



# Partitioning of large hyperspectral image data streams

Yuding Wang

## ► To cite this version:

Yuding Wang. Partitioning of large hyperspectral image data streams. Signal and Image processing. Université de Rennes, 2023. English. NNT : 2023URENS071 . tel-04446348

**HAL Id: tel-04446348**

**<https://theses.hal.science/tel-04446348>**

Submitted on 8 Feb 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THESE DE DOCTORAT DE

## L'UNIVERSITE DE RENNES 1

ECOLE DOCTORALE N° 601

*Mathématiques, Télécommunications, Informatique,  
Signal, Systèmes, Electronique*  
Spécialité : *Signal, Image, Vision*

Par

**Yuding WANG**

## Partitioning of large hyperspectral image data streams

Thèse présentée et soutenue à l'université de Rennes 1, le 15 décembre 2023

Unité de recherche : IETR (UMR CNRS 6164)

### Rapporteurs avant soutenance :

Jenny BENOIS-PINEAU

Gabriela CIUPERCA

Professeur / Université Bordeaux 1

MC, HDR / Université Claude Bernard Lyon 1

### Composition du Jury :

Président : Franck MARZANI

Examineurs : Jenny BENOIS-PINEAU

Gabriela CIUPERCA

Dir. de thèse : Kacem CHEHDI

Professeur / Université de Bourgogne

Professeur / Université Bordeaux 1

MC, HDR / Université Claude Bernard Lyon 1

Professeur / Université de Rennes

# Table of contents

Résumé en Français .....	2
Acknowledgments .....	6
Notation.....	7
List of abbreviations .....	10
1 Introduction .....	12
2 State of the art.....	15
2.1 Similarity criteria .....	15
2.2 Static dataset partitioning methods.....	17
2.2.1 Semi-supervised methods .....	17
2.2.2 Unsupervised methods .....	20
2.3 Data stream partitioning methods .....	28
2.3.1 Semi-supervised methods .....	28
2.3.2 Unsupervised methods .....	31
2.4 Partitioning validity indices .....	40
3 Developed unsupervised data stream partitioning method (STRFCM) .....	44
3.1 Principle.....	44
3.2 Static dataset partitioning methods employed in STRFCM.....	46
3.2.1 Choice of FCMO for optimal partition .....	46
3.2.2 WFCMO for final optimal partition.....	49
3.2.3 Evaluation of FCMO .....	51
3.2.3.1 Experimental datasets .....	51
3.2.3.2 Performance evaluation.....	54
3.2.4 Evaluation of WFCMO .....	62
4 Evaluation of developed method (STRFCM).....	64
4.1 Evaluation protocol .....	64
4.2 Evaluation on hyperspectral image of algae .....	65
4.3 Evaluation on hyperspectral image of invasive vegetation.....	72
4.4 Evaluation on Image Segmentation dataset .....	79
4.5 Discussion.....	80
5 Conclusion and perspectives .....	81
5.1 Conclusion .....	81
5.2 Perspectives .....	82
Bibliography.....	83
List of figures .....	91
List of tables .....	93

# Résumé en Français

Ces dernières années, le traitement de grandes quantités de données a suscité une attention considérable. Le traitement des flux de données hyperspectrales en fait partie pour de nombreuses applications, telles que la surveillance environnementale, la détection de cibles et d'anomalies, la cartographie géologique et l'étude de la végétation. Actuellement, la majorité du traitement des données hyperspectrales se fait hors ligne, après la fin de la mission d'acquisition des données. En effet, il est difficile de traiter et d'analyser en ligne les données présentant des dimensions spatiales et spectrales étendues.

Le partitionnement de flux de données utilise des données ou des lots de données qui arrivent en continu et les attribue à des classes sans avoir observé l'intégralité des données. Nous précisons que "le partitionnement" est une opération de subdivision d'un ensemble de données en classes homogènes pour former une partition. La Figure 1 illustre un exemple de partitionnement d'objets caractérisés par deux attributs où chaque classe présente peut être représentée par son centroïde (légende \*), également appelé exemplaire.

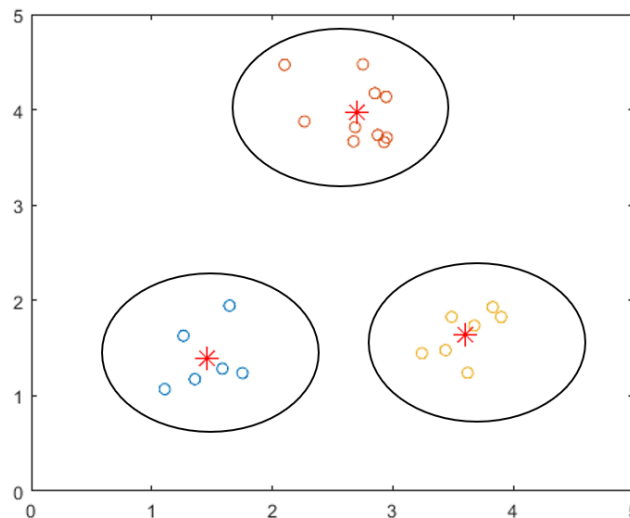


Figure 1 : Partitionnement d'objets en trois classes. A chaque classe est associé son centroïde ou exemplaire (\*).

Pour le partitionnement des données, plusieurs méthodes existent. Elles peuvent être divisées en trois catégories : supervisées, semi-supervisées et non supervisées. Les méthodes supervisées utilisent des échantillons d'apprentissage pour effectuer les partitionnements. Cependant, dans de nombreux contextes réels, l'obtention de données étiquetées peut être difficile, voire impossible pour certains domaines applicatifs. Les méthodes semi-supervisées ne nécessitent pas d'échantillons d'apprentissage mais exigent une connaissance *a priori* du

nombre de classes, tandis que les méthodes non supervisées ne nécessitent aucune connaissance *a priori*. En effet, pour certaines applications, le nombre de classes est souvent difficile à déterminer à l'avance et peut évoluer avec l'arrivée de nouvelles données, ce qui devient l'une des principales limitations des méthodes semi-supervisées. Par conséquent, les méthodes non supervisées sont mieux adaptées pour partitionner des flux de données.

Dans l'état de l'art, la plupart des méthodes sont paramétriques, ce qui nécessite que les utilisateurs spécifient un ou plusieurs paramètres et/ou le nombre de classes avant le processus de partitionnement. Cependant, l'utilisation de paramètres peut introduire un certain degré d'incertitude dans les performances des méthodes. Les limitations des méthodes paramétriques sont :

- 1) Le choix des valeurs de paramètres avant le processus de partitionnement qui peut avoir un impact plus ou moins important sur les performances de partitionnement. Différentes combinaisons des réglages de paramètres peuvent conduire à des résultats de partitionnement variables.
- 2) Le réglage empirique des paramètres qui peut être complexe et chronophage s'amplifiant avec les méthodes qui comportent plusieurs paramètres.
- 3) La difficulté de trouver des valeurs appropriées des paramètres lorsqu'il s'agit de partitionner des flux d'objets nouveaux.

Pour remédier à ces inconvénients, nous avons développé une approche non supervisée et non paramétrique appelée STRFCM (STReam Fuzzy C-Means). Cette méthode est capable d'estimer de manière automatique le nombre de classes et de partitionner les flux de données sans nécessiter de connaissance *a priori*, ni l'introduction de paramètres. Elle se compose de deux étapes, partitionnement des blocs de données et fusion des classes des blocs pour la formation de la partition finale :

- 1) **Partitionnement des blocs de données** : cette première étape emploie l'algorithme « Fuzzy C-Means-Optimized » (FCMO) [11], qui utilise la norme  $L_1$  comme critère de similarité et un indice de validité adaptatif de partition (noté  $WB-L_{IM}$ ) comme critère d'évaluation. Cet indice utilise le rapport entre la dispersion intra-classe et la dispersion inter-classes avec une pondération par la variance des classes. Cela permet d'obtenir des classes assez représentatives des blocs de données ainsi que leurs exemplaires.

FCM est une méthode semi-supervisée adaptée à l'origine pour partitionner des données statiques en un nombre prédéterminé de classes. FCMO optimise cette méthode en estimant le nombre de classes sans connaissance *a priori*, en intégrant une procédure incrémentale adaptative. Cet algorithme utilise la norme  $L_2$  [32], comme critère de similarité et la différence entre la dispersion globale inter-classes et intra-classe d'une partition comme critère d'évaluation (noté  $F$ ). La maximisation de ce critère non supervisé donne la partition optimale.

Pour une meilleure discrimination entre les objets sans amplification ni atténuation du critère de similarité et donc la formation de classes homogènes, la norme  $L_1$  [32] a été utilisée comme critère au lieu de la norme  $L_2$ . De plus, pour renforcer la préservation de l'ensemble des informations caractéristiques d'un bloc, nous avons proposé un nouvel indice appelé  $WB-L_{IM}$  comme critère d'évaluation de FCMO. La minimisation du critère  $WB-L_{IM}$  permet de déterminer une partition formée des classes fortement homogènes ainsi que l'exemplaire de chacune d'elles.

- 2) **Fusion** : cette étape emploie l'algorithme FCMO pondérée (WFCMO) pour partitionner l'ensemble des exemplaires obtenus dans la première étape en prenant en compte la taille des classes qu'ils représentent. Il utilise la norme  $L_1$  comme critère de similarité et l'indice  $F$  comme critère d'évaluation. Les résultats obtenus par l'algorithme WFCMO surpassent ceux de FCMO lors de la partition de l'ensemble des exemplaires.

L'organigramme de la méthode proposée est présenté dans la Figure 2.

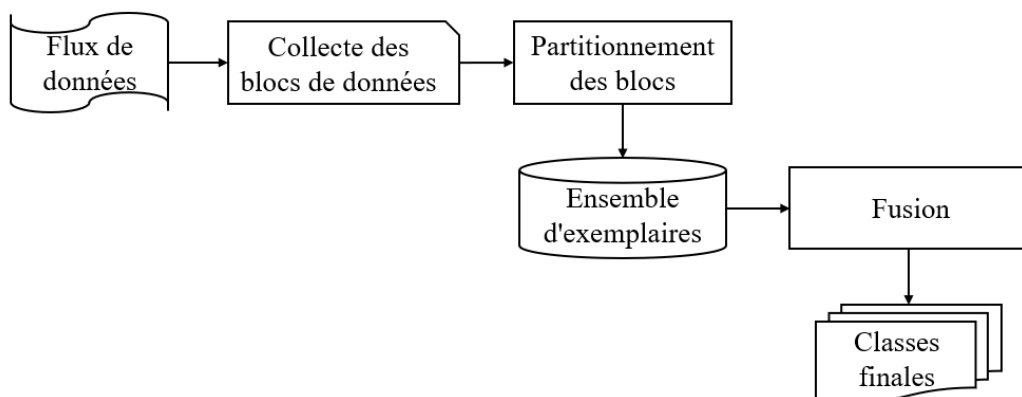


Figure 2 : Organigramme de la méthode de partitionnement non supervisée proposée.

La méthode proposée a été évaluée et comparée à trois méthodes paramétriques non supervisées et une méthode semi-supervisée, ainsi qu'à une méthode supervisée utilisant l'apprentissage actif. Cette évaluation a été réalisée sur des images hyperspectrales synthétiques et une base de données réelles en utilisant des critères objectifs comme la pureté, l'indice Kappa et l'instabilité des résultats causée par les différents réglages des paramètres. Le temps pour le réglage empirique des paramètres est également intégré. Les résultats montrent que notre méthode est plus performante que les méthodes comparées.

En conclusion, une nouvelle méthode non supervisée et non paramétrique appelée STRFCM a été développée pour le partitionnement de flux de données. Cette méthode est facile à appliquer par les utilisateurs, car elle élimine le recours à des connaissances *a priori* et évite la nécessité du réglage empirique des paramètres. Elle peut en effet, estimer de manière automatique le nombre de classes et a montré son efficacité pour le partitionnement de flux de données de grandes tailles spatiale et spectrale, en particulier les données hyperspectrales.

Le manuscrit de cette thèse est organisé en cinq chapitres :

*Le premier chapitre* sert d'introduction au contexte de la recherche visée et à la structure de la thèse.

*Le deuxième chapitre* passe en revue les principales méthodes de partitionnement des données statiques et des flux de données de l'état de l'art. Différentes catégories de méthodes sont étudiées en examinant leurs avantages et leurs limites respectifs. De plus, ce chapitre introduit un éventail d'indices objectifs pour la validation d'une partition.

*Le troisième chapitre* présente la méthode développée. Il introduit tout d'abord le principe de la méthode. Ensuite, nous décrivons et évaluons les améliorations apportées aux méthodes adaptées pour le partitionnement des données statiques qui sont utilisées dans la méthode proposée.

*Le quatrième chapitre* présente l'évaluation de la méthode développée incluant les comparaisons avec cinq méthodes de partitionnement de flux de données de l'état de l'art (une méthode supervisée utilisant l'apprentissage actif, une méthode semi-supervisée et trois méthodes paramétriques non supervisées). Les différents tests sont effectués sur des images hyperspectrales et sur une base de données largement utilisées par la communauté scientifique comme référence pour l'apprentissage automatique. Deux applications sont traitées dans le cas de partitionnement des images hyperspectrales. Le premier concerne la détection de plantes invasives et le second la détection d'algues marines.

*Le cinquième chapitre* conclut notre recherche et évoque les perspectives.

# Acknowledgments

I would like to express my sincere gratitude to the individuals who have been instrumental in the successful completion of my PhD journey. First and foremost, I extend my deepest appreciation to my thesis director, Prof. Kacem Chehdi, for his unwavering support, guidance, and mentorship throughout the entirety of my research. His invaluable insights and dedication have been pivotal in shaping this work. During the challenging times posed by the COVID-19 pandemic, he played a pivotal role in helping me navigate the inconveniences and disruptions, ensuring that my research progress remained on track. His support during these trying circumstances is deeply appreciated.

I would also like to extend my appreciation to the Region Bretagne and the Department of Côtes d'Armor (France) for their financial support.

High tribute shall be paid to Prof. Franck Marzani for serving as the committee chair and for his valuable patience and feedback. Sincere thanks go to committee members, Prof. Jenny Benois-Pineau and Assoc. Prof. Gabriela Ciuperca, for their thoughtful reviews and constructive comments.

I am immensely grateful to Assoc. Prof. Benoît Vozel and Assoc. Prof. Claude Cariou, for their continuous encouragement and valuable contributions to my research. Their expertise and collaborative spirit have enriched my academic experience. Special thanks also go to Mr. Josias Lefevre for his meticulous data preparation.

In addition to my academic mentors, I extend my heartfelt thanks to my parents for their unwavering belief in my capabilities and their constant encouragement. Their love and support have been my anchor throughout this journey.

I would also like to express my gratitude to my boyfriend Difei Li for his exceptional understanding, boundless patience, and dedicated care in every aspect of my life. His support provided me with the emotional strength to persevere through the challenges of academia.

