conditions:

- **Maximality Condition:** $\forall\ p, q$ if $p \in C$ and q is density-reachable from p with respect to Eps and $MinPts$,then $q \in C$.
- **Connectivity Condition:** $\forall p, q \in C$: p is density-connected to q with respect to Eps and $MinPts$.

Let $C_1, ..., C_k$ be the clusters of the database D with respect to Eps and $MinPts$. Then the noise is defined as the set of points in D not belonging to any cluster C_i, i.e.,

$$noise = \{p \in D | p \in C_i \forall i\}$$

These definitions help us distinguish three types of points:

- core points, i.e., points with a dense neighborhood ($|NEps(p)| \geq MinPts$),
- border points, i.e., points that belong to a cluster, but whose neighborhood is not dense, and
- noise points, i.e., points which do not belong to any cluster.

Algorithm

DBSCAN (Khan et al. (2014)) starts with an arbitrary database point p and retrieves all points density-reachable from p with respect to Eps and $MinPts$, performing region queries first for p and if necessary for p's direct and indirect neighbors. If p is not a core point, no points are density-reachable from p and DBSCAN assigns p to the noise.

Algorithm 2: DBSCAN Algorithm

Input: Dataset D, parameters ε and $minPts$

Output: Clusters $C_1, C_2, ..., C_k$

Initialize an empty set of clusters $C = \emptyset$

For each unvisited point p in D

  Mark p as visited

  Find all points within the ε-neighborhood of p (including p itself)

  If the number of points in the neighborhood is less than $minPts$

    Mark p as noise

  Else

    Create a new cluster C_i

    Expand the cluster by recursively adding all density-reachable points to C_i

2.2 The Incremental Gaussian mixture model

This section describes IGMM (standing for Incremental Gaussian Mixture Model), which was designed to learn Gaussian mixture models from data flows in an incremental and unsupervised way(Engel and Heinen (2010)).

IGMM adopts an incremental mixture distribution model, having special means to control the number of mixture components that effectively represent the so far presented data. The main focus of interest relies in modeling environments whose overall dynamics can be described by a set of persistent concepts which will be incrementally learned and represented by a set of mixture components. Therefore, this model uses a novelty criterion to overcome the problem of the model complexity selection, related to the decision whether a new component should be added to the current model. The mixture model starts with a single component with unity prior, centered at the first input data, with a baseline covariance matrix specified by default, i. e., $\mu_1 = x^1$, meaning the value of x for t = 1, and $(\Sigma_1)^1 = \sigma_{ini}^2 I$, where σ_{ini} is user-specified configuration parameter.(Engel and Heinen (2010))
New components are added by demand. IGMM uses a *minimum likelihood* criterion to recognize a vector x as belonging to a mixture component. For each incoming data point the algorithm verifies whether it minimally fits any mixture component. A data point x is not recognized as belonging to a mixture component j if its

probability $p(x|j)$ is lower than a previously specified *minimum likelihood-* (or *novelty−*) *threshold*(Engel and Heinen (2010)). In this case, $p(x|j)$ is interpreted as a likelihood function of the j-th mixture component. If x is rejected by all density components, meaning that it bears new information, a new component is added to the model, appropriately adjusting its parameters.

The novelty-threshold value affects the performance of the learning process to new concepts, with higher threshold values generating more concepts. Since the cutoff with which an incoming datapoint should be added to a specific cluster can vary with the structure of the cluster, this model takes a specified minimum value for the acceptable likelihood, τ_{nov}, as a fraction of maximum value of the likelihood function, making the novelty criterion independent of the covariance matrix.

Hence, a new mixture component is created when the instantaneous data point $x = (x_1, ..., x_i, ..., x_D)$ matches the novelty criterion written as

$$p(x|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|Cj|}} \forall j \tag{2.6}$$

as discussed in (Engel and Heinen (2010)).

An instantaneous data point that does not match the novelty criterion needs to be assimilated by the current mixture distribution, causing an update in the values of its parameters due to the information it bears. IGMM follows an incremental version for the usual iterative process to estimate the parameters of a mixture model based on two steps: an estimation step (E) and a maximization step (M).

When the probability density of the input data is a Gaussian mixture model with M components, an observation x^t is probabilistically assigned to a distribution j by the corresponding posterior probability $p(j|x_t)$. In this case, the equivalent number of samples used to compute the parameter estimates of the j-th distribution component corresponds to the sum of posterior probabilities that the data presented so far were generated from component j, the so called $0th$-order moment of $p(j|x)$ over the data, or simply the $0th$-order data moment for j. IGMM stores this summation as the variable sp_j which is periodically restarted to avoid an eventual saturation.

Hence, the recursive equations used by IGMM (Engel and Heinen (2010)),to update the parameters of the model are given as,

$$sp_j = sp_j + p(j|x) \tag{2.7}$$

$$\mu_j = \mu_j + \frac{p(j|x)}{sp_j}(x - \mu_j) \tag{2.8}$$

$$\Sigma_j = \Sigma_j - (\mu_j - \mu_j^{old})(\mu_j - \mu_j^{old})^T + \frac{p(j|x)}{sp_j}[(x - \mu_j)(x - \mu_j)^T - \Sigma_j] \tag{2.9}$$

$$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q} \tag{2.10}$$

where $p(j|x)\mu_{\mathbf{j}}^{\mathbf{old}}$ refers to the value of $\mu_{\mathbf{j}}$ at time $t-1$ (i.e., before updating). The IGMM algorithm has just two configuration parameters, σ_{ini} and τ_{nov}. The σ_{ini} parameter is not critical – the only requirement for σ_{ini} is to be large enough to avoid singularities. The τ_{nov} parameter, on the other hand, is more critical and must be defined carefully. It indicates how distant x must be from μ_j to be considered as a non-member of j .The algorithm for IGMM is summarized in Algorithm 3.

Algorithm 3: IGMM Algorithm procedure

Input: X
Parameters: $\mu, \Sigma, |\Sigma|, p, sp$
While $x \in X$ **Do**
1. Determine whether x matches the novelty criterion described in Eq.(2.2)
2. If x matches the criterion **Do**
3. Set $\mu_{|\mu|+1} = x$ and $\Sigma_{|\mu|+1} = \sigma_{ini}$
4. Append $(\mu_{|\mu|+1}, \Sigma_{|\mu|+1})$ to (μ, Σ)
5. **End**
6. Update (μ, Σ, p, sp) using Eq.(2.3) - Eq.(2.6)
End
Output: $\mu, \Sigma, |\Sigma|, p, sp$

2.3 Modified Incremental Gaussian mixture model

The MIGMM(standing for Modified Incremental Gaussian mixture model) framework contributes to the adaptive estimation of data-stream in 2 ways:
i. a more simple and efficient prediction matrix update, the core of the update procedure
ii. an effective exponential model (p_{Thv}) related to the number of output components, combined with the Mahalanobis distance-based logical matrix (LM), which was proposed to remove spurious components.(Sun et al. (2023)).

In this section, we focus only on removing the spurious components, i.e. the clusters having prior probabilities less than a certain threshold p_{Thv} Regarding the methods to remove spurious components, this approach can be divided into the following two branches:

2.3.1 p_{Thv}-based spurious component deletion

The generated spurious components by referring to the input components should be deleted when each of them satisfies two conditions: (1) it is far away from any input components; (2) it's pm is quite small.