Lastly, I would like to acknowledge the countless friends, colleagues, and fellow researchers who have contributed to my academic and personal growth. Your camaraderie and shared experiences have enriched my life in immeasurable ways.

To all those who have played a part, no matter how big or small, in this academic achievement, I extend my deepest thanks.



## Notation

$n$	Number of attributes of each object
$x_i$	Object characterized by $n$ attributes
$\{a_i^{(1)}, a_i^{(2)}, \dots, a_i^{(n)}\}$	Attributes of object $x_i$
$d(x_i, x_l)$	Distance between objects $x_i$ and $x_l$
$p, q$	Positive integers
$L_p$	$L_p$ norm
$L_1$	$L_1$ norm
$L_2$	$L_2$ norm
$N$	Number of objects to be partitioned
$K$	Number of classes
$\mathcal{X}$	Set of objects $\mathcal{X} = \{x_i\}_{i=1}^N$
$\mathcal{C}_j$	$j$ th class
$\mathcal{P}$	Partition containing $K$ classes $\mathcal{P} = \{\mathcal{C}_j\}_{j=1}^K$
$z_j$	Centroid of class $\mathcal{C}_j$
$J(\mathcal{P})$	Partitioning objective function
$m$	Weighting exponent
$u_{ij}$	Membership grade of object $x_i$ to class $\mathcal{C}_j$
$U$	Membership matrix
$N_j$	Number of objects in class $\mathcal{C}_j$
$\underline{D}(\mathcal{C}_j)$	Intra-class dispersion of class $\mathcal{C}_j$
$\underline{D}$	Global intra-class dispersion
$\overline{D}(\mathcal{C}_j)$	Inter-class dispersion of class $\mathcal{C}_j$
$\overline{D}$	Global inter-class dispersion
$\eta$	Coefficient of ponderation
$s(x_i, x_k)$	Similarity between objects $x_i$ and $x_k$
$S$	Similarity matrix
$pr$	Preference parameter
$r(x_i, x_k)$	Responsibility
$a(x_i, x_k)$	Availability
$\lambda$	Damping rate
$\tau$	Current iteration
$\epsilon$	User-defined radius of a class

$MinPts$	Minimum number of points within a class with a radius of $\epsilon$
$X = x_1, x_2, \dots, x_t, \dots$	Series of sequential objects of data stream
$t$	Current time
$\xi$	Learning rate
$\overline{CF2^x}$	Sum of squares of objects in a micro-cluster
$\overline{CF1^x}$	Sum of objects in a micro-cluster
$CF2^t$	Sum of squares of timestamps in a micro-cluster
$CF1^t$	Sum of timestamps in a micro-cluster
$nb_i$	Number of objects that are aggregated into just one object $x_i$
$\varepsilon_i$	Average mutual distance between $x_i$ and $nb_i$ objects
$e_i$	$i$ th exemplar
$d_i$	Sum of the squares of distances between associated objects and $e_i$
$lastEdit_i$	Last timestamp when an object is associated with $e_i$
$h$	User-specified window length
$\omega$	Weight of a micro-cluster
$\gamma$	Decay factor
$\overline{CF^1}$	Weighted linear sum of objects
$\overline{CF^2}$	Weighted squared sum of objects
$t_o$	Creation time of the outlier-micro-cluster
$r_0$	Radius of non-core region
$zS$	Centroid of dataset $\mathcal{X}$
$x, y$	Objects $x$ and $y$
$\sigma_i$	Standard deviation of class $C_i$
$\sigma_{\mathcal{X}}$	Standard deviation of dataset $\mathcal{X}$
$N_d$	Size of the data chunk
$B_t$	Data chunk arriving at time $t$
$x_i(t)$	The object $x_i$ in the data chunk $B_t$
$DS_t$	Data stream $DS_t = \cup_{j=1}^t \{B_j\}$
$K_t$	Number of classes in data chunk $B_t$
$z_i(t)$	$i$ th centroid of the $i$ th class within data chunk $B_t$
$BC_t$	Exemplar set obtained by partitioning data chunk $B_t$

$BC$	Exemplar set $BC = \cup_{j=1}^t \{BC_j\}$
$N_e$	Number of exemplars
$ES$	Set of exemplars $ES = \{e_i\}_{i=1}^{N_e}$
$SC_i$	$i$ th subclass
$N_{SC_i}$	Number of objects in subclass $SC_i$
$K_{GT}$	Number of the main ground truth classes
$N_i^j$	Number of objects in the $i$ th estimated class allocated to the $j$ th main ground truth class
$p_o$	Actual observed agreement
$p_e$	Hypothetical probability of chance agreement

# List of abbreviations

STRFCM	STReam Fuzzy <i>C</i> -Means
FCM	Fuzzy <i>C</i> -Means
FCMO	Fuzzy <i>C</i> -Means-Optimized
WFCMO	Weighted FCMO
SVM	Support Vector Machines
MLBG	Modified LBG
AP	Affinity Propagation
DBSCAN	Density-Based Spatial Clustering of Application with Noise
GDBSCAN	Generalized Density-Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points to Identify the Clustering Structure
CLIQUE	Clustering in Quest
EHCF	Exponential Histogram of Cluster Feature
WAP	Weighted AP
DP-Tree	Dependency Tree
EDDS	Enhanced Density-Based Method for Clustering Data Streams
CODAS	Clustering Online Data-streams into Arbitrary Shapes
MDSC	Multi-Density Stream Clustering
DFPS-Clustering	Dynamic Fitness Proportionate Sharing Clustering
CMC	Core Micro-Clusters
AAP	Active Affinity Propagation
GT	Ground Truth
RMSSTD	Root-Mean-Square Standard Deviation Validity Index
RS	R-Squared Validity Index
$I$	Modified Hubert $I$ Statistic Validity Index
CH	Calinski-Harabasz Validity Index
PS	Partition Separation Validity Index
DB	Davies-Bouldin Validity Index
XB	Xie-Beni Validity Index
WB- $L_1$	WB Index with the $L_1$ Norm
WB- $L_{IM}$	Modified WB- $L_1$

FCMO- $L_1$ - $F$	FCMO using the $L_1$ norm and the $F$ index
FCMO- $L_2$ - $F$	FCMO using the $L_2$ norm and the $F$ index
FCMO- $L_1$ - $D$	FCMO using the $L_1$ norm and the Dunn's index
FCMO- $L_1$ - $SIL$	FCMO using the $L_1$ norm and the Silhouette index
FCMO- $L_1$ - $WB$	FCMO using the $L_1$ norm and the $WB$ - $L_1$ index
FCMO- $L_1$ - $WBM$	FCMO using the $L_1$ norm and the $WB$ - $L_{IM}$ index
NC	Number of Estimated Classes
CPU	CPU Execution Time
OT	Number of Detected Outliers

# Chapter 1

## Introduction

In recent years, the processing of large data has attracted widespread attention, and the processing of hyperspectral data streams has aroused interest for many applications, such as environmental monitoring, target and anomaly detection, geological mapping, and vegetation survey [1]-[4].

Hyperspectral imaging sensors deployed on manned or unmanned aerial vehicles [5,6,7] have emerged as popular tools for observing territories and their evolution. These sensors enable the temporal monitoring of urban and rural green spaces, greenways, algal accumulation, invasive plants, crop diseases, roof mapping, and more [8, 9]. However, the processing capacity for handling the substantial volume of generated data has remained limited until now. Most of the data processing occurs offline, post-completion of the data acquisition mission. Presently, the challenge lies in processing and analyzing the data online, a crucial aspect for effectively interpret the content of data [5]. In this domain, there is still a deficiency in state-of-the-art techniques tailored to this type of processing, especially for hyperspectral images with expansive spatial and spectral dimensions [8]. The purpose of our research is to develop a data stream partitioning method to solve the issue of unsupervised and online learning of statistical patterns within hyperspectral data streams. The developed method can be easily applied to hyperspectral images acquired by hyperspectral sensors embedded in manned or unmanned aerial platforms.

Data stream partitioning [10] uses data or data batches that arrive continuously and assigns them to classes without the benefit of having observed the entire dataset. We specify that "partitioning" is an operation of subdivision of a dataset into homogeneous classes to form a partition.

Data stream partitioning methods can be divided into three categories: supervised, semi-supervised and unsupervised. Supervised partitioning methods deal with training samples to make partitions. However, in some real-world scenarios, obtaining labeled data can be challenging or even impossible. Semi-supervised methods do not require training samples and only necessitate prior knowledge of the number of classes, while unsupervised methods do not need any prior information. In certain practical applications, the number of classes is difficult to determine in advance and can change with incoming data, which becomes one of the main

limitations of semi-supervised methods. Therefore, unsupervised partitioning methods are better suited to partition data streams without the need for prior information.

Nowadays, most of the data stream partitioning methods are parametric requiring users to specify one or more parameters and/or the number of classes before the partitioning process. The introduction of parameters will have a more or less impact on the partitioning performance. The employed parameters often need to be tuned by users in advance in order to predetermine their values.

In practice, there are situations where prior knowledge of the dataset cannot be known in advance. Therefore, it is difficult to conduct empirical parameter tuning to determine the appropriate values of parameters to obtain the optimal partition. Furthermore, the values of these parameters are not universally applicable to all types of datasets, and when the type of dataset changes, the parameters need to be retuned. Eliminating the reliance on parameters presents a substantial challenge in data stream partitioning. To address this challenge, an unsupervised and non-parametric approach, named STRFCM (STReam Fuzzy C-Means), was developed that can automatically estimate the number of classes and partition the data stream without the need for prior information or the introduction of any parameters.

Our proposed method mainly consists of two fundamental steps. The data chunks in the stream are first partitioned by the Fuzzy C-Means-Optimized (FCMO) [11] method in order to identify classes and their exemplars in each data chunk. In this step, we optimized FCMO to align with the requirements of our approach. FCMO is an unsupervised static dataset partitioning method. It seeks to enhance the performance and efficiency of the traditional Fuzzy C-Means partitioning algorithm (FCM) [12]. The subsequent fusion step employs the proposed Weighted FCMO (WFCMO) method to partition the weighted exemplar set and obtain the final optimal partitioning results.

In contrast to parametric partitioning methods, such as CluStream [13], which requires the predefined number of classes, and STRAP [14], which necessitates the predetermination of six user-defined parameters, our proposed method performs without the need for labeled data or any other prior information. Furthermore, it does not require parameter tuning, making it adaptable to a wide range of datasets.

The proposed method was assessed on synthetic hyperspectral images and a real-world dataset. Its partitioning performance was evaluated by using two external metrics: purity and kappa index. Besides, the number of estimated classes and execution time were also considered. To demonstrate the effectiveness of our proposed method, we compared it to five parametric partitioning methods: three unsupervised methods (STRAP [14], CODAS [15] and DenStream

[16]), one semi-supervised method (CluStream [13]), and one supervised method using active learning (AAPStream [17]). The results show that our method performs better than the compared methods.

This thesis is organized as follows. Chapter 2 provides an overview of the current state of the art in the fields of static dataset and data stream partitioning. Chapter 3 introduces the proposed unsupervised and non-parametric approach in detail. Chapter 4 presents the assessment of our proposed method and gives comparative analysis with five parametric partitioning methods mentioned above. Some conclusions and perspectives for further research are presented in Chapter 5.



# Chapter 2

## State of the art

This chapter reviews and discusses the state of the art related to partition static datasets and data streams. Partitioning methods are widely used in various fields, such as environmental monitoring, sensor networks, network intrusion detection, fraud detection, fault detection, and medical diagnosis [10], [18]-[25]. These methods can be divided into three categories: supervised, semi-supervised and unsupervised partitioning methods. Each category can also be subdivided into parametric and non-parametric methods.

Supervised methods require labeled training data to learn and accomplish the partitioning task. Maximum likelihood classifier [26] and support vector machines (SVM) [27] are two commonly used approaches in supervised partitioning. Semi-supervised and unsupervised methods do not require training samples. However, semi-supervised methods need to specify the number of classes in advance, while unsupervised methods do not require the number of classes or any other prior knowledge. In the case of supervised methods, a significant limitation is their reliance on training samples, which, for certain applications, cannot be obtained. Semi-supervised and unsupervised methods do not have this limitation. Considering this constraint, we will mainly present semi-supervised and unsupervised partitioning methods in the following sections. The advantages and drawbacks of these methods are explained in detail in this chapter. In addition, the partitioning validity indices are introduced.

### 2.1 Similarity criteria

Data partitioning involves the split of a dataset of objects into different classes following a similarity criterion [28]. In essence, this process aims to group objects of the same class as closely as possible while ensuring significant separation between objects from different classes. Figure 2.1 depicts a partitioning example of objects characterized by two attributes. Each class can be represented by its centroid (legend \*), also called an exemplar.

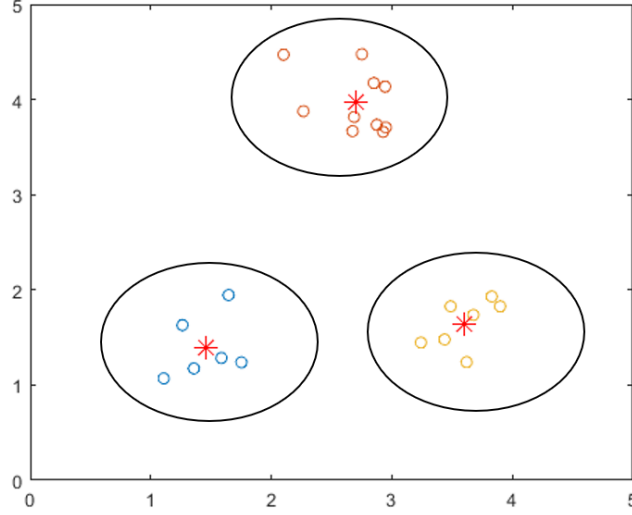


Figure 2.1: Partitioning of objects in three classes. To each class is associated its centroid or exemplar (\*).

Data partitioning methods critically rely on the choice of similarity criteria to form classes in fields such as data analysis [29], natural language processing [30], and image processing [31].

Depending on the nature of the data and the specific problem being addressed, different types of similarity measures are used. Some common similarity criteria are introduced below.

$L_p$  norm is referred as Minkowski distance [32] which is most often used for comparing numerical data. Suppose we have two objects  $x_i$  and  $x_j$ , each characterized by  $n$  attributes:  $\{a_i^{(1)}, a_i^{(2)}, \dots, a_i^{(n)}\}$  and  $\{a_j^{(1)}, a_j^{(2)}, \dots, a_j^{(n)}\}$ . The  $L_p$  norm distance between the objects  $x_i$  and  $x_j$  is computed as:

$$d(x_i, x_j) = \left( \sum_{l=1}^n |a_i^{(l)} - a_j^{(l)}|^p \right)^{1/p} \quad (2.1)$$

where  $p$  is a positive integer.

When  $p = 2$ , the  $L_2$  norm is the standard Euclidean distance.

$$d(x_i, x_j) = \left( \sum_{l=1}^n |a_i^{(l)} - a_j^{(l)}|^2 \right)^{1/2} \quad (2.2)$$

It's often employed for data partitioning and classification tasks.