Conductive experimental analysis, the behavior of the average of P_M with M could be fitted by the exponential equation

$$\hat{P_M} = ae^{bM} \tag{2.11}$$

where the coefficient of b should be a negative, giving rise to the decreasing behavior as shown in Fig. 3.1. Based on the curve-fitting toolbox ($cftool$) and solving for the value of a and b, the new curve model

$$p_{Thv} = \frac{\pi}{10} e^{-\frac{\pi}{10}M} \tag{2.12}$$

is determined by using $\frac{\pi}{10}$ to replace a and $-b$ for convenience in computation.(Sun et al. (2023)) Therefore, the curve model Eq. (2.8) is used as threshold value to

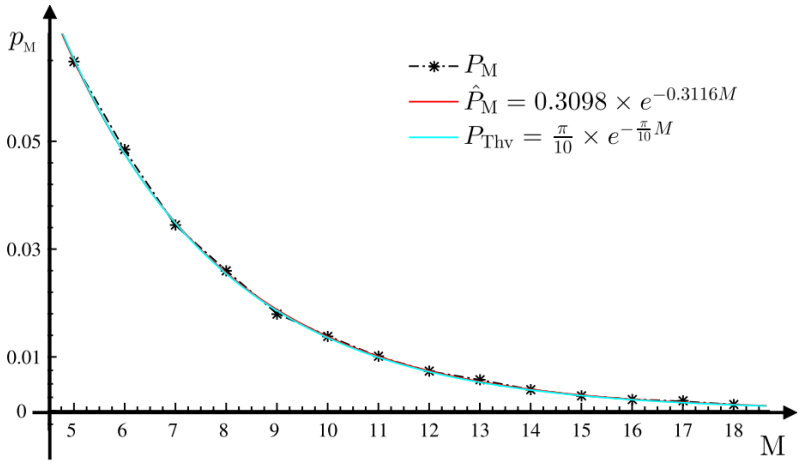


FIGURE 2.1: The errorbar characters of the relationship between P_M and M

remove any component m when $p_m \leq p_{Thv}$.

2.3.2 LM-based removal of spurious components

Logical Matrix (LM) definition The logical matrix, denoted as LM , is defined by $LM = [a_{i,j}]_{MM}$ such that

$$a_{i,j} = \begin{cases} 1, & \text{if } (\mu_j - \mu_i)' \Sigma_i^{-1} (\mu_j - \mu i) \leq \chi^2_{D,1-\beta} \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

where $\chi^2_{D,0.95}$, is chosen according to the 95% confidence level, which is a widely used classification criterion in many studies.(Sun et al. (2023))

Removal Procedure On the basis of the experimental studies, the following conclusions were made: i. a component i such that $a_{i,j} = 1$ and $length(j) \geq 2$ can be considered spurious and removed, or ii. a component i such that $a_{i,j} = 1$ and $length(j) = 1$ and $p_i \leq p_j$ can be considered spurious and removed.
The LM-based spurious component deletion algorithm is summarized in Algorithm 4.

Algorithm 4: LM-Based spurious components deletion algorithm

Input: $\mu, \Sigma, |\Sigma|, p, sp$
Determine the components number M = length of p
Resort component i (i = 1,2,...,M-1) subject to $p_i \geq p_{i+1}$
Generate logical matrix $LM = [a_{i,j}]_{MxM}$ by Eq.(2.9)
Calculate the sum of each column in LM, $SLM = \text{sum}(LM^{'})$
While SLM(k) ≥ 2 **Do**
$LM(k,:) = [\,], LM(:,k) = [\,]; \mu_k = [\,]; p_k = [\,]; sp_k = [\,]; \Sigma_k = [\,]; |\Sigma_k| = [\,]$
subject to $p_k \leq p_m(SLM(m) \geq 2)$
SLM = sum(LM)
End
While SLM(k)=1 **Do**
$LM(k,:) = [\,], LM(:,k) = [\,]; \mu_k = [\,]; p_k = [\,]; sp_k = [\,]; C_k = [\,]; dc_k = [\,]$
subject to $p_k \leq p_m(SLM(m) = 1)$
SLM = sum(LM)
End
Recompute the prior probability p_m, $\forall m \in [1, M]$ where $M = \text{length}(sp)$
Output: $\mu, \Sigma, |\Sigma|, p, sp$

Therefore, based on the threshold value line calculated by Eq. (2.8), which is related to the number of output components M, and based on LM, the removal procedure algorithm is summarized in Algorithm 5.

Algorithm 5: The spurious component removal procedure

1. Input $\mu, \Sigma, |\Sigma|, p, sp$
2. Determine the components number M = length of p
3. Find component k subject to $p_k \leq \frac{\pi}{10} e^{-\frac{\pi}{10}M}$
4. Remove component k, that is p_k=[ ], C_k=[ ], sp_k=[ ], dc_k=[ ], μ_k=[ ], $M = M - length(k)$
5. Remove components based on LM-based spurious components deletion algorithm described in **Algorithm 4**

Chapter 3

A Two-Fold IGMM Model Formulation

Two-fold incremental Gaussian mixture model processes the adaptive algorithm framework as well as the novel methodology to identify the outliers in a parallel manner, which reflects in twofold;

- creation of an adaptive list of the outlier data subject to change as the data-stream continues to get inculcated in the model.
- continuous adaptive data-driven model which efficiently updates the parameters even for a long irregular continuous stream of data points.

3.1 General Framework

3.1.1 Formal Definitions

We now intuitively state the different categories of samples which we assume for our model framework. Assume the model has $C_1, C_2, ..., C_K$ clusters at some time t. Then:

- **Cluster sample point**: A datapoint x at each update is termed cluster sample point with respect to τ_{nov} if $p(x|C_j) \geq \tau_{nov}$ for some C_j where $j \in \{1, ..., K\}$.

- **Outlier point**: A datapoint x at each update is termed outlier point with respect to τ_{nov} if $p(x|C_j) < \tau_{nov}$ $\forall C_j \in \{C_1, .., C_K\}$

3.1.2 Methodology for the Detection of Outliers

A data point is considered an outlier when it is significantly different from the majority of the data in a data set. Hence, outliers are the part of the new samples which are not well described by any current components in the GMM framework. Hence, a new data point x does not belong to a component j if the probability $p(x|j)$ is lower than a specified threshold. If x is rejected by all the components, x is hence, considered to be an outlier.
This criterion is exactly similar to the novelty-criterion τ_{nov} in the IGMM model. However, for our framework, if a new incoming data point x is accepted, instead of creating a new component, the model classifies x as an outlier point and x is inserted into the set of outlier points. This process can, hence, be described as

$$\mathcal{I}(x \; is \; an \; outlier) = \begin{cases} 1, & \text{if } p(x|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|Cj|}} \forall j \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where τ_{nov} is the user-specified threshold, C_j is the covariance matrix of mixture component j and D corresponds to the dimensions of the data.

3.1.3 Process to evaluate an incoming datapoint

When a new data point is introduced, it undergoes a process to determine whether it should be considered as an outlier or included in the existing clusters. This process involves inserting the new data point into the **base model** and evaluating its conformity to the established clusters. The decision is made by examining its proximity to the existing clusters using a predefined threshold, typically specified by hyperparameters.

If the new data point is deemed to be within the established clusters according to a certain criterion, denoted by Equation (3.1), it is considered to be a member of the

existing clusters. In this case, the base model is updated using the new data point to refine the parameters of the existing clusters.

However, if the new data point is identified as an outlier based on Equation (3.1), indicating that it does not conform to the existing clusters, an alternative model, referred to as an **auxiliary model**, is employed. This model operates above the base model and is designed to handle outliers or data points that fall outside the scope of the established clusters.

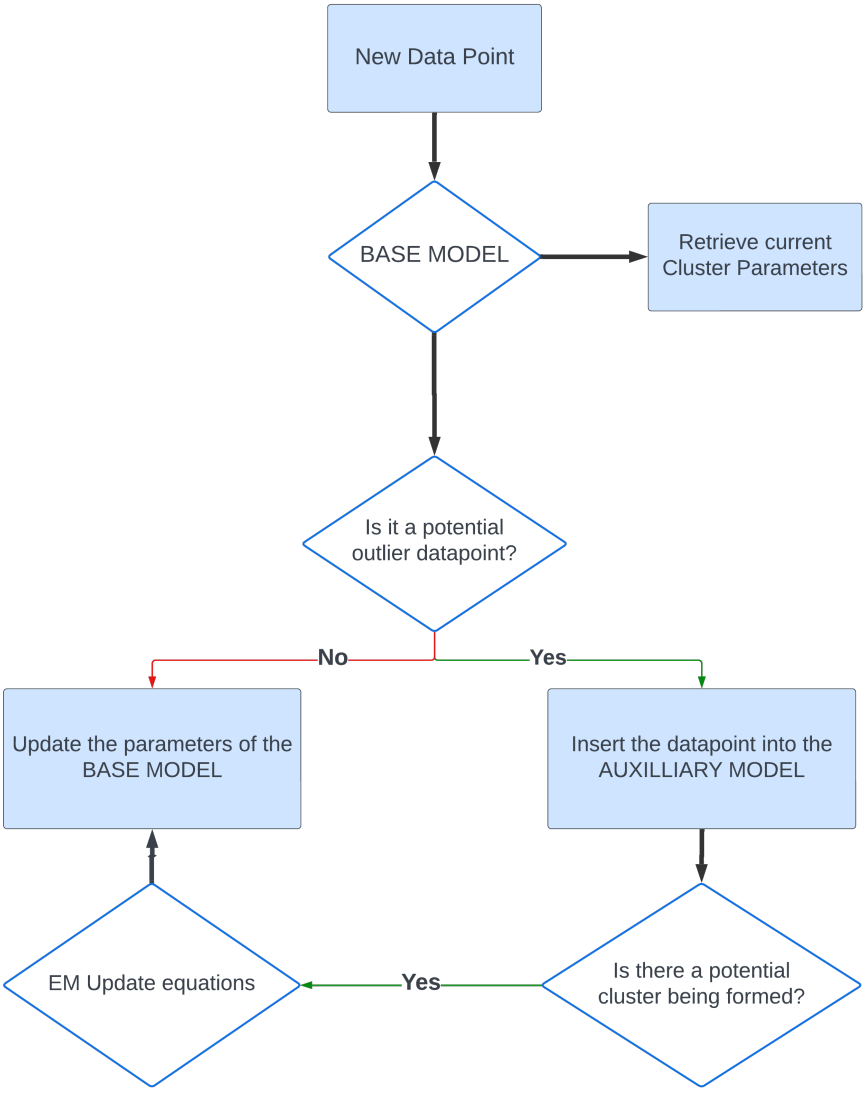


FIGURE 3.1: General Workflow of the Model

The auxiliary model incorporates different threshold hyperparameters compared to the base model, allowing it to adapt to the presence of outliers and adjust its parameters accordingly. Upon identifying an outlier, the auxiliary model processes the data point and applies its threshold criteria. Depending on the outcome, it may recursively return updates to the base model, contributing to the refinement of the clustering process.

We now develop the auxilliary models for both probabilistic detection of the outliers and a density based detection of the outliers.

3.2 A Probabilistic Auxilliary Model: Incremental Gaussian Mixture

When significant amount of outlier data points can be recognised as one cluster whose prior probability crosses a threshold p_{Thv}, these data points start to form a recognised mixture component in the data stream.
Suppose $\Theta_1 = (\pi_j, \mu_j, \Sigma_j)_{j=1,...,m1}$ be the representation of Gaussian mixtures realised by the IGMM framework taking as input the normal data points recognised by the model, where $m1$ is the number of mixture components of the model. Let X_1 be the set of the normal data points in the model. Similarly, suppose $\Theta_2 = (\pi_j, \mu_j, \Sigma_j)_{j=1,...,m2}$ be the representation of Gaussian mixtures realised by the IGMM framework taking as input the outlier data points recognised by our model so far, where $m2$ is the number of mixture components of the model. Let X_2 be the set of outliers realised by the model.
Note that an observation x is probabilistically assigned to a distribution j by the corresponding posterior probability $p(j|x)$. This implies the equivalent number of samples η_j used to compute parameter estimates of the j-th distribution can be described as,

$$\sum_{i=1}^{N} p(j|x_i) \propto \sum_{i=1}^{N} p(x_i|j)p(j) = \eta_j \tag{3.2}$$

where $x_i \in X \ \ \forall i$ and N is the number of outlier data points.
Hence, if η_j crosses a threshold η_{Th}, we can realise and inculcate the component j into our main representation of the model where the representation of the main

model are realised by normal data points.

Therefore, if $\eta_j \geq \eta_{Th}$ for some component $\theta_j \in \Theta_2$, the set of outlier data points, Y which are realised the most by j-th component should be shifted to the normal data set, by updating Θ_1 and Θ_2, while shifting Y from X_2 to X_1.

For this model, the mathematical formulation of Y can be given as

$$Y = (y_i)_{i=1,\ldots,|Y|}, \text{ where } p(y_i|j) = \max_{1 \leq k \leq m2} p(y_i|k) \tag{3.3}$$

where j-th component satisfies

$$\eta_j = \max_{1 \leq k \leq m2} \eta_k \text{ and } \eta_j \geq \eta_{Th} \tag{3.4}$$

The algorithm of using the incremental Gaussian mixture model as the auxilliary model is hence, summarized in Algorithm 6.

Algorithm 6: Two-Fold IGMM using a IGMM Auxilliary Model

Input: X
Parameters: $(\mu_1, \Sigma_1, \pi_1, sp_1), (\mu_2, \Sigma_2, \pi_2, sp_2), \tau_{nov}, \sigma_{ini}, \eta_{Th}, X_1, X_2$
While $x \in X$ **Do**
  Determine if x is an outlier using Eq.(3.1)
  If x is an outlier **Do**
    Append x to X_2
    Update $(\mu_2, \Sigma_2, \pi_2, sp_2)$ using Eq.(2.3) - Eq (2.7)
    Check if there exists a component j satisfying Eq.(3.4)
    If j found **Do**
      Compute Y using Eq.(3.3)
      Update $(\mu_1, \Sigma_1, \pi_1, sp_1)$ using Eq.(2.3) - Eq.(2.7)
      Remove Y from X_2
      Re-estimate $(\mu_2, \Sigma_2, \pi_2, sp_2)$ using **Algorithm 3** for dataset X_2
    End
  else Do
    Append x to X_1
    Update $(\mu_1, \Sigma_1, \pi_1, sp_1)$ using Eq.(2.3) - Eq.(2.7)
  End
End
Output: $(\mu_1, \Sigma_1, \pi_1, sp_1), (\mu_2, \Sigma_2, \pi_2, sp_2)$

3.3 Density-Based Auxilliary Model: Incremental DBSCAN

To address the deficiency in the MIGMM model, particularly its inability to effectively eliminate spurious components in a fast manner, a crucial enhancement involves the auxiliary model undergoing incremental updates of its parameters. Consequently, this section proposes an incremental variant of DBSCAN wherein parameters are iteratively updated with the introduction of each data point. This incremental DBSCAN method operates by adjusting the potential non-outlier cluster in accordance with its mean and covariance, achieving a time complexity of $O(D^3)$, where D represents the number of dimensions within the dataset.

3.3.1 Formulation of an Incremental DBSCAN Algorithm

In this approach, first we form clusters based on the initial objects and a given radius Eps and minimum number of points $MinPts$.Thus we finally get some clusters fulfilling the conditions and some outliers. Now, when new data are inserting into the existing database, we have to update our clusters using DBSCAN. At first we compute the means between every core object of clusters and the new coming data and insert the new data into a particular cluster based on the minimum mean distance. The new data which are not inserted into any clusters,they are treated as noise or outliers. Sometimes outliers which fulfil the $MinPts$ and Eps criteria, combinely can form clusters using DBSCAN.