When  $p = 1$ , the  $L_1$  norm measures the sum of absolute differences between two objects.

$$d(x_i, x_j) = \sum_{l=1}^n |a_i^{(l)} - a_j^{(l)}| \quad (2.3)$$

It can also be used for data partitioning and classification tasks.

Other similarity criteria such as Cosine similarity [33], Hamming distance [34], Levenshtein distance [35], and Jaccard similarity [36] can also be employed.

Different partitioning methods employ different similarity criteria according to the characteristics of objects and the specific problems being addressed. A good partitioning method will produce classes with high intra-class similarity and low inter-class similarity. The following sections will introduce the static dataset and data stream partitioning methods.

## 2.2 Static dataset partitioning methods

Static dataset partitioning methods consider all objects are available from the beginning. They process the entire dataset at once, partitioning objects into classes based on similarity measures. As explained above, we only analyze semi-supervised and unsupervised static partitioning methods, which differ based on whether the number of classes needs to be determined in advance. Depending on whether they employ user-defined parameters, they can be further divided into parametric and non-parametric methods.

### 2.2.1 Semi-supervised methods

In this section, semi-supervised static dataset partitioning methods are presented and analyzed. Assume that  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$  is a set of  $N$  objects, where each object is characterized by  $n$  attributes. The partitioning result on  $\mathcal{X}$  is denoted as  $\mathcal{P} = \{C_1, C_2, \dots, C_K\}$  which contains  $K$  classes. The centroid of class  $C_j$  is defined as  $z_j$ .

- *K-Means*

*K-Means* [37] is a popular and traditional partitioning method, aiming to partition a dataset into predetermined number of classes. During partitioning, an objective function is used for assessing the partitioning quality, which is defined as:

$$J(\mathcal{P}) = \sum_{j=1}^K \sum_{i=1}^N u_{ij} d^2(x_i, z_j) \quad (2.4)$$

where  $u_{ij} = 1$  if  $x_i \in C_j$  and 0 otherwise. It calculates the sum of Euclidean distance between each object and its corresponding class centroid. For a given  $K$ , the optimal partition is achieved when  $J(\mathcal{P})$  is minimized, which means the intra-class dispersion is the lowest at this time and the objects are close to their class centroids with the highest intra-class similarity.

$K$ -Means adopts a greedy optimization approach that begins with a random partition and iteratively moves objects from one class to another to minimize the objective function value until a local optimum is reached. However, it's important to note that the local optimum can be influenced by the initial setup of the greedy optimization process.

The  $K$ -Means algorithm is an iterative scheme and mainly has four steps:

*Input:*

- Dataset  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$
- Number of classes  $K$

- 1) Randomly select  $K$  objects from the dataset  $\mathcal{X}$  as the initial centroids of  $K$  classes.
- 2) Calculate the distance between each object in the dataset and each class centroid. Assign each object to its nearest class.
- 3) Update the class centroids to minimize the objective function  $J(\mathcal{P})$  (Equation 2.4).
- 4) Repeat steps 2 and 3 until the class centroids stabilize.

*Output:* Discovered  $K$  classes and exemplar of each class

The  $K$ -Means method is very simple to use and can be easily implemented in solving many practical problems. It works well for compact and hyper-spherical classes. In addition,  $K$ -Means is effective in partitioning large-scale datasets due to its low time and space complexity. Nevertheless, it comes with limitations. The number of classes should be specified in advance, which is sometimes impractical in real-world scenarios. It is sensitive to the initial selection of class centroids and might converge to a local optimum rather than the global optimum.

Based on the same iteration process,  $K$ -Median [38] and  $K$ -Modes [39] were proposed.  $K$ -Median employs the median of attribute values as class exemplars. It is often chosen in cases where the data could potentially include outliers or when the focus is on prioritizing robustness over computational speed.  $K$ -Modes is designed for categorical data and adopts categorical similarity criterion.

- *Fuzzy C-Means (FCM)*

While  $K$ -Means assigns each object to only one exact class, FCM [12] provides a more flexible approach that allows objects to belong to multiple classes. The primary concept is to use the membership grade to determine the degree to which each object belongs to a certain class. Hence, objects positioned at the periphery of a class with lower membership grades are more inclined to have a lesser affiliation with that class compared to objects situated closer to the class centroid.

The partitioning objective function of FCM algorithm is defined as:

$$J(\mathcal{P}) = \sum_{j=1}^K \sum_{i=1}^N u_{ij}^m d^2(x_i, z_j), \quad u_{ij} \in [0,1] \quad (2.5)$$

subject to:

$$\sum_{j=1}^K u_{ij} = 1 \quad (2.6)$$

where  $u_{ij}$  denotes the membership grade of the object  $x_i$  to the class  $C_j$ ,  $z_j$  is the centroid of class  $C_j$ ,  $d(\cdot, \cdot)$  calculates the Euclidian distance, and  $m$  is the weighting exponent that controls the fuzziness of the partition,  $m \in [1, +\infty)$ . The objective function calculates the sum of weighted squared Euclidean distance between each object in dataset  $\mathcal{X}$  and each class centroid. The main goal of FCM is to minimize  $J(\mathcal{P})$ .

FCM algorithm is an iterative scheme involving four steps [12]:

*Input:*

- Dataset  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$
- Number of classes  $K$
- Parameter defined by users:  $m$ : weighting exponent

- 1) Fix  $K$ ,  $m$  and randomly initialize the membership matrix  $U = [u_{ij}]$  ( $i \in 1, 2, \dots, N$ ,  $j \in 1, 2, \dots, K$ ). The values of  $u_{ij}$  must satisfy the constraint of Equation 2.6.
- 2) Compute class centroids using the following equation:

$$z_j = \frac{\sum_{i=1}^N u_{ij}^m x_i}{\sum_{i=1}^N u_{ij}^m}, \quad j = 1, 2, \dots, K \quad (2.7)$$

3) Update the membership matrix  $U = [u_{ij}]$ , the equation is shown below:

$$u_{ij} = \frac{\left(\frac{1}{d(x_i, z_j)}\right)^{\frac{2}{m-1}}}{\sum_{k=1}^K \left(\frac{1}{d(x_i, z_k)}\right)^{\frac{2}{m-1}}}, \quad i = 1, 2, \dots, N \quad (2.8)$$

4) Repeat steps 2 and 3 until the class centroids stabilize or the maximum number of iterations is reached.

*Output:* Discovered  $K$  classes and exemplar of each class

The weighting exponent  $m$  determines the level of partitioning fuzziness. A large value of  $m$  results in smaller membership grade values so that the partition is fuzzier. When  $m = 1$ , the membership grade  $u_{ij}$  will converge to 0 or 1, which implies a crisp partitioning. At this point, the FCM method is equivalent to the  $K$ -Means method. For most data,  $1.5 \leq m \leq 3$  gives good results [12].

FCM offers the advantage of soft partitioning by assigning objects to classes with membership grades. This property enables a single object to belong to multiple classes with varying degrees of membership. FCM algorithm also has the same drawback as  $K$ -Means algorithm, that is, it is easy to fall into the local optimum.

### 2.2.2 Unsupervised methods

Unsupervised static dataset partitioning methods do not require predetermining the number of classes which eliminates the need for prior information. Some typical unsupervised methods are presented below.

- *Modified LBG (MLBG)*

Modified version of LBG method (MLBG) [40] is an unsupervised method based on the  $K$ -Means algorithm and LBG [41] algorithm. Its primary objective is to automatically determine the optimal number of classes and achieve the optimal partition. It evaluates the intermediate partitioning results and modifies the current result by exploiting previous results. The MLBG algorithm mainly consists of five steps:

*Input:*

- Dataset  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$
- Parameter defined by users:  $\eta$ : coefficient of ponderation

- 1) Subdivide the dataset into two classes.
- 2) Select the class with the largest dispersion which will be subdivided into two subclasses.  
The intra-class dispersion is measured as follows:

$$\underline{D}(C_j) = \frac{1}{N_j} \sum_{i=1}^N u_{ij} d(x_i, z_j) \quad (2.9)$$

where  $u_{ij} = 1$  if  $x_i \in C_j$  and 0 otherwise,  $N_j$  is the number of objects in class  $C_j$ . If the partition is deemed invalid in the end, choose the class with the next largest dispersion.

- 3) Choose the initial two subclass centroids from the selected class. The first subclass centroid is defined as the centroid of the selected class, while the second centroid is defined as the object farthest from the first centroid in the selected class. If the partition is not valid in the end, select the next farthest object as the second centroid.
- 4) Use K-Means to partition the objects from  $k$  to  $k + 1$  classes.
- 5) Evaluate the partition with  $k + 1$  classes. This final step validates or rejects the partitioning results to find the optimal final partition and the number of estimated classes. The principle is to examine the evolution of the global intra-class dispersion, defined as follows:

$$\underline{D}(\mathcal{P}^k) = \frac{1}{k} \sum_{j=1}^k \underline{D}(C_j) \quad (2.10)$$

The partition with  $k + 1$  classes is considered valid if:

$$\underline{D}(\mathcal{P}^k) - \underline{D}(\mathcal{P}^{k+1}) > \eta \underline{D}(\mathcal{P}^{k+1}) \quad (2.11)$$

where  $\eta$  is a coefficient of ponderation related to the precision guaranteeing the stopping of the algorithm. If the validation criterion is satisfied, one goes back to step 2 with  $k + 1$  determined classes and attempts to create  $k + 2$  classes. Conversely, if the criterion is

not met, return to step 3 with  $k$  classes and change the selected subclass centroid. If none of the choices of subclass centroids results in a valid partition, the selected class for subdivision in step 2 needs to be changed. If none of the classes provides a valid partition, the optimal partitioning result is achieved.

*Output:* Discovered  $K$  classes and exemplar of each class

MLBG does not require specifying the number of classes in advance. However, it introduces a coefficient of ponderation  $\eta$ , the value of which may impact the quality of the partitioning results.

- *Affinity Propagation (AP)*

Affinity propagation (AP) [42] is an unsupervised partitioning method developed by Frey and Dueck. The fundamental concept of AP is to treat all objects as potential class exemplars and connect the objects to form a network (similarity matrix) which consists of similarities of pairs of objects. The message (responsibility and availability) exchanges through each edge in the network to find the most representative exemplar of each object. The responsibility  $r(x_i, x_k)$  indicates how suitable object  $x_k$  is to be the exemplar for  $x_i$ . The availability  $a(x_i, x_k)$  indicates the suitability of  $x_i$  to select  $x_k$  as its exemplar. Responsibility and availability messages are updated iteratively.

The responsibility is computed by the following equations:

$$\begin{aligned} r(x_i, x_k) &= s(x_i, x_k) - \max_{j, j \neq k} [a(x_i, x_j) + s(x_i, x_j)], \\ s(x_i, x_k) &= -d^2(x_i, x_k), \text{ for } i \neq k \end{aligned} \quad (2.12)$$

$$r(x_k, x_k) = s(x_k, x_k) - \max_{j, j \neq k} [a(x_k, x_j) + s(x_k, x_j)], \quad s(x_k, x_k) = pr \quad \forall k$$

where  $s(x_i, x_k)$  indicates the similarity between objects  $x_i$  and  $x_k$ ,  $s(x_k, x_k)$  represents the suitability of the object  $x_k$  as an exemplar, and  $pr$  is the preference parameter which can be set as the median of the input similarities or other possible values.

The availability is defined as follows:

$$a(x_i, x_k) = \min \left\{ 0, r(x_k, x_k) + \sum_{j, j \neq \{i, k\}} \max\{0, r(x_j, x_k)\} \right\}, \text{ for } i \neq k \quad (2.13)$$



$$a(x_k, x_k) = \sum_{j, j \neq k} \max\{0, r(x_j, x_k)\}$$

During partitioning, the responsibilities and availabilities are updated using the following equations.

$$r(x_i, x_k)_\tau = \lambda r(x_i, x_k)_{\tau-1} + (1 - \lambda) r(x_i, x_k)_\tau \quad (2.14)$$

$$a(x_i, x_k)_\tau = \lambda a(x_i, x_k)_{\tau-1} + (1 - \lambda) a(x_i, x_k)_\tau \quad (2.15)$$

where  $\tau$  is the current iteration, and  $\lambda$  is defined as the damping rate which is used to avoid numerical oscillations that may arise under some circumstances.

The AP algorithm follows an iterative scheme with the main steps shown below:

*Input:*

- Dataset  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$
- Parameters defined by users:
  - $\lambda$ : damping rate ( $\lambda \in ]0, 1[$ )
  - $pr$ : preference parameter

- 1) Initialize the responsibilities and availabilities using Equation 2.16, and calculate the similarity matrix  $S$  of size  $N \times N$ , where  $N$  is the number of objects.

$$r(x_i, x_k) = 0, a(x_i, x_k) = 0, \text{ for all } i, k \quad (2.16)$$

- 2) Update the responsibility matrix using Equation 2.12 and 2.14.
- 3) Update the availability matrix using Equation 2.13 and 2.15.
- 4) Combine availabilities and responsibilities for each object  $x_i$  to identify its exemplar  $x_k$  which maximizes  $[a(x_i, x_k)_\tau + r(x_i, x_k)_\tau]$ .
- 5) Repeat steps 2-4 until one of the following conditions is met: a predetermined number of iterations is reached, changes in the messages fall below a threshold, or the local decisions remain constant for a certain number of consecutive iterations.

*Output:* Discovered  $K$  classes and exemplar of each class

Figure 2.2, sourced from [42], illustrates the exchange of availability and responsibility messages between every pair of objects until three classes are formed. Each class is represented by a real object chosen as an exemplar by all other objects within the class.

The AP method eliminates the requirement for specifying the number of classes in advance. However, users are still required to provide input for two key parameters: the preference parameter (initial suitability of an object as an exemplar) and the damping rate, which aids in algorithm convergence.