The psuedocode for the Incremental DBSCAN is summarized in Algorithm 7.

3.3.2 Shifting process of Fault Outliers to the Base Model

Using the incremental DBSCAN as auxilliary model, we not only track the evolution of clusters but also monitor the growth of each cluster by keeping count of the number of data points it encompasses. Additionally, we introduce a parameter, denoted as η, which serves as a threshold representing the desired ratio between the ideal number

of data points required to form a cluster and the total number of data points in the base model.

Algorithm 7: Psuedocode for Incremental DBSCAN Algorithm

Input: D: A dataset containing n objects. $\{X_1, X_2, ..., X_n\}$

n: number of data items

$MinPts$: Minimum number of data objects.

Eps: the radius of the cluster

Start

K $\leftarrow$ already existing clusters

* When new data is coming into the database, the new data will be directly clustered by calculating the minimum Mean (M) between the data and the core object of the existing clusters. *\

For i=1 to n **Do**

Find some mean M in some cluster K_p in K such that $dist(C, M_i)$ is the smallest

if ($dist(C, M_i)$ is minimum) and ($C_i \leq Eps$ and (size(K_p) $\geq MinPts$) then

$K_p \leftarrow K_p \cup C_i$

else

if $dist(C_i! = min)$ or $(C_i > Eps)$ or $(size(Kp) < MinPts)$ then

$C_i \leftarrow Outlier(O_i)$

else

if $Count(O_i) \geq$MinPts$then$

Form new cluster (M_i)

Repeat until all data samples are clustered

End

Output: Clusters $\{C_1, C_2, ..., C_K\}$

During the incremental update process, as we adjust the parameters of each cluster, we continuously evaluate whether the ratio of the number of data points in a given cluster to the total number of data points in the base model exceeds the threshold η. If this condition is met, it indicates that the cluster has grown beyond what is considered acceptable or optimal according to the predefined threshold.

$$\mathcal{I}(C_j \ forming\ a\ potential\ cluster) = \begin{cases} 1, & \text{if } n_j \geq \eta N \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

where n_j is the number of the datapoints in the cluster C_j, N specifies the total number of datapoints in the base model and η is the user-specified ideal ratio for a necessary number of datapoints.

In such cases, rather than retaining the excessively large cluster in the auxiliary model, we employ a corrective action. Specifically, we remove the cluster from the auxiliary model and trigger an update mechanism. This update involves recalculating the parameters of the cluster using the Expectation-Maximization (EM) update equations. These equations take into account the number of data points, their respective means, and the calculated variance, which is typically computed using a scatter matrix of the data points.

Algorithm 8: Two-Fold IGMM using DBSCAN Auxilliary Model

Input: X

Parameters: $(\mu_1, \Sigma_1, \pi_1, sp_1), \tau_{nov}, \sigma_{ini}, \eta_{Th}, X_1, X_2$

While $x \in X$ **Do**

  Determine if x is an outlier using Eq.(3.1)

  If x is an outlier **Do**

    Add x to Incremental DBSCAN using **Algorithm 7**

    Append x to X_2

    For each cluster **Do**

      if a cluster j satisfies Eq.(3.5) **Do**

        Remove cluster datapoints d_{C_j} from X_2 and add it to X_1

        Update the base model using the EM update equations

    End

  else Do

    Append x to X_1

    Update $(\mu_1, \Sigma_1, \pi_1, sp_1)$ using Eq.(2.3) - Eq.(2.7)

  End

End

Output: $(\mu_1, \Sigma_1, \pi_1, sp_1)$

Subsequently, after updating the parameters of the removed cluster, we integrate it back into the base model, thereby ensuring that the model adapts to the changing dynamics of the data distribution. This process not only helps maintain the integrity of the clustering model but also prevents the formation of overly large or unwieldy clusters that could potentially skew the clustering results.

By incorporating this incremental DBSCAN framework with the base model, we enhance its robustness and scalability, enabling it to effectively handle evolving datasets while maintaining the quality and coherence of the clustering results.

Chapter 4

Implementation and Results

All the implementation and the empirical analysis of the experiments is publicly available on the Github repository here.(Link: https://t.ly/uLsEo)

4.1 Evaluation of Models on a Synthetic Dataset

4.1.1 Limitations of IGMM and MIGMM

While the incremental Gaussian mixture model adopts an efficient estimation approach which can be expanded to accommodate new information from an input data-point, it is sensitive to the τ_{nov}-threshold, the novelty criterion which acts as a key-parameter in determining whether an incoming data point x is accepted in the set of originally formed clusters or not.
Empirical analysis over synthetic data set shows that a slight perturbation in the value of τ_{nov} significantly changes the estimation of the parameters as well as the total number of clusters.
We performed a simulation of 1000 points from a one-dimensional two-component GMM. Fig. 4(a) shows the initial GMM source used to realise the data stream.
Fig. 4(b),(c),(d) show the different results when the model is subjected to different values of τ_{nov}. We can clearly see that in Fig.4(d), the estimation has clearly failed

sure to considerably large value of τ_{nov}. This is an indication that significant importance to the novelty criterion must be ensured for the efficiency of IGMM.

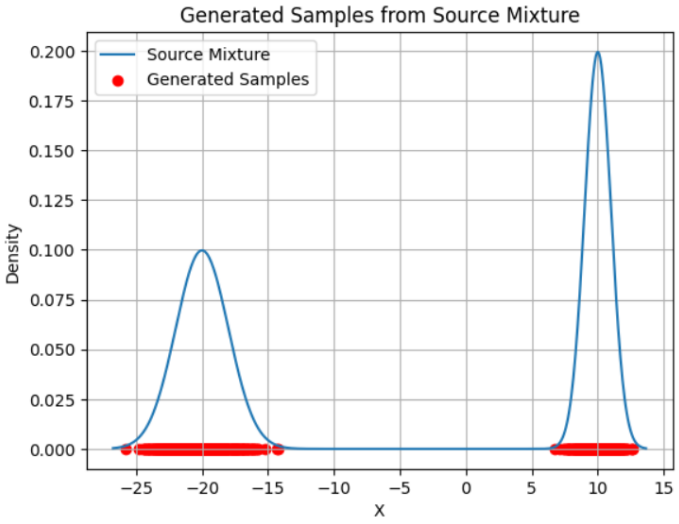
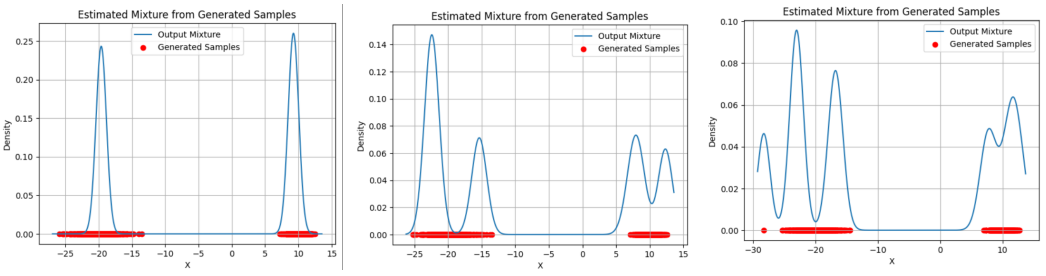