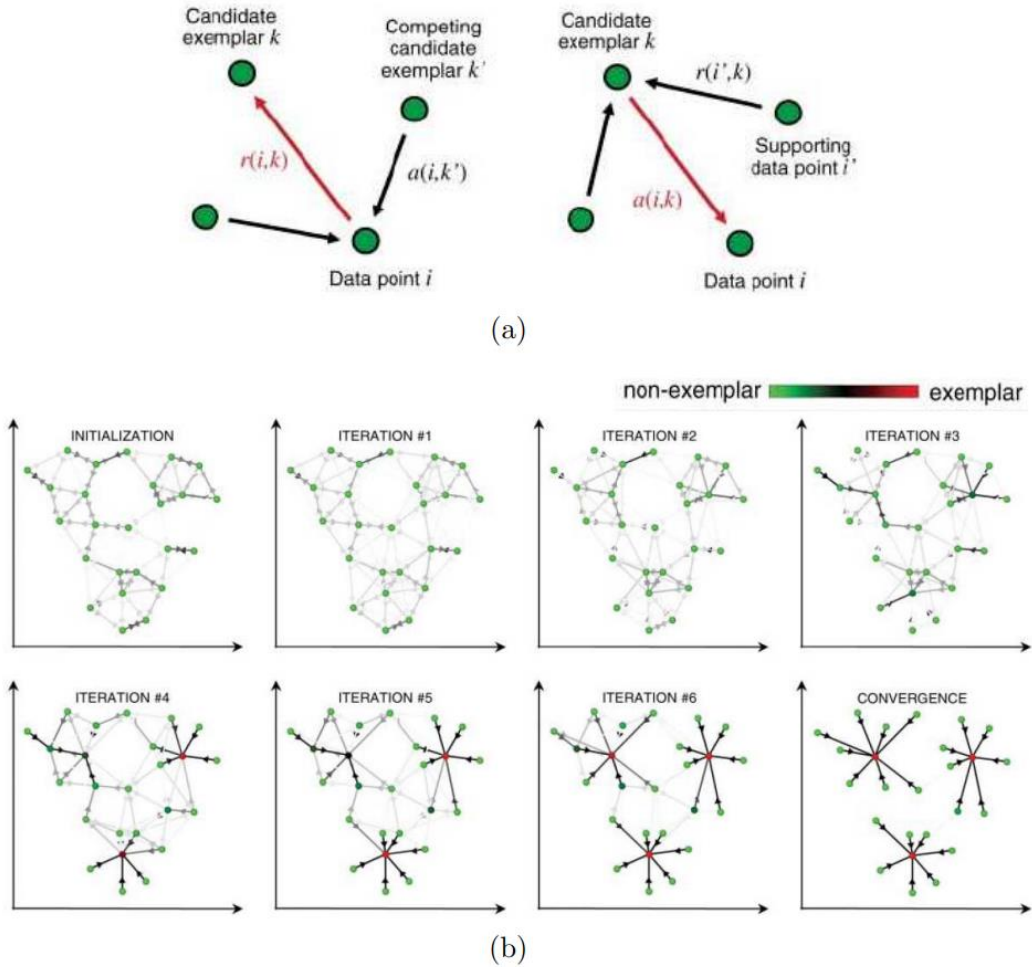


Figure 2.2: AP message passing process [42]. (a): availability and responsibility messages, (b): iterations of message passing.

LSS-AP method [43], as an extension of AP, was proposed to address the issue of the preference parameter. This method streamlines the partitioning process by consolidating closely resembling objects into a singular representative. Subsequently, it employs an automated search to identify the optimal value of the preference parameter, aiming to maximize an evaluation

criterion based on the inter-class variance, as defined by Levine and Nazif [44]. Finally, the optimal number of classes can be identified.

- *Mean Shift*

Comaniciu et al. [45] introduced Mean Shift, an iterative density-based unsupervised partitioning method that eliminates the need for specifying the number of classes in advance. In the Mean Shift algorithm, the class centroids are determined by the density of objects within a user-defined range. These centroids are then continuously updated through a density gradient ascent procedure until the class centroids stabilize. Density-based methods are particularly adept at discovering arbitrarily shaped classes. It relies on the assumption that dense regions are classes, and classes are separated by low-density regions [46].

- *DBSCAN*

DBSCAN [46] (Density-Based Spatial Clustering of Application with Noise) is a popular density-based parametric unsupervised algorithm in the field of partitioning. It defines classes as the largest collection of density-connected objects, identifying regions with sufficient density as classes. This approach enables the identification of classes with arbitrary shapes (Figure 2.3 (a)) in noisy spatial datasets. In DBSCAN, the objects are divided into three categories: core points, reachable points, and noise points. A core point must have at least *MinPts* points within its user-defined radius  $\epsilon$ . Points that are reachable from a core point within the radius  $\epsilon$  are considered density-reachable for that core point. These points can be directly connected with a series of core points to extend classes. All such points are thus density-connected (Figure 2.3 (b)). DBSCAN can automatically estimate the number of classes and identify classes with arbitrary shapes. However, its performance is sensitive to the configuration of two user-defined parameters: *MinPts* and radius  $\epsilon$ .

GDBSCAN (Generalized Density-Based Spatial Clustering of Applications with Noise) [47] method is an extension of the DBSCAN method that can adaptively adjust the radius  $\epsilon$  according to the local density. OPTICS (Ordering Points to Identify the Clustering Structure) [48] was proposed to improve DBSCAN algorithm to reduce the sensitivity of the user-defined radius. It reveals the underlying density relationships among objects by generating an "ordering" of objects according to their reachability distance. However, it is necessary to predetermine the input parameter *MinPts*.

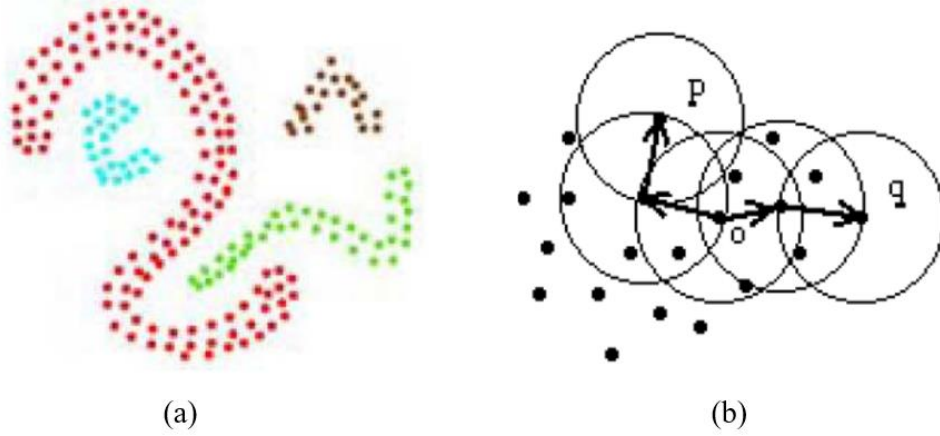


Figure 2.3: DBSCAN [46]. (a): arbitrary-shaped classes, (b): point  $p$  and point  $q$  are density-connected.

- *WaveCluster*

Grid-based partitioning methods partition data in a grid structure. Typically, they create a regular grid that divides the data space into cells (Figure 2.4) and perform the required operations within the quantization space. As illustrated in Figure 2.4, each attribute's domain is divided into segments. A cell emerges from the conjunction of these segmented attributes. Each object is assigned to a cell. Subsequently, the grid cells are partitioned based on their density.

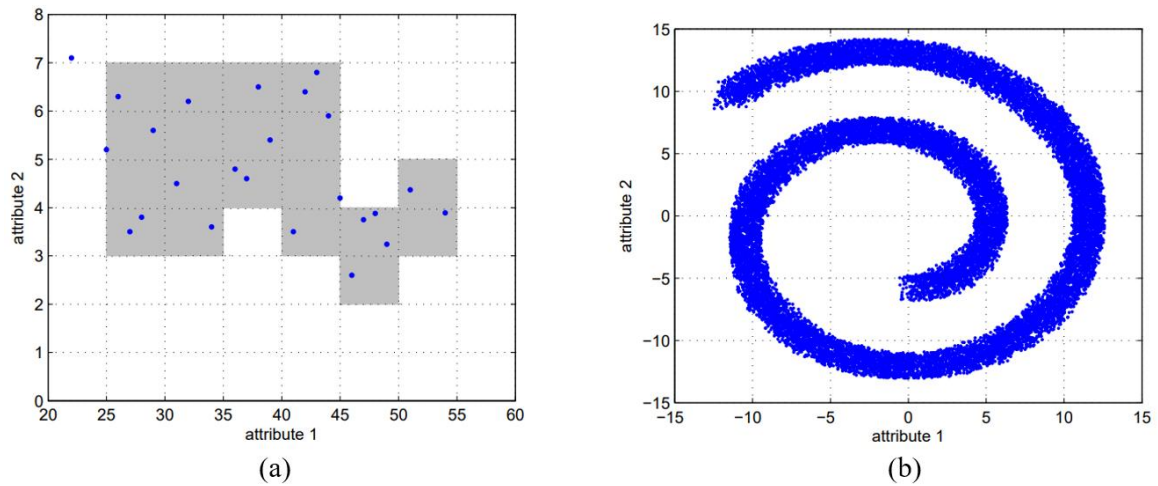


Figure 2.4: Grid-based partitioning: imposing grids on data space [49].

The main advantage of the grid-based methods lies in their fast processing time, which depends on the number of cells in each dimension within the quantization space. Additionally, they are capable of identifying classes with arbitrary shapes. However, it's important to note that grid-based methods necessitate the predefinition of the grid size [10].

WaveCluster [50] is a grid-based unsupervised partitioning method which employs a wavelet-based approach to partition the data space into cells of varying sizes. It establishes a two-dimensional grid over the dataset and represents the data points in each cell by the number of points [51]. Therefore, objects are represented in a grey-scale format, akin to an image. This leads to the redefinition of the problem of identifying classes as an image segmentation problem, where wavelets are used for smoothing and multi-scaling purposes.

The WaveCluster algorithm mainly consists of four steps:

*Input:*

- Dataset  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$
- Wavelet function
- Parameter defined by users: grid size

- 1) Quantize space to create a data grid and allocate each data point to a cell.
- 2) Apply wavelet transformation to the cells.
- 3) Use the average of all sub-images to find connected classes.
- 4) Map the identified classes back to the points in the original space.

*Output:* Discovered  $K$  classes

WaveCluster exhibits the capability to partition classes with arbitrary shapes, eliminating the requirement of predefining the number of classes. However, it is sensitive to the choice of wavelet functions and the grid size defined by users.

- *CLIQUE*

Agrawal et al. [52] proposed CLIQUE (Clustering in Quest), a grid and density based partitioning method designed to partition high-dimensional datasets. It can automatically find low-dimensional subspaces of high-dimensional data space and identify dense classes in these subspaces. However, the partitioning accuracy might suffer from the initial segmentation of the data space.

- *Fuzzy C-Means-Optimized (FCMO)*

FCMO [11] is an unsupervised, non-parametric extension of the FCM algorithm [12]. One of its notable features is the capability to automatically estimate the optimal number of classes, effectively eliminating the influence of random selection for initial class centroids encountered

in FCM. Due to its independence from the need for pre-determining the number of classes and parameter tuning, we incorporate it into our proposed method. A detailed introduction of FCMO will be presented in Subsection 3.2.1.

## 2.3 Data stream partitioning methods

Different from the static dataset partitioning using the entire obtained dataset, as described in the previous section, the data stream partitioning uses the data or data chunks that arrive continuously and assigns the arrived data to classes without the benefit of having observed the entire dataset.

In the state-of-the-art, many methods have been proposed for data stream partitioning as presented in [53]-[56]. This section provides a literature review on the state-of-the-art semi-supervised and unsupervised methods. Semi-supervised methods require the prior information about the number of classes in the data stream, while unsupervised methods do not need any prior information. According to their need for input parameters, they can be further divided into parametric methods and non-parametric methods. The state-of-the-art data stream partitioning methods are introduced in the following subsections.

### 2.3.1 Semi-supervised methods

Assume that a data stream consists of a series of sequential objects  $X = x_1, x_2, \dots, x_t, \dots$ , where  $x_t$  is the object arriving at time  $t$ .

- *Online K-Means*

Barbakh et al. [57] proposed Online  $K$ -Means method extending classic  $K$ -Means to the case of online data partitioning where the number of classes must be specified in advance. The partitioning objective function  $J(\mathcal{P})$  for  $K$ -Means has been shown in Equation 2.4. The online  $K$ -Means algorithm mainly has two steps:

*Input:*

- A series of sequential objects  $X = x_1, x_2, \dots, x_t, \dots$
- Number of classes  $K$
- Parameter defined by users:  $\xi$ : learning rate

- 1) Randomly initialize the class centroids  $z_1, z_2, \dots, z_K$ .
- 2) Sequentially input object  $x_t$  and find its closest class centroid  $z_j$ . Then update  $z_j$  as

$$z_j^{(new)} = z_j + \xi(x_t - z_j) \quad (2.17)$$

where  $\xi$  is a learning rate usually set to be a small positive value.

*Output:* Discovered  $K$  classes and exemplar of each class

The Online  $K$ -Means algorithm has the same advantages and drawbacks with the  $K$ -Means algorithm.

- *STREAM*

STREAM [58] is one of the earliest data stream partitioning methods. Its primary concept is to divide the data stream into a series of data batches with fixed batch size, then employ  $K$ -Median to partition the data batches and identify classes in each batch. Before partitioning, it's necessary to specify the number of classes.

- *CluStream*

Aggarwal et al. [13] proposed a method based on user-specified, online clustering queries, called CluStream. It divides the partitioning process into two phases. The online phase employs the concept of micro-clusters to periodically store detailed information (summaries) of the stream, and the offline phase uses  $K$ -Means to partition the stored summaries in a user-defined time horizon to identify macro-clusters.

A micro-cluster for a set of  $n$ -dimensional objects  $x_1, x_2, \dots, x_{N_i}$  with timestamps  $t_1, t_2, \dots, t_{N_i}$  is defined as the  $(2 \cdot n + 3)$  tuple  $(\overline{CF2^x}, \overline{CF1^x}, CF2^t, CF1^t, N_i)$ , wherein  $\overline{CF2^x}$  and  $\overline{CF1^x}$  each correspond to a vector of  $n$  entries [13]. The definitions of these entries are as follows:

- $\overline{CF2^x}$  contains the sum of squares of objects (i.e.,  $\sum_{j=1}^{N_i} x_j^2$ ).
- $\overline{CF1^x}$  contains the sum of objects (i.e.,  $\sum_{j=1}^{N_i} x_j$ ).
- $CF2^t$  contains the sum of squares of timestamps  $t_1, t_2, \dots, t_{N_i}$ .
- $CF1^t$  contains the sum of timestamps  $t_1, t_2, \dots, t_{N_i}$ .
- $N_i$  is the number of objects in the  $i$ th micro-cluster.

Figure 2.5 depicts the diagram of the CluStream algorithm to show how it works. CluStream first initializes the micro-clusters by employing  $K$ -Means on an initial batch that contains the first specific number of objects in the stream. During the online phase, when a new object

arrives, it is partitioned into its closest micro-cluster if the distance between the object and the centroid of that micro-cluster is less than the threshold. If the distance exceeds the threshold, a new micro-cluster will be created. Since the number of micro-clusters is fixed, it is necessary to reduce the old micro-clusters by one. This can be achieved by either deleting an old micro-cluster or merging two of the old micro-clusters. During the offline phase, CluStream employs a modified  $K$ -Means algorithm to partition the identified micro-clusters within the user-specified time horizon to obtain  $K$  macro-clusters.

While CluStream is capable of detecting data stream evolution, it introduces several parameters that require pre-tuning, which can impact partitioning performance. Additionally, it necessitates specifying the number of micro-clusters and macro-clusters in advance.