FIGURE 4.1: Initial GMM with sampling points 1000

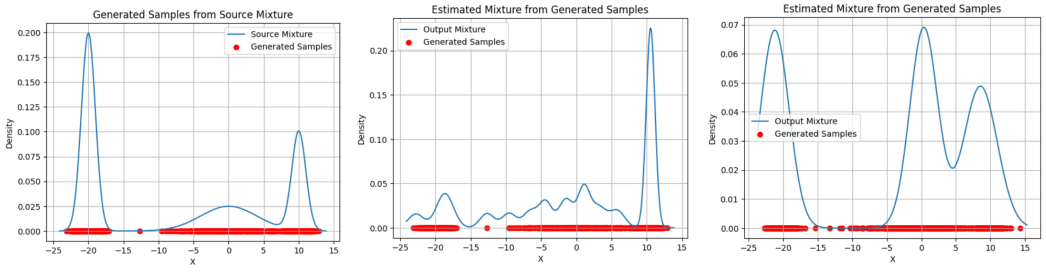


(a) $\tau_{nov} = 0.1$ (b) $\tau_{nov} = 0.3$ (c) $\tau_{nov} = 0.45$

FIGURE 4.2: Estimation showing the sensitivity of τ_{nov} on IGMM

Moreover, if x is rejected, it creates a new cluster having the mean to be the value of a user-determined σ_{ini} variance. Experimental analysis show that the convergence rate to the optimal estimation is greatly affected by σ_{ini}. If $|\sigma_{ini} - \Sigma_{m0}|$ is significantly large, the recursive update equation require more iterations, and henceforth, more number of data points to converge to Σ_{m0}, where Σ_{m0} can be regarded as the optimal covariance matrix for some cluster m. This reduces the convergence rate significantly. Fig 4.3 shows the initial GMM source as well as the different results of the IGMM when subjected to different σ_{ini} values. It is quite evident in Fig. 4.3(b), the

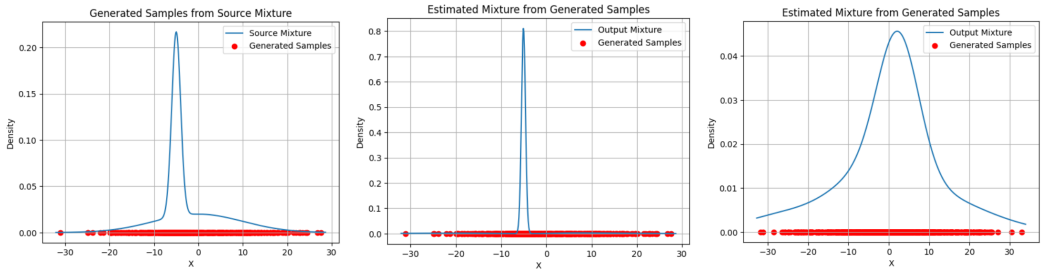
estimation has suffered a significant loss when σ_{ini}=1.



(a) Initial GMM (b) $\sigma_{ini} = 1$ (c) $\sigma_{ini} = 1.5$

FIGURE 4.3: Estimation showing the sensitivity of σ_{ini} on IGMM

Another limitation for the IGMM model succumbs to the source Gaussian mixture being non-uniform. If there exists a significant difference between the covariances of the mixture components, only one initial value of σ_{ini} may fail to realise a component which has a covariance significantly different from the initially assumed σ_{ini}.
Fig. 4.4 shows an initial GMM consisting of two-components with $\Sigma_1 = 1$ and $\Sigma_2 = 10$. This shows that the components are not realised if their covariances are far away from what the initially assumed covariances.



(a) Initial GMM (b) $\sigma_{ini} = 1$ (c) $\sigma_{ini} = 2.5$

FIGURE 4.4: Estimation of IGMM on differing component covariances

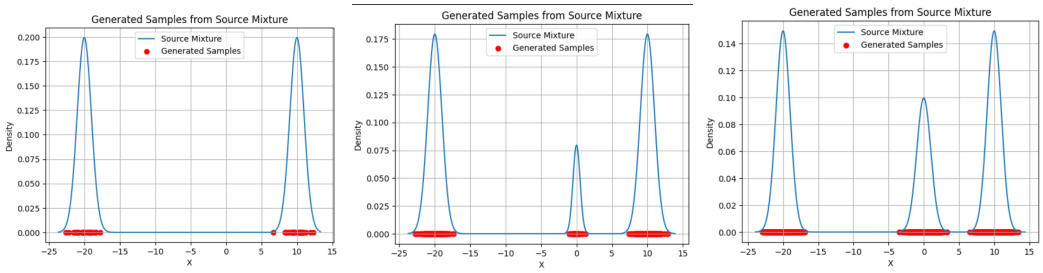
Regarding the modified-incremental Gaussian mixture model, while it shows a great efficiency in removing the spurious components and determining the correct output number of components, the major limitation of MIGMM relies on the fact that while it achieves a good accuracy, it needs to have a short runtime. MIGMM algorithm

takes as input an online data-stream in its first stage whilst estimating the parameters of the model, and removes the spurious components in the later and final stage. This procedure has two major limitations:

i. A full-stream of input has to be given to the algorithm, which would then be used to estimate the correct number of components and optimal parameters. Hence, this partially limits the model for short-time data streams.

ii. If a long-time data-stream is continuously passed through the model by diving it into blocks of shorter data-streams, the model fails to recognise the cluster formed by the data points which would be evenly distributed among the short data-streams but when accumulated, would have a significant contribution to identify as a new cluster.

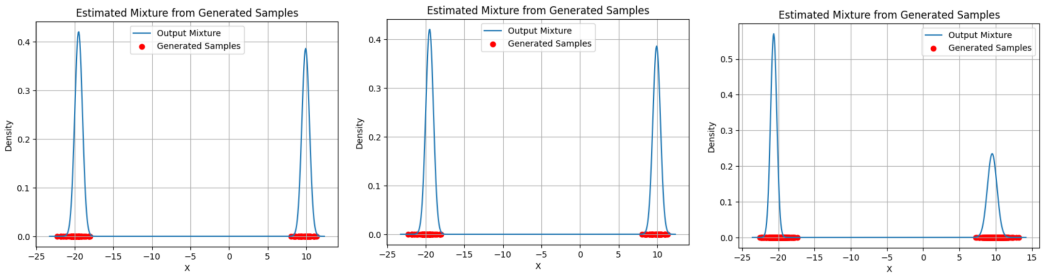
Henceforth, MIGMM fails to adapt the correct number of components as well as the efficient values of parameters when subjected to a longer and irregular data-stream. Fig. 4.5 shows the estimation of data stream through EM algorithm for different periods. A new cluster begins to realise as more data points are streamed into the model.



(a) Sampling of 100 datapoints (b) Sampling of 500 datapoints (c) Sampling of 5000 datapoints

FIGURE 4.5: Estimation through EM Algorithm using an irregular data stream

Fig. 4.6 shows the estimation of parameters when each of these set of data points is passed through MIGMM, which is reset after 200 datapoints. It is evident that it identifies the new datapoints as outliers, and removes the p_k prior probability which is contributed by these set of data points.



(a) $n = 100$ (b) $n = 500$ (c) $n = 5000$

FIGURE 4.6: Estimation through MIGMM using an irregular data stream

4.1.2 Evaluation of two-fold IGMM with other unsupervised models

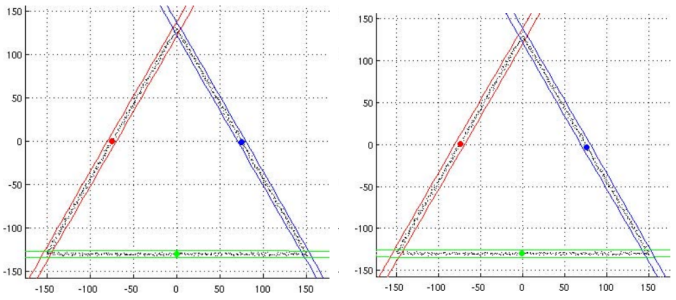
This section describes the experiments accomplished with the proposed model to evaluate its performance and to compare it with the MIGMM algorithm. In these experiments, we empirically assess the accuracy of our model and how close a mixture model is to the Gaussian mixture source. It is important to say that no exhaustive search is performed to train and optimize the hyperparameters.

The first experiment realises data points from a two-dimensional three-component Gaussian mixture model, which does not succumb to outliers forming a cluster. Table 1 shows the results obtained in this experiment.

Actual Parameters of Distribution						
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cluster 1	0.3333	-75.00	0.00	1850.00	3200.00	5600.00
Cluster 2	0.3333	75.00	0.00	1800.00	-3200.00	5600.00
Cluster 3	0.3333	0.00	-130.00	7500.00	-1.0	0.00
Model parameters estimated by MIGMM Algorithm						
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cluster 1	0.3651	-75.22	-0.47	1874.4	3245.9	5523.8
Cluster 2	0.2996	75.12	-0.12	1214.6	-3186.6	5421.6
Cluster 3	0.3353	-0.18	-128.84	5112.4	-1.6	1.86
Model parameters estimates by two-fold IGMM Algorithm						
	$p(j)$	μ_{j1}	μ_{j2}	c_{j11}	c_{j12}	c_{j22}
Cluster 1	0.3375	-75.22	-0.92	1924.6	3316.8	5321.9
Cluster 2	0.2167	76.14	0.21	1254.2	-3712.4	5128.7
Cluster 3	0.4458	-0.23	-129.92	4157.8	-0.85	0.92

TABLE 4.1: Model parameters computed for the triangular dataset

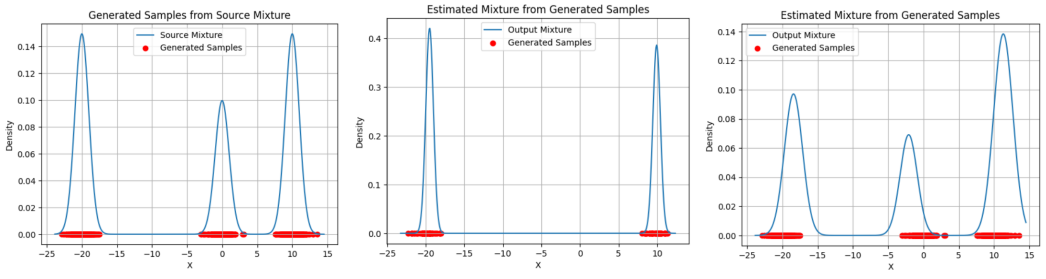
It can be noticed that parameters computed by the algorithm are very close in terms of computing the means of the mixture components, but show significant difference in computing the prior probabilities and covariances. Fig. 4.7 shows the data points and the model parameters obtained by the experiment. It is evident that the distribution are elongated, but the model were capable to align components with the actual principal axis.



(a) MIGMM Model (b) 2-Fold IGMM Model

FIGURE 4.7: Concepts computed using the triangular dataset

The second experiment realises data points from a one-dimenstional three-component Gaussian mixture model, which forms one cluster at a later stage. Fig. 4.8 shows the data points and the model parameters obtained by the experiments. It shows the inefficiency of MIGMM to not realise the concept of outliers forming a cluster since it needs to be reset after a few iterations, partially destroying the property of online-training. The two-fold IGMM model computes the stream parallely, and realises the concept, but it fails to develop the parameter values with more significant accuracy.



(a) Actual Model (b) MIGMM Model (c) 2-Fold IGMM Model

FIGURE 4.8: Concepts computed using a one-dimensional dataset

4.1.3 Quantitative Evaluation of the Model

4.1.3.1 Silhouette Scores

The silhouette score is a metric used to measure the quality of clustering in a dataset. It provides a way to assess both the cohesion and separation of the clusters formed by a clustering algorithm. A higher silhouette score indicates better-defined clusters.

Procedure to evaluate the model

1. **Calculate the Silhouette Coefficient for Each Data Point**:
   - For each data point i, calculate the silhouette coefficient $s(i)$.
   - The silhouette coefficient for a single data point i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

     where:
     - $a(i)$ is the average distance from i to all other points in the same cluster.
     - $b(i)$ is the smallest average distance from i to all points in any other cluster, where i is not a member.
2. **Calculate the Silhouette Score for the Entire Dataset**:
   - The silhouette score for the entire dataset is the mean of the silhouette coefficients for all data points.
   - It can be calculated as:

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

     where N is the total number of data points.

4.1.3.2 Davies-Bouldin Index

The Davies-Bouldin Index (DBI) is a metric used to evaluate the effectiveness of a clustering algorithm. It measures the average similarity between each cluster and its most similar cluster, while also considering the dispersion within the clusters. Lower DBI values indicate better clustering.

1. **Calculate the Cluster Dispersion**: For each cluster, calculate the dispersion or spread of data points within that cluster. This can be measured using a metric such as the average distance between each data point and the centroid of the cluster.
2. **Calculate the Cluster Separation**: For each pair of clusters, calculate the dissimilarity or distance between their centroids. This can be measured using a distance metric such as Euclidean distance.
3. **Compute the Davies-Bouldin Index (DBI)**: For each cluster i, calculate the Davies-Bouldin Index R_i using the formula:

$$R_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \max_{j \neq i} \left(\frac{s_i + s_j}{d(c_i, c_j)} \right)$$

   Where:

   - n_i is the number of clusters.
   - s_i is the average dispersion of cluster i.
   - c_i is the centroid of cluster i.
   - $d(c_i, c_j)$ is the distance between the centroids of clusters i and j.
4. **Average the Indices**: Calculate the average of all R_i values to obtain the overall Davies-Bouldin Index for the clustering solution.

4.1.3.3 Clustering Analysis for the Model

To give a more detailed quantitative analysis of the model, we implement the following steps, calculating the silhouette scores for the model:

1. **Dataset Generation**: We generate a synthetic dataset tailored to adhere to the assumptions of the Gaussian Mixture Model (GMM). This ensures that the dataset is amenable to clustering analysis using GMM.
2. **GMM Application with Varying Cluster Numbers**: We employ the classical GMM algorithm on the synthetic dataset, varying the number of components (clusters) across a predefined range. This step allows us to explore the clustering behavior under different cluster configurations.
3. **Silhouette Score Calculation**: For each GMM model with a different number of components, we calculate silhouette scores. The silhouette score quantifies the quality of clustering, providing insight into the cohesion and separation of clusters.

4. **Silhouette Score Visualization**: We create a plot illustrating the relationship between the silhouette scores and the number of components in the GMM. This visual representation aids in understanding how the clustering quality evolves with varying cluster numbers.
5. **Model Implementation and Silhouette Score Calculation**: We implement our clustering model on the same dataset, ensuring it aligns with the optimal number of clusters identified from the silhouette score analysis. This step involves calculating silhouette scores to evaluate the clustering performance of our model.
6. **Consistency Evaluation**: By comparing the silhouette scores obtained from our model with those generated using the classical GMM across different cluster numbers, we assess the consistency of our model's performance and validate the optimal number of clusters.
7. **Insight Generation**: The analysis of silhouette scores and the comparison between different clustering configurations provide valuable insights into the clustering behavior of the dataset.

Following the above steps, we find that the optimal number of clusters generated by the model and the silhouette score roughly match with a certain amount of variation to that of the performance determined by the classical Gaussian Mixture Model determined by EM Algorithm.