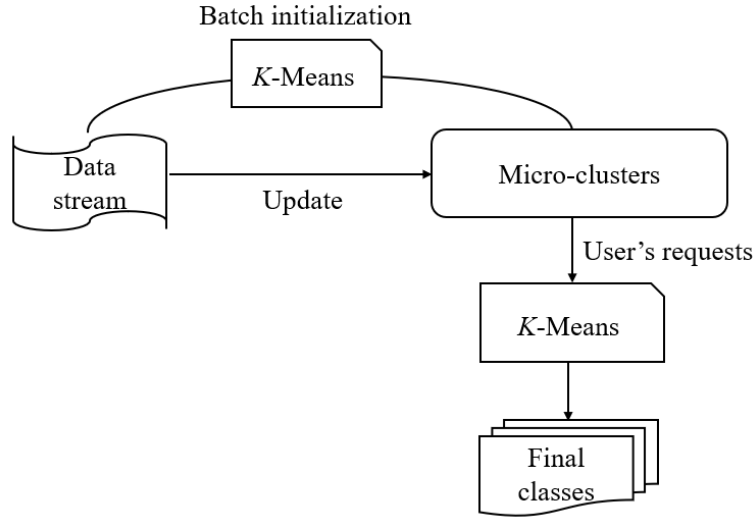


Figure 2.5: Diagram of CluStream algorithm.

- *StreamKM++*

StreamKM++ [59] extends the principles of  $K$ -Means++ [60] to the realm of data stream partitioning.  $K$ -Means++ leverages a randomized seeding technique to solve the issue of initializing centroids in  $K$ -Means partitioning. In the case of StreamKM++, the algorithm creates a small weighted set of objects chosen from the data stream, referred to as a coreset. To speed up the time for the non-uniform sampling during coreset construction, a novel data structure named the coreset tree is introduced. After deriving the coreset from the data stream, a weighted  $K$ -Means algorithm is applied to this coreset to identify the final classes. It's important to note that this method requires predefining the number of classes and the coreset size, both of which can significantly influence partitioning performance.



- *SWClustering*

SWClustering [61] introduced a novel data structure known as the Exponential Histogram of Cluster Feature (EHCF), which is applied for sliding window partitioning. EHCF serves the purpose of analyzing the evolution of classes. The partitioning process using the *K*-Means algorithm is performed on the EHCFs within the sliding window, and the expired objects are subsequently removed. The number of expired objects is controlled by a user-defined error parameter. The memory consumption of SWClustering is constrained by a predefined maximal number of EHCFs. Additionally, users need to define the sliding window size.

### 2.3.2 Unsupervised methods

In contrast to semi-supervised methods, unsupervised methods do not necessitate prior knowledge of the number of classes.

- *STRAP*

STRAP [14] is an unsupervised partitioning method based on AP [42]. It employs the weighted AP algorithm (WAP) to partition the data stream and a reservoir to store outliers. Besides, it introduces two methods for detecting changes in the data stream and starting the repartition of the data stream.

The WAP algorithm is an extension of AP. Its fundamental concept involves aggregating the similar objects and representing them with a single object, which serves as the exemplar for these similar objects. The main distinction between WAP and AP lies in the introduction of a novel method for calculating the similarity matrix, while the remaining components of WAP remain consistent with the AP algorithm. Assume that there are  $nb_i$  objects which are aggregated into just one object  $x_i$ ,  $\varepsilon_i$  is the average mutual distance between  $x_i$  and  $nb_i$  objects, and  $x_j$  is another object in the data stream. The similarity between objects  $x_i$  and  $x_j$  can be defined as:

$$s'(x_i, x_j) = \begin{cases} nb_i s(x_i, x_j) & \text{if } i \neq j \\ pr + (nb_i - 1)\varepsilon_i & \text{otherwise, with } \varepsilon_i \geq 0 \end{cases} \quad (2.18)$$

where  $pr$  is the preference parameter.

Figure 2.6 illustrates the flow chart of the STRAP algorithm.

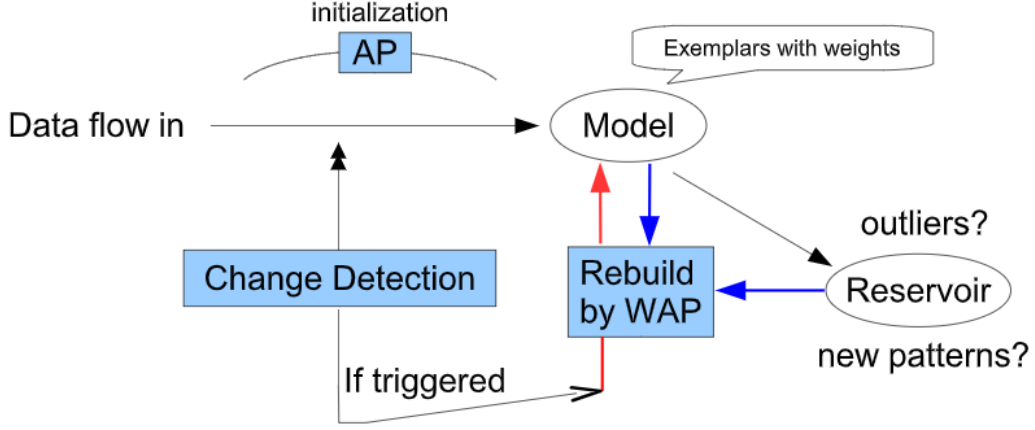


Figure 2.6: Diagram of STRAP algorithm [14].

The STRAP algorithm primarily consists of four main steps:

*Input:*

- A series of sequential objects  $X = x_1, x_2, \dots, x_t, \dots$
- Parameters defined by users: initial batch size, threshold distance, max cache (reservoir size), sliding window size, and preference parameter and damping rate of AP.

- 1) Initialize model based on the AP algorithm. Employ the AP algorithm to partition the initial batch of the input data stream, with the batch size defined by users. This process generates the initial data stream model and identifies the first exemplars of the classes. The model consists of a set of 4-tuple  $(e_i, nb_i, d_i, lastEdit_i)$  which are stored in the memory. Here,  $e_i$  represents the  $i$ th exemplar in the model,  $nb_i$  is the number of objects associated with exemplar  $e_i$ ,  $d_i$  is the sum of the squares of distances between associated objects and  $e_i$ , and  $lastEdit_i$  denotes the last timestamp when an object is associated with  $e_i$ .
- 2) Update AP-based model. The input object  $x_t$  is assigned to its closest class, and the model is updated if the distance between them is below the threshold distance; otherwise, it is stored in the reservoir. Additionally, a user-specified window length, denoted as  $h$ , is introduced. Any exemplar that hasn't been visited for a time period of  $h$  will be forgotten. The process of updating the model is as follows:

$$nb_i = nb_i \times \left( \frac{h}{h + (t - lastEdit_i)} + \frac{1}{nb_i + 1} \right) \quad (2.19)$$

$$d_i = d_i \times \frac{h}{h + (t - \text{lastEdit}_i)} + \frac{nb_i}{nb_i + 1} d^2(x_t, e_i)$$

$$\text{lastEdit}_i = t$$

where  $t$  is the current time.

- 3) Detect changes in the data stream. When certain conditions are reached, the model will be rebuilt on the basis of the existing model and objects in the reservoir to find new classes.
- 4) Rebuild the AP-based model. The WAP algorithm is employed to partition the objects in the current model and in the reservoir to generate new exemplars. Similarities between exemplars, between exemplars and objects, and between objects are defined as:

$$\begin{aligned} s(e_i, e_i) &= pr + d_i \\ s(e_i, e_j) &= -nb_i d^2(e_i, e_j) \\ s(e_i, x_j) &= -nb_i d^2(e_i, x_j) \\ s(x_j, e_i) &= -d^2(x_j, e_i) \\ s(x_j, x_j) &= pr \end{aligned} \tag{2.20}$$

*Output:* Discovered  $K$  classes and exemplar of each class

STRAP possesses the capability to detect the evolution of the data stream and can effectively handle high-dimensional data, albeit with an increase in time complexity. To achieve optimal partitioning performance, it is necessary to tune and determine the appropriate values of the initial batch size, threshold distance, max cache (reservoir size), sliding window size, along with the preference parameter and damping rate of AP.

- *DenStream*

Cao et al. [16] introduced the DenStream method, a density-based parametric unsupervised method using the DBSCAN algorithm [46]. Similar to CluStream, DenStream comprises both online and offline phases.

During its online phase, two types of micro-clusters are created, namely potential and outlier micro-clusters, to handle the noise and outliers in the data stream. Each micro-cluster is assigned a weight that exponentially decreases over time. If the weight of a micro-cluster exceeds a specified threshold, it is categorized as a potential-micro-cluster; otherwise, it is classified as an outlier-micro-cluster.

A potential-micro-cluster for a set of objects  $x_1, x_2, \dots, x_{N_i}$  with timestamps  $t_1, t_2, \dots, t_{N_i}$  is defined as  $(\overline{CF^1}, \overline{CF^2}, \omega)$ . The definitions of these entries are shown as follows:

- $\omega$  is the weight of a micro-cluster (i.e.,  $\sum_{j=1}^{N_i} 2^{-\gamma(t-t_j)}$ , where  $\gamma$  is the decay factor, and  $t$  is the current time). When  $\omega$  is greater than a user-defined threshold, the micro-cluster is defined as the potential-micro-cluster.
- $\overline{CF^1}$  is the weighted linear sum of objects (i.e.,  $\sum_{j=1}^{N_i} 2^{-\gamma(t-t_j)} x_j$ ).
- $\overline{CF^2}$  is the weighted squared sum of objects (i.e.,  $\sum_{j=1}^{N_i} 2^{-\gamma(t-t_j)} x_j^2$ ).

The centroid of the potential-micro-cluster is  $z = \frac{\overline{CF^1}}{\omega}$ .

An outlier-micro-cluster is defined as  $(\overline{CF^1}, \overline{CF^2}, \omega, t_o)$ . The definitions of  $\overline{CF^1}$ ,  $\overline{CF^2}$ ,  $\omega$  and centroid are the same as the potential-micro-cluster.  $t_o = t_1$  denotes the creation time of the outlier-micro-cluster. The value of  $\omega$  is less than the user-defined threshold.

The arriving data will be assigned to its nearest micro-cluster if their distance is equal to or less than the threshold radius; otherwise, it forms a new outlier-micro-cluster. Outdated outlier-micro-clusters will be removed.

In the offline phase, the DBSCAN algorithm is applied to partition the potential-micro-clusters. DenStream requires users to predefine four parameters, each of which can impact the final results.

SDStream [62] is a density-based unsupervised method, which serves as a modified version of DenStream, specifically designed for sliding windows. Similar to SWClustering, SDStream employs the Exponential Histogram of Cluster Feature (EHCF). It has three user-defined parameters that need to be predetermined.

rDenStream [63] adds an extra retrospect phase to the original two phases of DenStream. This retrospective phase provides an opportunity to put discarded objects back into the partitioning process, which can improve the partitioning performance.

To accommodate high-dimensional data streams, extensions of DenStream have been developed, including HDDStream [64] and PreDeConStream [65]. These extensions are designed to handle the challenges posed by high-dimensional data streams.

- *MuDi-Stream*

Amini et al. [66] proposed a density-grid based unsupervised partitioning method, namely MuDi-Stream. Like DenStream, it comprises both online and offline phases. During the online

phase, a hybrid approach that combines grid-based and micro-class-clustering techniques is used to capture summaries of the objects. During the offline phase, a density-based partitioning algorithm, namely M-DBSCAN, is introduced. This algorithm partitions the summaries to form final classes with varying densities. MuDi-Stream effectively handles outliers and reduces merging time due to its grid-based methodology. However, it may not be well-suited for high-dimensional data [66] and necessitates the pre-tuning of five parameters.

- *EDMStream*

EDMStream [67] is a density-based unsupervised partitioning algorithm, which leverages the concept of the Evolution of Density Mountain. The density mountain serves as an abstraction of the data distribution, with its changes reflecting the evolution of the data distribution. The evolution of classes is monitored by tracking variations in these density mountains. To enable real-time online partitioning, EDMStream utilizes a novel data structure known as Dependency Tree (DP-Tree) and employs filtering schemes to facilitate the real-time update of density mountains. For EDMStream, three parameters need to be determined before partitioning.

- *EDDS*

EDDS (Enhanced Density-Based Method for Clustering Data Streams) [68] is proposed to identify classes with arbitrary shapes and detect outliers. It modified the DBSCAN algorithm by summarizing each class with a set of surface-core points. EDDS employs the density-reachable concept from DBSCAN as its merging strategy, along with a heuristic solution to prune core points to keep only the surface-core points for the class. Furthermore, the algorithm employs a fading function to eliminate aged core points and outliers. However, EDDS necessitates the predetermination of four parameters.

- *CODAS*

Hyde et al. [15] suggested CODAS (Clustering Online Data-streams into Arbitrary Shapes), a density-based unsupervised partitioning method capable of partitioning data streams into arbitrarily shaped classes. It utilizes a user-defined local density to create core micro-clusters and non-core micro-clusters, with each micro-cluster comprising a non-core region of radius  $r_0$  and a core region of radius  $0.5r_0$  (Figure 2.7 (a)). Global clusters are formed when the core region of one micro-cluster intersects with the non-core region of another (Figure 2.7 (b)).

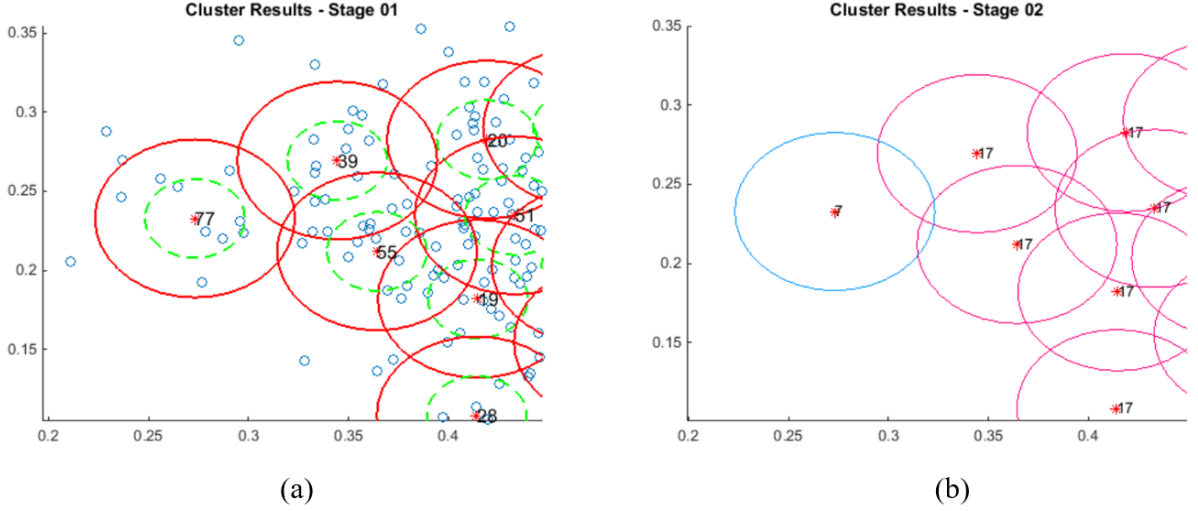


Figure 2.7: Illustration of micro-cluster regions showing (a): micro-cluster radius in red and micro-cluster core radius in green, (b): micro-clusters combined to the global clusters [15].

The CODAS algorithm mainly consists of two steps:

*Input:*

- A series of sequential objects  $X = x_1, x_2, \dots, x_t, \dots$
- Parameters defined by users:
  - $r_0$ : threshold radius
  - Threshold density

- 1) Assign the incoming object. If the space is empty or the distance between the input object and its closest micro-cluster is greater than  $r_0$ , the object will form a new non-core micro-cluster. Otherwise, the input object is assigned to its closest micro-cluster. If the distance is less than  $r_0$  but greater than  $0.5r_0$ , the micro-cluster centroid is updated using the following equation:

$$z_i = \frac{(N_i - 1)z_i + x_t}{N_i} \quad (2.21)$$

where  $z_i$  represents the centroid of the  $i$ th micro-cluster and  $N_i$  denotes the number of objects in the  $i$ th micro-cluster.

- 2) Update global clusters. If the distance between two micro-cluster centroids is less than  $1.5r_0$ , these two micro-clusters are intersected. All intersecting micro-clusters, whose density is greater than the user-defined local density, are merged to form global clusters.

*Output:* Discovered  $K$  classes and exemplar of each class

CODAS is able to partition data streams into arbitrarily shaped classes. However, it has a drawback as it necessitates parameter tuning for both the local density and the threshold radius of micro-clusters. The selection of parameter values can significantly impact CODAS's partitioning performance.

- *MDSC*

Fahy et al. [69] suggested the Multi-Density Stream Clustering (MDSC) method designed to partition data streams with varying density classes. It addresses two main challenges: multi-density class identification and dynamic stream change detection. MDSC utilizes micro-clusters and an outlier buffer to identify classes. Each micro-cluster represents a multi-dimensional sphere with a centroid and a radius. Micro-clusters have a maximum radius which is adaptive and local to each micro-cluster. If an incoming object falls within a micro-cluster's area, it is assigned to that micro-cluster. The set of micro-clusters forms macro-clusters with arbitrary shapes. MDSC introduces an ant-inspired swarm intelligence approach to discover new micro-clusters in the buffer and adopts the concept of density reachability, allowing the merging of two micro-clusters if they are density-reachable. MDSC does not require the specification of the number of classes in advance, but it needs to tune four input parameters.

- *DFPS-Clustering*

Yan et al. [70] introduced the Dynamic Fitness Proportionate Sharing Clustering (DFPS-Clustering) algorithm, a density-based parametric unsupervised approach. This method extends the Gaussian Kernel function and introduces a novel recursive lower bound for the Gaussian Kernel function to capture the data stream evolution. Its fundamental principle is to treat the fitness of each object as a density value, reflecting its attractiveness to neighboring objects. It employs the fitness proportionate sharing strategy to search and determine the class centroids. However, four parameters need to be specified in advance by users to initiate the partitioning process.

- *D-Stream*

D-Stream [71] is a grid-based unsupervised partitioning method comprising two distinct phases. During the online phase, D-Stream maps input objects into grid cells. During the offline phase, it partitions these grid cells based on a computation of the cell density. Here, the density of a cell is determined by the number of objects it encompasses. A grid class is a group of connected grid cells with a higher density than the surrounding grid cells. D-Stream utilizes a density decaying technique to capture the data stream evolution and discard the outdated data, which

introduces a user-specified decay factor. Notably, D-Stream does not require the user to predefine the number of classes, but it necessitates the determination of four parameters.

Amini et al. [72] extended D-Stream by adopting the concept of sliding window and introduced the grid-based partitioning method known as DENGRIIS, which only focuses on the most recent data with user-specified window size.

Jia et al. [73] proposed a grid-based unsupervised method called DD-stream, which shares a similar partitioning procedure with D-Stream. However, DD-Stream takes sporadic grids into account in its cluster analysis which improves the overall partitioning quality. It is important to note that DD-Stream is a parametric method, necessitating the predetermined values of parameters.

- *MR-Stream*

MR-Stream [74] is a grid-based unsupervised method that utilizes a tree-like data structure and defines that any grid cell can be further subdivided into more subcells. Each cell corresponds to a tree node. The maximum tree height is predefined by users. The weighted subcells are partitioned offline by specifying a user-defined height. A fading function is introduced to decay cell weights. MR-Stream offers a finer grid spacing and enables the discovery of classes at multiple resolutions. However, this method requires numerous parameters to be predefined, which is impossible without knowledge of the data stream.

- *CEDGM*

Tareq et al. [75] suggested a density grid-based unsupervised method called CEDGM to improve the partitioning quality. It defines the core micro-clusters (CMCs) as the primary entities to construct the cluster and combines the intersected CMCs into macro-clusters. The partitioning process is performed on the grids. The CMC radius needs to be specified in advance. Before partitioning, a tuning process is required to determine five parameters.

- *AAPStream*

In addition to the semi-supervised and unsupervised methods introduced above, we also analyze a supervised method whose performance will be compared with our proposed method in Chapter 4.

AAPStream [17] is an active partitioning method based on AP. It employs the active affinity propagation method (AAP) for partitioning data stream with available labeled data (prior knowledge), making it a supervised partitioning approach to improve partitioning performance. AAPStream utilizes active learning to train a classification model from labeled data to identify



the most representative and informative data for supervision, which will be applied to the AP method. It mainly has three steps.

*Input:*

- A series of sequential objects  $X = x_1, x_2, \dots, x_t, \dots$
  - Parameters defined by users: initial batch size, decay factor, buffer size, and preference parameter and damping rate of AP.
- 1) Identify the most informative and representative exemplars in the initial batch of objects by using AAP to create the stream model.
  - 2) Assign incoming objects. If the distance between the incoming object and its nearest class is less than a heuristic threshold, the object is assigned to this class; otherwise, it is placed in the buffer.
  - 3) Update the stream model. When the buffer is full, the stream model is updated using AAP, and the outdated objects are deleted.

*Output:* Discovered  $K$  classes and exemplar of each class

AAPStream improves partitioning performance by leveraging labeled data. However, labeled data may not be available in some circumstances. In addition, it needs to tune five parameters (initial batch size, decay factor, buffer size, damping rate and preference parameter) in advance to achieve optimal performance.

In conclusion, the state-of-the-art data stream partitioning methods typically employ many parameters or necessitate the predetermination of number of classes, such as the methods presented above. However, the utilization of parameters can introduce a level of uncertainty into the method's partitioning performance. The limitations of parametric methods include:

- 1) The choice of input parameter values can have a more or less impact on partitioning performance. Different parameter settings can lead to varying partitioning results.
- 2) Empirical parameter tuning process is necessary to find appropriate parameter values in order to achieve high partitioning performance. In the case of partitioning methods with multiple parameters, the tuning process can become more complex and time-consuming.
- 3) Finding appropriate parameter values is particularly challenging for parametric methods,

when it comes to partitioning unknown and novel sets of objects, due to the lack of information about datasets.

Therefore, we aim to develop an unsupervised method capable of overcoming the limitations imposed by parameters on the performance of partitioning methods, thereby achieving broad applicability.

## 2.4 Partitioning validity indices

The typical procedure of partitioning analysis comprises four basic steps: feature selection or extraction, partitioning algorithm design, partitioning validation, and results interpretation [76, 77]. Following the application of the partitioning algorithm, it becomes imperative to assess the validity of the partitioning results.

Partitioning validation consists of two primary categories: external and internal partitioning validation [78, 79]. The main difference between these two types of validations lies in their reliance on external information. External validation relies on external information which has been known in advance (ground truth -GT-, expert knowledge, etc.), whereas internal validation is solely dependent on the data.

Since external validation criteria rely on prior information, they are mainly utilized for selecting an optimal partitioning algorithm for a specific dataset. Additionally, these criteria can be employed to validate algorithm performance on synthetic data or data accompanied by reliable GT annotations. The classification rate is one of the most significant and credible criteria. However, in many circumstances, obtaining prior information can be challenging.

Internal validation criteria can be used to choose the optimal partitioning algorithm as well as the optimal number of classes without using prior information. These criteria are more relevant for unsupervised partitioning methods. In the state-of-the-art, many partitioning validity indices are proposed as internal validation criteria to evaluate the partitioning validity. Table 2.1 lists some common partitioning validity indices.

Table 2.1: Common partitioning validity indices.

Partitioning validity index	Formulation	Optimal value	Ref.
Root-mean-square standard deviation	$RMSSTD = \left( \frac{\sum_{i=1}^K \sum_{x \in C_i} d^2(x, z_i)}{n \sum_{i=1}^K (N_i - 1)} \right)^{\frac{1}{2}}$	Min	[80]
<i>R</i> -squared	$RS = \frac{\sum_{x \in \mathcal{X}} d^2(x, zS) - \sum_{i=1}^K \sum_{x \in C_i} d^2(x, z_i)}{\sum_{x \in \mathcal{X}} d^2(x, zS)}$	Max	[81]
Modified Hubert <i>F</i> statistic	$\Gamma = \frac{2}{N(N-1)} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{X}} d(x, y) d_{x \in C_i, y \in C_j}(z_i, z_j)$	Max	[82]
<i>WB</i>	$WB = \frac{K \sum_{i=1}^K \sum_{x \in C_i} d^2(x, z_i)}{\sum_{i=1}^K N_i d^2(z_i, zS)}$	Min	[83]
Calinski-Harabasz	$CH = \frac{\sum_{i=1}^K N_i d^2(z_i, zS) / (K - 1)}{\sum_{i=1}^K \sum_{x \in C_i} d^2(x, z_i) / (N - K)}$	Max	[84]
<i>I</i>	$I = \left( \frac{1}{K} \frac{\sum_{x \in \mathcal{X}} d(x, zS)}{\sum_{i=1}^K \sum_{x \in C_i} d(x, z_i)} \max_{i \neq j} d(z_i, z_j) \right)^q$	Max	[85]
<i>F</i>	$F = (\bar{D} - \underline{D}) / 2$	Max	[11]
<i>PBM</i>	$PBM = \left( \frac{1}{K} \frac{\sum_{x \in \mathcal{X}} d(x, zS)}{\sum_{i=1}^K \sum_{x \in C_i} d(x, z_i)} \max_{i \neq j} d(z_i, z_j) \right)^2$	Max	[86], [87]
Dunn's index	$DUNN = \min_i \left\{ \min_j \left( \frac{\min_{x \in C_i, y \in C_j} d(x, y)}{\max_k \left( \max_{x, y \in C_k} d(x, y) \right)} \right) \right\}$ $SIL = \frac{1}{N} \sum_{i=1}^K \left( \frac{1}{N_i} \sum_{x \in C_i} \frac{b(x) - a(x)}{\max(b(x), a(x))} \right)$	Max	[88]
Silhouette	$a(x) = \frac{1}{N_i - 1} \sum_{y \in C_i, y \neq x} d(x, y)$ $b(x) = \min_{j, i \neq j} \left( \frac{1}{N_j} \sum_{y \in C_j} d(x, y) \right)$	Max	[89]
Partition Separation	$PS = \sum_{i=1}^K \left\{ \frac{N_i}{\max_j N_j} - \exp \left( - \frac{\min_{i \neq j} d^2(z_i, z_j)}{\frac{1}{K} \sum_{l=1}^K \ z_l - \frac{1}{K} \sum_{k=1}^K z_k\ ^2} \right) \right\}$	Max	[90], [91]
Davies-Bouldin	$DB = \frac{1}{K} \sum_{i=1}^K \max_{j, j \neq i} \left( \frac{\frac{1}{N_i} \sum_{x \in C_i} d(x, z_i) + \frac{1}{N_j} \sum_{y \in C_j} d(y, z_j)}{d(z_i, z_j)} \right)$	Min	[92]

$$\text{Xie-Beni} \quad XB = \frac{\sum_{i=1}^K \sum_{x \in C_i} d^2(x, z_i)}{N \cdot \min_{i \neq j} d^2(z_i, z_j)} \quad \text{Min} \quad [93]$$

$$\begin{aligned} SD &= Scat(K) + Dis(K) \\ Scat(K) &= \frac{1}{K} \sum_{i=1}^K \sigma_i / \sigma_x \\ Dis(K) &= \frac{\max_{i \neq j} d(z_i, z_j)}{\min_{i \neq j} d(z_i, z_j)} \sum_{i=1}^K \left( \sum_{j=1}^K d(z_i, z_j) \right)^{-1} \end{aligned} \quad \text{Min} \quad [94]$$

---

$\mathcal{X}$  represents the dataset with  $N$  objects,  $zs$  is the centroid of  $\mathcal{X}$ ,  $n$  represents the number of attributes,  $K$  represents the number of classes,  $C_i$  represents the  $i$ th class,  $N_i$  represents the number of objects in  $C_i$ ,  $z_i$  represents the centroid of  $C_i$ ,  $d(x, y)$  represents the Euclidean distance between objects  $x$  and  $y$ ,  $q$  is a positive integer,  $\overline{D}$  and  $\underline{D}$  represents the global inter-class and intra-class dispersion, respectively,  $\sigma_i$  represents the standard deviation of class  $C_i$ , and  $\sigma_{\mathcal{X}}$  represents the standard deviation of dataset  $\mathcal{X}$ .

---

Root-mean-square standard deviation (*RMSSTD*) is a statistical measure to calculate the dispersion within classes. It computes the square root of the sum of object variance of all attributes. It considers only the compactness within the class. A lower *RMSSTD* indicates better partitioning quality. It implies that objects within each class are closer to their class centroid, resulting in more compact and well-defined classes. *R*-squared (*RS*) represents the ratio of the sum of squares between classes to the total sum of squares of the entire dataset. It measures the degree of dissimilarity between classes. A higher value of *RS* signifies a better partitioning result. The modified Hubert  $\Gamma$  statistic ( $\Gamma$ ) assesses the dissimilarity between classes by considering the difference of pairs of objects in two classes. The optimum is achieved when the value of  $\Gamma$  is at its highest. These three indices are monotonous, since they only take either intra-class dispersion or inter-class dispersion into account.

The *WB* index assesses the partitioning quality by considering the ratio of within-class sum-of-squares to between-class sum-of-squares. A smaller *WB* value indicates better data partitioning quality.

The Calinski-Harabasz index (*CH*) calculates the ratio of the average sum of squares between classes to the average sum of squares within classes. It takes into account both intra-class similarity and inter-class dispersion. When the index value is maximum, the number of classes is the optimum.

The *I* index measures the maximum distance between class centroids and the sum of distances between objects and their class centroids. If  $q = 2$ , the *I* index reduces to the *PBM* index. Larger values of *I* suggest better partitioning results.