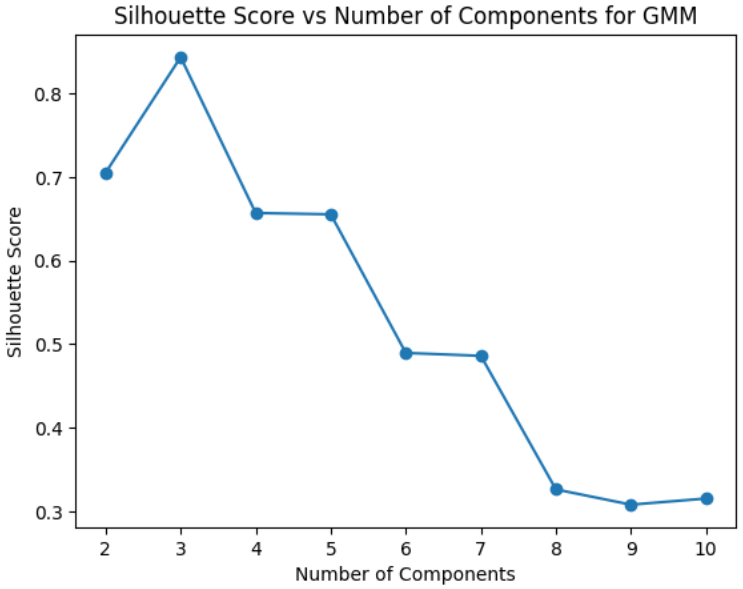


FIGURE 4.9: Silhouette Scores to find optimal number of components

Corresponding to this generated data, we finally plot the changes in the Davies-Bouldin index by varying the hyperparameters of the model.

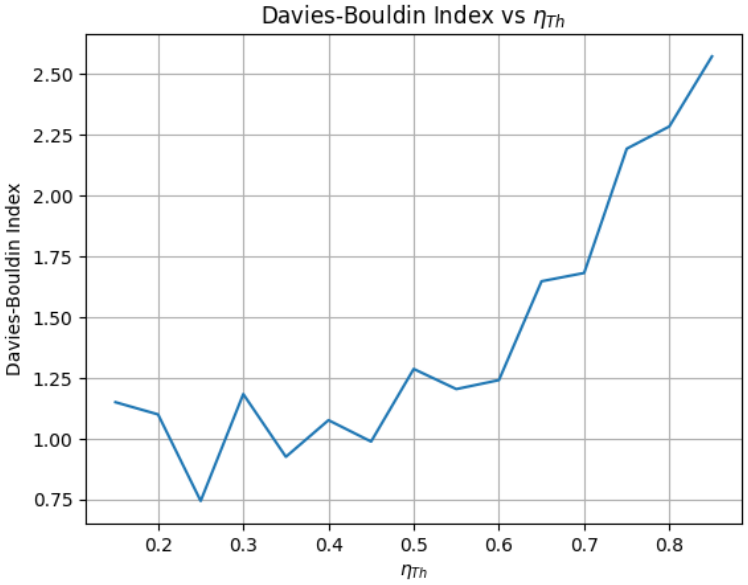


FIGURE 4.10: DB Index Scores v/s Value of η_{Th}

We find that the efficiency of the two-fold incremental framework, while performing comparably to most of the classical algorithm, suffers from a high sensitivity to the parameters η_{th} and τ_{nov}.

4.2 Implementation of the Model on a Financial Dataset

We gathered the data of daily stocks of financial returns of 5 Indian stocks, of the past 10 years, and we ran our model to cluster these financial returns, to distinguish the similarity and variability between the operation of each stock.

4.2.1 Preprocessing of the Dataset

Given the dataset, we first extracted some financial features which are described as follows:

a) **Simple Moving Average (SMA):**
SMA is a basic yet widely used technical analysis tool. It smoothens out price data by creating a constantly updated average price over a specific time period. It helps in identifying trends and potential support/resistance levels.
Calculation: Add up the closing prices over a specific number of periods and divide by that number of periods.

b) **Exponential Moving Average (EMA):**
EMA gives more weight to recent prices, making it more responsive to changes in price direction compared to SMA. Traders often use EMAs to identify trend direction and potential entry/exit points.
Calculation: Similar to SMA, but with more weight assigned to recent prices. The formula is more complex, involving exponential smoothing.

c) **Relative Strength Index (RSI):**
RSI is a momentum oscillator that measures the speed and change of price movements. It oscillates between 0 and 100 and is used to identify overbought or oversold conditions in a market.
Calculation: Calculate the average of up closes and down closes over a specific period, then use these averages to calculate the Relative Strength (RS). Finally, use the RS to calculate the RSI.

d) **Moving Average Convergence Divergence (MACD) Line:**
MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. It helps traders identify changes in trend direction, momentum, and potential buy/sell signals.
Calculation: Subtract the longer-term EMA from the shorter-term EMA to get the MACD line.

e) **MACD Signal Line:**
The MACD signal line is the exponential moving average of the MACD line. It's used to generate buy and sell signals when it crosses above or below the MACD line.
Calculation: Calculate the EMA of the MACD line over a specified period to get the MACD signal line.

After extracting financial features, it's essential to preprocess the data to ensure optimal performance in downstream tasks like clustering. One common preprocessing step is normalization, which involves scaling the features to have a mean of 0 and a standard deviation of 1. This step is crucial for preventing any single feature from dominating the clustering process.

We use the scikit-learn library providing a convenient tool for normalization called `StandardScaler`.