The *F* index considers both the global dispersion between classes and the global dispersion within classes. The optimal number of classes is determined when the *F* index reaches the maximum.

The Dunn's index focuses on the compactness and separation of classes. It calculates the minimum pairwise distances between objects in different classes and the maximum pairwise distances between objects in the same classes. A higher value of Dunn's index suggests better partitioning quality, indicating that the classes are compact (small within-class dispersion) and well-separated (large between-class dispersion).

The Silhouette index evaluates the partitioning validity based on the difference between between-class and within-class dispersions. The optimal number of classes is determined when the value of the index is the maximum.

The Partition Separation index (*PS*) for hard partitioning version is introduced in the table above and specifically measures the separation between class centroids. The optimal number of classes is determined by maximizing the *PS* value.

The Davies-Bouldin index (*DB*) calculates the similarities between each class and all other classes, with the highest value for each class representing that class similarity. Then, the average of all class similarities is computed. A smaller index value indicates a better partitioning result.

The Xie-Beni index (*XB*) calculates the mean square distance between each object and its class centroid and the minimum square distance between class centroids. The optimal number of classes is determined by minimizing the *XB* value.

The *SD* index calculates the variances of class objects to evaluate the compactness within classes and also calculates the distances between class centroids to evaluate differences between classes. The optimal number of classes is reached when the minimum value of *SD* is found.

Partitioning validity indices are not only used to evaluate the quality of partitioning results but can also be used to determine the optimal number of classes. The selection of an appropriate index relies on the specific features of the data and the partitioning task.

## Chapter 3

### Developed unsupervised data stream partitioning method (STRFCM)

To eliminate the impact of parameters and the need for prior information, an unsupervised and non-parametric method named STRFCM was ultimately proposed, which can automatically partition the data stream and identify the optimal number of classes. This chapter provides an in-depth description of our proposed method.

#### 3.1 Principle

Our STRFCM method employs FCMO to partition data chunks and Weighted FCMO (WFCMO) to partition the exemplar set obtained during the chunk partitioning process. It consists of two main steps:

*Input:*

- Data stream

- 1) **Data chunk partitioning:** STRFCM employs the FCMO algorithm, which utilizes the  $L_1$  norm as its similarity criterion and the  $WB-L_{IM}$  partitioning validity index as its evaluation criterion, to partition incoming data chunks and obtain classes as well as the exemplar set.
- 2) **Fusion:** STRFCM utilizes the WFCMO algorithm, which uses the  $L_1$  norm as its similarity criterion and the  $F$  index as its evaluation criterion, to partition the exemplar set and obtain the final optimal partition.

*Output:* Optimal partition  $\mathcal{P}$  with discovered  $K$  classes and exemplar of each class

The details of the FCMO and WFCMO methods will be introduced in the next section.

The flow chart of our proposed method is presented in Figure 3.1.

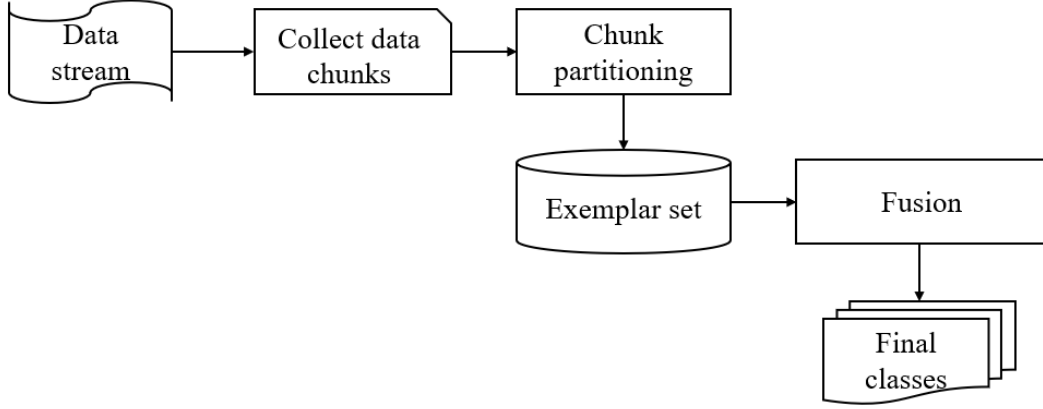


Figure 3.1: Flow chart of the proposed unsupervised partitioning method.

The data stream and data chunks can be expressed respectively as follows:

$$DS_t = \bigcup_{j=1}^t \{B_j\} \quad (3.1)$$

$$B_t = \bigcup_{i=1}^{N_d} \{x_i(t)\} \quad (3.2)$$

where  $t$  denotes the current time of the arriving data chunk and  $N_d$  is the size of the chunk. Each data chunk is partitioned using the FCMO algorithm and the centroid of each estimated class is retained as its exemplar. The exemplar set obtained by partitioning data chunk  $B_t$  is defined as follows:

$$BC_t = \{z_i(t)\}, \quad i = 1, 2, \dots, K_t \quad (3.3)$$

where  $z_i(t)$  represents the  $i$ th centroid of the  $i$ th class in data chunk  $B_t$  and  $K_t$  is the number of classes in data chunk  $B_t$ .

Eventually, a dataset  $BC$  containing all class centroids (exemplars) can be obtained, which is defined as:

$$BC = \bigcup_{j=1}^t \{BC_j\} \quad (3.4)$$

Partitioning of dataset  $BC$  by using the WFCMO algorithm gets a final optimal partition  $\mathcal{P} = \{C_1, C_2, \dots, C_K\}$ , where  $K$  is the final number of estimated classes.

## 3.2 Static dataset partitioning methods employed in STRFCM

In this section, we introduce the FCMO and WFCMO methods employed in our proposed STRFCM method and evaluate their performances.

### 3.2.1 Choice of FCMO for optimal partition

FCMO [11] is an optimization of the FCM algorithm [12], specifically designed for partitioning static datasets. It is an unsupervised and non-parametric method that eliminates the need for parameter tuning or specifying the number of classes in advance. It can automatically estimate the optimal number of classes with stable results. Therefore, our proposed method employs and optimizes FCMO for partitioning data chunks. FCMO utilizes an adaptive incremental procedure to initialize class centroids, and an unsupervised evaluation criterion (partitioning validity index) is adopted to estimate the optimal number of classes. In comparison to the FCM, support vector machines (SVM) [27], and ISODATA [95] algorithms, FCMO demonstrates superior partitioning performance [11]. FCMO mainly consists of five steps:

*Input:*

– Dataset  $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$

- 1) Subdivide the dataset into two classes.
- 2) Select the class with the largest dispersion which will be subdivided into two subclasses. The intra-class dispersion is measured by using Equation 2.9. If the partition is not valid in the end, choose the class with the next largest dispersion.
- 3) Choose the initial two subclass centroids from the selected class. The first subclass centroid is defined as the centroid of the selected class. The second centroid is chosen randomly from the selected class.
- 4) Use FCM algorithm to partition the dataset from  $k$  to  $k + 1$  classes and fine-tune class centroids.
- 5) Evaluate the partition with  $k + 1$  classes. This final step validates or rejects the partitioning results to find the optimal final partition and number of estimated classes. Unlike the validation criterion utilized by MLBG (Equation 2.11), FCMO does not use the coefficient of ponderation  $\eta$ . This validation criterion is as follows:



$$F(\mathcal{P}^{k+1}) > F(\mathcal{P}^k) \quad (3.5)$$

where  $F$  is a partitioning validity index defined as follows:

$$F(\mathcal{P}^k) = \frac{\bar{D}(\mathcal{P}^k) - \underline{D}(\mathcal{P}^k)}{2} \quad (3.6)$$

where  $\bar{D}(\mathcal{P}^k)$  and  $\underline{D}(\mathcal{P}^k)$  represents the global inter-class and intra-class dispersion, respectively.  $\underline{D}(\mathcal{P}^k)$  calculates the average weighted intra-class dispersion.

$$\underline{D}(\mathcal{P}^k) = \frac{1}{k} \sum_{j=1}^k \frac{N_j}{N} \underline{D}(C_j) \quad (3.7)$$

where  $N_j$  is the number of objects in class  $C_j$ , and  $N$  is the number of objects in dataset  $\mathcal{X}$ .  $\bar{D}(\mathcal{P}^k)$  calculates the average weighted dispersion between each class and other classes.

$$\bar{D}(\mathcal{P}^k) = \frac{1}{k} \sum_{j=1}^k \frac{N_j}{N} \bar{D}(C_j) \quad (3.8)$$

If the validation criterion is satisfied, one goes back to step 2 with  $k + 1$  determined classes and attempts to create  $k + 2$  classes. Conversely, if the criterion is not satisfied, return to step 2 with  $k$  classes and change the selected class by choosing the next largest dispersion class and so on. If none of the classes provides a valid partition, the algorithm stops and finds the optimal partition.

*Output:* Discovered  $K$  classes and exemplar of each class

The original similarity criterion utilized in FCMO is the  $L_2$  norm, which corresponds to the Euclidean distance. Instead of using  $L_2$  norm, our proposed method employs FCMO with the  $L_1$  norm as the similarity criterion. This choice is motivated by the fact that, the  $L_1$  norm can measure the exact difference between the attributes of two objects, providing their true similarity without amplification or attenuation.

FCMO employs the partitioning validity index as the evaluation criterion. It calculates the value of the validity index in each iteration and compares it with the value from the previous iteration to find the optimal partition. The index used in original FCMO is the  $F$  index [11],

which takes into account both the dispersion between classes and within classes. Larger values of  $F$  indicate better partitioning performance, and the optimal partition is achieved with the maximum value of  $F$ .

In addition to the  $F$  index, many partitioning validity indices have been proposed in the state of the art. Table 2.1 presents some common validity indices. The choice of the evaluation criterion significantly affects partitioning performance of FCMO. Consequently, we compared FCMO using four different validity indices, the  $F$  index, the Dunn's index [88], the Silhouette index [89] and the  $WB$  index [83], to find the optimal evaluation criterion. All of these criteria consider both intra-class and inter-class dispersion when evaluating a partition. The  $WB$  index was selected as the evaluation criterion for FCMO, as it outperformed the other three indices. The comparison results are displayed in Subsection 3.2.3. Since our proposed method employs the  $L_1$  norm as its similarity criterion, the selected  $WB$  index also employs the  $L_1$  norm to calculate both intra-class and inter-class similarity. The equation of the  $WB-L_I$  index is presented as follows:

$$WB-L_I = K \sum_{i=1}^K \sum_{x \in C_i} d(x, z_i) / \sum_{i=1}^K N_i d(z_i, zs) \quad (3.9)$$

where  $x$  is an object in class  $C_i$ ,  $N_i$  is the number of objects in class  $C_i$ ,  $z_i$  is the centroid of class  $C_i$ ,  $zs$  is the center of the whole dataset,  $d(x, z_i)$  calculates the  $L_1$  norm distance between object  $x$  and centroid  $z_i$ .

In the chunk partitioning process, it is expected that FCMO finds finer classes (more homogeneous classes with smaller variances) in order to preserve more detailed information so that during the fusion process more accurate results can be obtained. On this basis, we modified the  $WB-L_I$  index and introduced a novel index called  $WB-L_{IM}$  index, defined as follows:

$$WB-L_{IM} = \sum_{i=1}^K \sigma_i^2 \sum_{x \in C_i} d(x, z_i) / \sum_{i=1}^K N_i d(z_i, zs) \quad (3.10)$$

where  $\sigma_i^2$  is the variance of class  $C_i$ . The optimal number of classes and exemplars can be found by minimizing the value of  $WB-L_{IM}$  index.

$WB-L_{IM}$  differs from  $WB-L_I$  by introducing variance into its numerator, which leads to an increased number of iterations, ultimately allowing for the refinement of classes. During the iteration process using  $WB-L_{IM}$ , singleton classes are not retained as exemplars. These singleton classes are reassigned to the nearest classes in the next iteration.

The algorithm of FCMO using the  $L_1$  norm and the  $WB-L_{IM}$  index, referred to as FCMO- $L_I$ -WBM, is illustrated in Algorithm 3.1.

---

**Algorithm 3.1 FCMO- $L_1$ -WBM**

---

**Input:**

- Incoming data chunk  $B_j$

**Procedure:**

1. Subdivide the data chunk  $B_j$  into two classes
2. Select the class with the largest dispersion (Equation 2.9 with  $L_1$  norm) for subdivision into two subclasses
3. Select the initial two subclass centroids from the chosen class, with the first centroid being the class's centroid and the second centroid selected randomly from the same class
4. Partition the data chunk  $B_j$  into  $k + 1$  classes using FCM with  $L_1$  norm
5. Evaluate the partition with  $k + 1$  classes  
    **If**  $WB-L_{IM}(\mathcal{P}^{k+1}) < WB-L_{IM}(\mathcal{P}^k)$   
        Validate the partition of  $k + 1$  classes  
        Repeat steps 2-4 to try to create  $k + 2$  classes  
    **Else** return to step 2 with  $k$  classes and change the selected class by choosing the next largest dispersion class  
    **End if**  
    **If** none of the selected classes provides a valid partition  
        Stop  
    **End if**

**Output:** Discovered  $K_j$  classes in data chunk  $B_j$  and exemplar of each class

---

### 3.2.2 WFCMO for final optimal partition

The WFCMO method, an extension of FCMO, is designed to partition the exemplars of the classes obtained from multiple data chunks to obtain the final optimal partition while considering the size of the classes. It follows the same partitioning steps as FCMO, with modifications made to the FCMO algorithm in step 4.

Assume there is a set of  $N_e$  exemplars  $ES = \{e_1, e_2, \dots, e_i, \dots, e_{N_e}\}$ , which represents  $N_e$  subclasses, and  $e_i$  is the exemplar of the  $i$ th subclass  $SC_i$  which contains  $N_{sc_i}$  objects.  $ES$  can be partitioned into  $K$  classes  $(C_1, C_2, \dots, C_K)$ . When partitioning the exemplar set  $ES$  using original FCMO, the membership grade  $u_{ij}$  of the exemplar  $e_i$  to class  $C_j$  is calculated using Equation 2.8.

However, if we consider the number of objects  $N_{sc_i}$  in the subclass  $SC_i$  represented by the exemplar  $e_i$ , the membership level of  $e_i$  to class  $C_j$  is defined as:

$$u'_{ij} = u_{ij} \times Nsc_i \quad (3.11)$$

WFCMO uses weighted membership grade to partition the exemplar set. During this fusion process, it utilizes the  $F$  index as its evaluation criterion rather than the  $WB-L_{IM}$  or  $WB-L_I$  index. The reason is that the  $WB-L_{IM}$  index tends to obtain finer classes, and the  $WB-L_I$  index, despite outperforming the other three compared indices ( $F$  index [11], Dunn's index [88] and Silhouette index [89]), is not suitable for partitioning small population datasets like the exemplar set. The algorithm of WFCMO is presented in Algorithm 3.2.