This function scales each feature independently by subtracting the mean and dividing by the standard deviation. This ensures that all features have a similar scale, preventing any particular feature from having undue influence on the clustering algorithm.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
normalized_data = scaler.fit_transform(data)
```

`data` represents our feature matrix, where each row corresponds to a data point and each column represents a different feature. The `StandardScaler` instance is then used to fit the scaler to the data and transform it to obtain the `normalized_data`.

By normalizing the data with `StandardScaler`, we ensure that each feature contributes equally to the clustering process, leading to more reliable and meaningful results.

4.2.2 Evaluation of two-fold framework on a real-world data

In our analysis of the financial dataset, we begin by employing the Expectation-Maximization (EM) Algorithm to derive optimal results. Subsequently, we apply a two-fold Integrated Gaussian Mixture Model (IGMM) framework to the normalized dataset. The resultant cluster parameters are meticulously detailed in the Table 4.2.

Model Parameters estimated by EM Algorithm							
	$p(j)$	μ_{j1}	μ_{j2}	μ_{j3}	μ_{j4}	μ_{j5}	μ_{j6}
Cluster 1	0.28143	1.3738	1.3749	0.13307	0.30009	0.3143	0.01621
Cluster 2	0.71857	-0.53807	-0.5385	-0.05211	-0.1175	-0.1231	-0.00635
Model parameters estimates by two-fold IGMM Algorithm							
	$p(j)$	μ_{j1}	μ_{j2}	μ_{j3}	μ_{j4}	μ_{j5}	μ_{j6}
Cluster 1	0.09906	1.57495	1.5714	0.0282	0.5099	0.6459	-0.3311
Cluster 2	0.18028	1.2909	1.2944	0.1862	0.2045	0.1581	0.1911
Cluster 3	0.72065	-0.5394	-0.5398	-0.05046	-0.1213	-0.12834	-0.0023

TABLE 4.2: Model parameters computed for the financial dataset

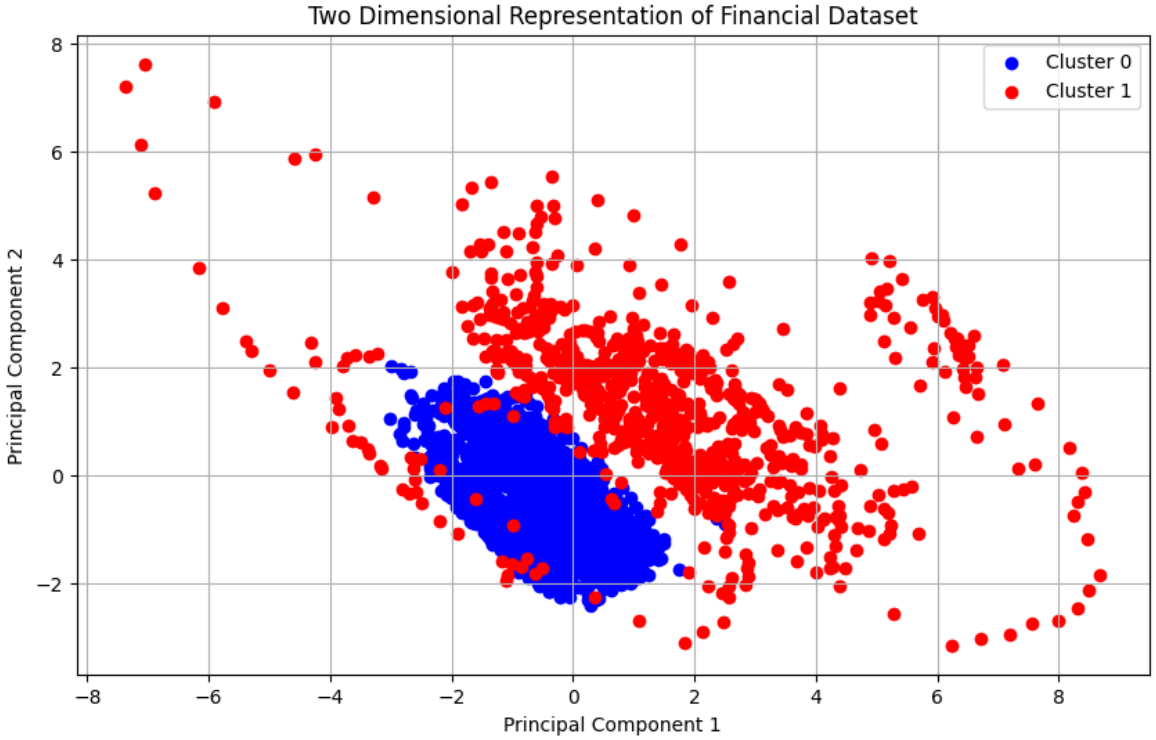


FIGURE 4.11: Performance of EM Algorithm

To enhance our understanding of the dataset's inherent structure and discern the clustering patterns, we utilize Principal Component Analysis (PCA). PCA enables us to transform the multidimensional financial dataset into a comprehensible two-dimensional representation. The resultant figure vividly illustrates the clusters formed by both the EM Algorithm and the two-fold IGMM framework, providing valuable insights into the underlying data distribution and clustering behavior.

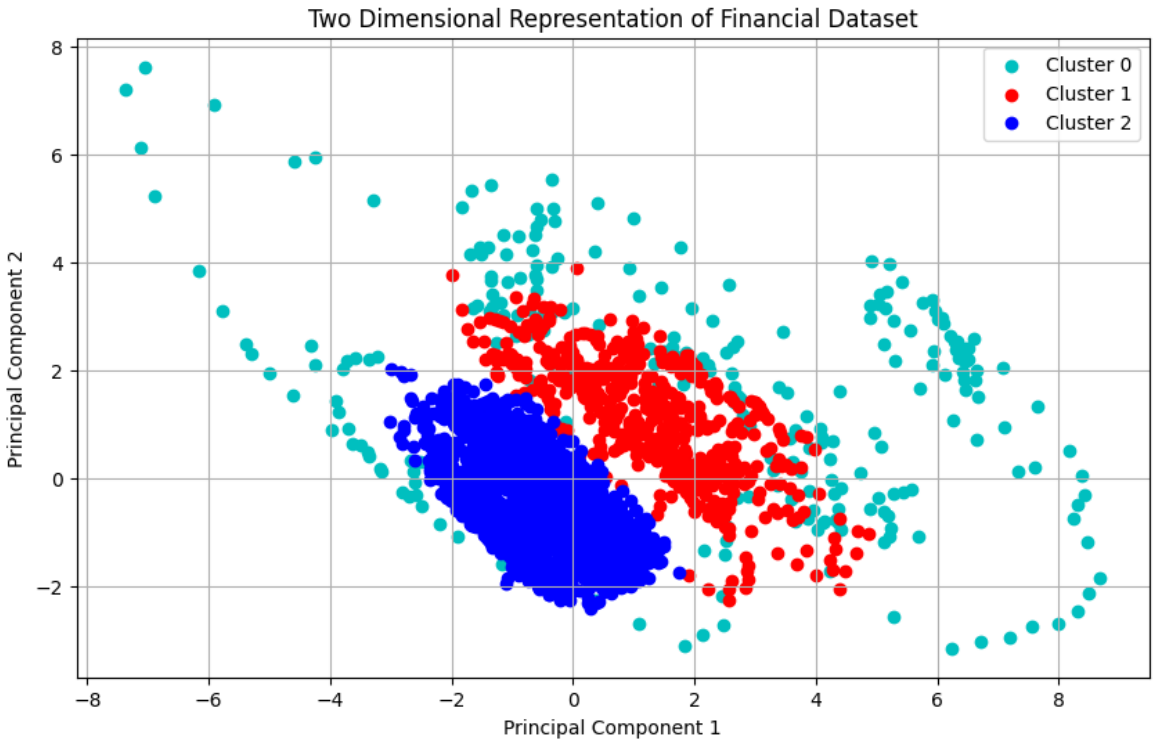


FIGURE 4.12: Performance of Two-Fold IGMM

The clustering process applied to the current dataset reveals limited significance in reflecting the behavior of financial data. To extract more meaningful insights, it is imperative to transition towards utilizing real-time datasets with updated financial features. By leveraging such datasets, we can identify features that are more closely aligned with real-world market dynamics. This shift allows for a more robust evaluation of the clustering process, enabling us to derive significant insights and compare them with current market trends and conditions. Additionally, real-time datasets offer the advantage of capturing the most recent market movements and developments, enhancing the relevance and practical significance of the clustering analysis. Therefore, the exploration and extraction of features from real-time datasets are essential steps towards gaining valuable and actionable insights from the clustering process in the context of financial analysis

Chapter 5

Conclusion and Future Directions

In this study, we proposed a novel two-fold partition framework called the two-fold incremental Gaussian mixture model. The major contributions in this study are

i. Development of a model which can handle a continuous, but irregular flow of a data stream, and

ii. the classification of outliers through a given input of data stream.

The performed experiments and evaluation of the model on a synthetically created data set as well as a financial dataset collected over 10 years of 5 different stocks after a single pass through the data closely resemble to the ones computed by the EM algorithm(Moon (1996)) after 100 iterations over the same data.

An efficient criterion to select hyperparameters through the sensitivity analysis of τ_{nov}, σ_{ini} and η_{Th} using hyper parameter optimizations(Bergstra et al. (2011)), as well as a comparison between the convergence rate of the model and σ_{ini} also warrant a further research. Application of the model to more general real-time applications such as high-frequency trading dataset to establish the mutual information which potentially correlate the market trends of stock in different market regimes is a potential future direction. Furthermore, an approach to improvise the hyperparameters of the two-fold incremental framework with the help of neural networks using multiple passes on the data to improve the accuracy, and the reduction of computational complexity of the model also are some of the potential research directions.

Bibliography

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24(2):12–21.

Engel, P. M. and Heinen, M. R. (2010). Incremental learning of multivariate gaussian mixture models. *Advances in Artificial Intelligence–SBIA 2010: 20th Brazilian Symposium on Artificial Intelligence, São Bernardo do Campo, Brazil*, 2(2):82–91.

Khan, K., Rehman, S. U., Aziz, K., Fong, S., and Sarasvady, S. (2014). Dbscan: Past, present and future. *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, 9(5):232–238.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60.

Pinto, R. C. and Engel, P. M. (2015). A fast incremental gaussian mixture model. *PloS one*, 10(10):30–42.

Reynolds, D. A. et al. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, 7(4):659–663.

Sun, S., Tong, Y., Zhang, B., Yang, B., Yan, L., He, P., and Xu, H. (2023). A novel adaptive methodology for removing spurious components in a modified incremental gaussian mixture model. *International Journal of Machine Learning and Cybernetics*, 14(2):551–566.