---

**Algorithm 3.2 WFCMO**


---

**Input:**

- Exemplar set  $BC$

**Procedure:**

1. Subdivide the exemplar set  $BC$  into two classes
2. Select the class with the largest dispersion (Equation 2.9 with  $L_1$  norm) for subdivision into two subclasses
3. Select the initial two subclass centroids from the chosen class, with the first centroid being the class's centroid and the second centroid selected randomly from the same class
4. Partition the exemplar set  $BC$  into  $k + 1$  classes using FCM with  $L_1$  norm, and update the membership matrix  $U = [u'_{ij}]$  using Equation 2.8 and 3.11
5. Evaluate the partition with  $k + 1$  classes
  - If**  $F(\mathcal{P}^{k+1}) > F(\mathcal{P}^k)$ 
    - Validate the partition of  $k + 1$  classes
    - Repeat steps 2-4 to try to create  $k + 2$  classes
  - Else** return to step 2 with  $k$  classes and change the selected class by choosing the next largest dispersion class
  - End if**
  - If** none of the selected classes provides a valid partition
    - Stop
  - End if**

**Output:** Optimal partition  $\mathcal{P}$  with discovered  $K$  classes and exemplar of each class

---

The algorithm of STRFCM is presented as follows (Algorithm 3.3):

---

**Algorithm 3.3 STRFCM**

---

**Input:**

- Data stream  $DS_t = \cup_{j=1}^t \{B_j\}$

**Procedure:**

1. Load incoming data chunk  $B_j$
2. Partition data chunk  $B_j$  using FCMO- $L_I$ -WBM to obtain exemplar set  $BC_j$  (Algorithm 3.1)
3. Add exemplar set  $BC_j$  to global exemplar set  $BC$
4. **If** data stream continues, repeat steps 1-3  
**Else** proceed to step 5  
**End if**
5. Partition exemplar set  $BC$  using WFCMO (Algorithm 3.2)

---

**Output:** Optimal partition  $\mathcal{P}$  with discovered  $K$  classes and exemplar of each class

---

### 3.2.3 Evaluation of FCMO

In this subsection, we conduct a comparative assessment of the FCMO method using different similarity criteria and evaluation criteria, in order to illustrate that the FCMO using the  $L_1$  norm and the  $WB$ - $L_I$  index (referred as FCMO- $L_I$ -WB) outperforms other options. In addition, we compare FCMO with the unsupervised and parametric partitioning method AP. Furthermore, FCMO- $L_I$ -WBM is tested to demonstrate its capability to identify finer classes.

The FCMO using the  $L_1$  norm and the  $F$  index is referred to as FCMO- $L_I$ - $F$ , and the FCMO using the  $L_2$  norm and the  $F$  index is denoted as FCMO- $L_2$ - $F$ . Correspondingly, we have the FCMO- $L_I$ - $D$  method for FCMO using the  $L_1$  norm and the Dunn's index and FCMO- $L_I$ - $SIL$  employing the  $L_1$  norm and the Silhouette index. We point out that when FCMO employs the  $L_1$  norm, the same norm is also utilized in these partitioning validity indices (evaluation criteria).

#### 3.2.3.1 Experimental datasets




Two synthetic hyperspectral image datasets are utilized to evaluate partitioning performance. Detailed information about these two datasets are presented in Table 3.1 and 3.2.

**Dataset 1** is a synthetic hyperspectral image of algae, displayed in Table 3.1 (a). It exhibits a spatial resolution of 60x60 pixels, characterized by 100 spectral bands. It consists of nine GT

subclasses which can be aggregated into three main GT classes: water, substrate and vegetation. These knowledges are only used to assess the partitioning results.

The main water class encompasses three subclasses: deep water ( $C_1^1$ ), shallower water ( $C_1^2$ ), and turbid water ( $C_1^3$ ). The main substrate class consists of two subclasses: pebble ( $C_2^1$ ) and sand ( $C_2^2$ ). The main vegetation class is subdivided into four subclasses: two subclasses of green algae, ulva ( $C_3^1$ ) and enteromorpha ( $C_3^2$ ), one subclass of fucus ( $C_3^3$ ), and the final subclass ( $C_3^4$ ), which is a mixed class encompassing elements of both substrate and vegetation. Table 3.1 (b) and (c) represent GT images, with (b) consisting of three main GT classes and (c) containing nine GT subclasses.



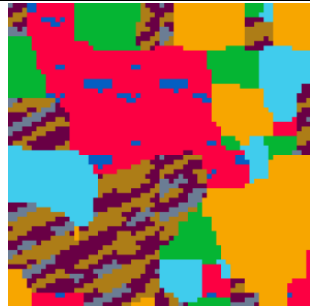













Table 3.1: Dataset 1 – hyperspectral image of algae, GT images and GT class details.

		
(a) Original image 60x60x100 (visualized in RGB mode)	(b) GT1 image (3 main GT classes)	(c) GT2 image (9 GT subclasses)
Labels of 3 main GT classes	Labels of 9 GT subclasses	
■ $C_1$ Water (1324 pixels)	■ $C_1^1$ Deep water (989 pixels)	
	■ $C_1^2$ Shallow water (158 pixels)	
	■ $C_1^3$ Turbid water (177 pixels)	
■ $C_2$ Substrate (1067 pixels)	■ $C_2^1$ Pebble (281 pixels)	
	■ $C_2^2$ Sand (786 pixels)	
■ $C_3$ Vegetation (1209 pixels)	■ $C_3^1$ Ulva (green algae) (255 pixels)	
	■ $C_3^2$ Enteromorpha (green algae) (416 pixels)	
	■ $C_3^3$ Fucus (brown algae) (493 pixels)	
	■ $C_3^4$ Substrate and other types of vegetation (45 pixels)	

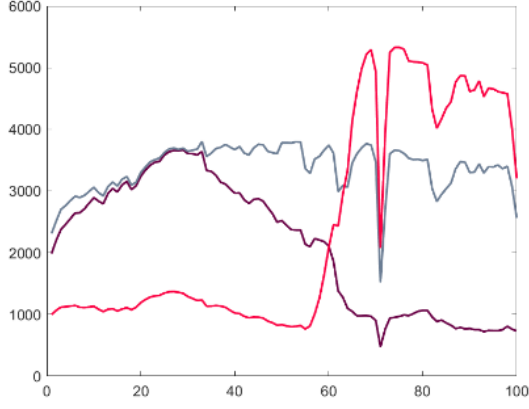
**Dataset 2** is a hyperspectral image of invasive vegetation with a spatial resolution of 64x64 pixels, characterized by 54 spectral bands (Table 3.2 (a)). It consists of five main GT classes which can be further subdivided into eight GT subclasses.

The main *pinus halepensis* class consists of two subclasses: dense *pinus halepensis* ( $C_2^1$ ) and sparse *pinus halepensis* ( $C_2^2$ ). The main peach trees class comprises three subclasses: healthy peach trees ( $C_3^1$ ), early wilting peach trees ( $C_3^2$ ), and wilting peach trees ( $C_3^3$ ). GT images for this dataset are presented in Table 3.2 (b) and (c), containing five main GT classes and eight GT subclasses, respectively.

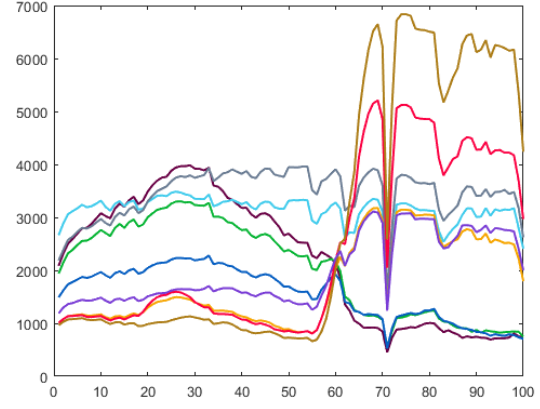
Table 3.2: Dataset 2 – hyperspectral image of invasive vegetation, GT images and GT class details.

			
(a) Original image 64x64x54 (visualized in RGB mode)	(b) GT1 image (5 main GT classes)	(c) GT2 image (8 GT subclasses)	
Labels of 5 main GT classes		Labels of 8 GT subclasses	
 C <sub>1</sub> River (452 pixels)		C <sub>1</sub> River (452 pixels)	
 C <sub>2</sub> Pinus halepensis (1068 pixels)		C <sub>2</sub> <sup>1</sup> Pinus halepensis (dense) (986 pixels)	
		C <sub>2</sub> <sup>2</sup> Pinus halepensis (sparse) (82 pixels)	
 C <sub>3</sub> Peach trees (1189 pixels)		C <sub>3</sub> <sup>1</sup> Peach trees (healthy) (175 pixels)	
		C <sub>3</sub> <sup>2</sup> Peach trees (early wilting) (513 pixels)	
		C <sub>3</sub> <sup>3</sup> Peach trees (wilting) (501 pixels)	
 C <sub>4</sub> Arundo donax (500 pixels)		C <sub>4</sub> Arundo donax (500 pixels)	
 C <sub>5</sub> Buildings (887 pixels)		C <sub>5</sub> Buildings (887 pixels)	

The mean spectral signatures of the classes in these two hyperspectral images are depicted in Figure 3.2 and 3.3. Each class's mean spectral signature is represented by a line of a distinct color, corresponding to the colors used in the GT images displayed in Table 3.1 and 3.2.

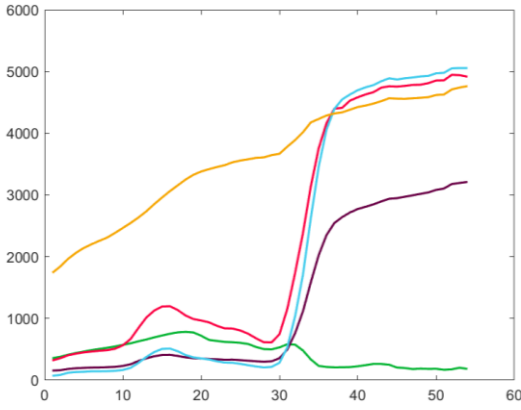


(a)

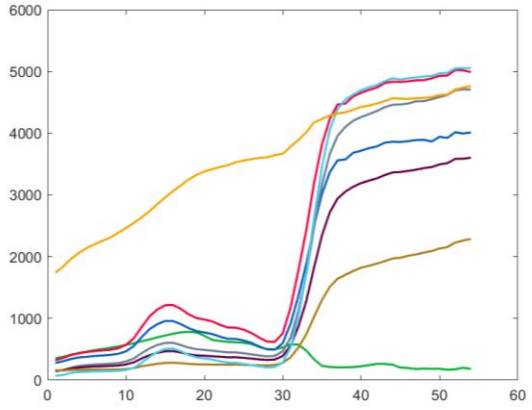


(b)

Figure 3.2: Mean spectral signatures of the GT classes in Dataset 1. (a): 3 main GT classes, (b): 9 GT subclasses.



(a)



(b)

Figure 3.3: Mean spectral signatures of the GT classes in Dataset 2. (a): 5 main GT classes, (b): 8 GT subclasses.

### 3.2.3.2 Performance evaluation

To evaluate the partitioning performance, we used the external validity index: purity. An ideal partition yields a purity value of 100%, while a poor partition results in a value close to 0%. The average purity of all estimated classes is defined as follows:

$$purity = \frac{1}{K} \sum_{i=1}^K \max_{j=1, \dots, K_{GT}} \left( \frac{N_i^j}{N_i} \right) \quad (3.12)$$



where  $K_{GT}$  is the number of the main GT classes,  $N_i$  is the number of objects in the  $i$ th estimated class,  $N_i^j$  is the number of objects allocated to the  $j$ th main GT class and  $K$  is the number of estimated classes.

Table 3.3 presents the purity value, number of estimated classes (NC), and CPU execution time obtained by FCMO using criteria (similarity criteria and evaluation criteria)  $L_1-F$ ,  $L_2-F$ ,  $L_1-D$ ,  $L_1-SIL$  and  $L_1-WB$  when partitioning Dataset 1. In addition, we compared these methods with AP. The preference parameter and the damping rate are input parameters of AP. In the assessment, the damping rate ranges from 0.6 to 0.9, and preference parameter is set to either the median or the minimum value of the similarity matrix. The partitioning performance of AP on Dataset 1 is illustrated in Table 3.4.

Table 3.3: Partitioning performances of FCMO using different similarity criteria and evaluation criteria on Dataset 1.

Methods	Purity (%)	NC	CPU (s)
FCMO- $L_1-F$	100	6	35.38
FCMO- $L_2-F$	98.94	6	<b>32.57</b>
FCMO- $L_1-D$	100	7	44.85
FCMO- $L_1-SIL$	100	6	127.38
FCMO- $L_1-WB$	<b>100</b>	<b>8</b>	127.30

Table 3.4: Partitioning performance of the AP method on Dataset 1.

<i>Input parameters set by user:</i>		Purity (%)	NC	CPU (s)
<i>Preference parameter</i>	<i>Damping rate</i>			
Median	0.6	Non-convergence		
	0.7	Non-convergence		
	0.8	100	13	58.43
	0.9	100	13	64.12
Minimum	0.6	Non-convergence		
	0.7	Non-convergence		
	0.8	Non-convergence		
	0.9	<b>100</b>	<b>9</b>	<b>46.56</b>

In comparison to FCMO utilizing the  $L_2$  norm, FCMO using the  $L_1$  norm achieves a higher purity of 100%. FCMO- $L_1$ - $F$  gives a purity of 100% (Table 3.3), while FCMO- $L_2$ - $F$  provides 98.94%. Among the methods using the  $L_1$  norm, we can find that the FCMO- $L_1$ - $WB$  method identified more classes than others, which is closest to the GT of 9 subclasses. FCMO- $L_1$ - $WB$  compared to FCMO using criteria  $L_1$ - $F$ ,  $L_1$ - $D$  and  $L_1$ - $SIL$ , discovers more detailed information. Compared with the partitioning results of the AP method displayed in Table 3.4, with appropriate parameter values of AP, the purity values of FCMO using the  $L_1$  norm and AP are the same (100% purity) and the numbers of estimated classes are within reasonable range. However, we observed that when the value of parameter damping rate is wrongly chosen, the AP method does not converge, which means the choice of input parameter values significantly impacts the performance of AP.

Figure 3.4 presents the partitioning results on Dataset 1 using the FCMO method with three criteria  $L_1$ - $F$ ,  $L_2$ - $F$  and  $L_1$ - $WB$  and the AP method, offering an intuitive assessment of the partitioning performance. In Figure 3.4 (c), all main GT classes were successfully identified. However, FCMO- $L_1$ - $F$  failed to identify the GT pebble subclass (blue), the shallow water subclass (green) and the subclass containing data of mixed types (purple). Additionally, some points belonging to the ulva subclass (orange) were assigned to the enteromorpha subclass (red). Nevertheless, both ulva and enteromorpha subclasses belong to the same main GT class of vegetation. Therefore, it does not significantly impact the partitioning performance of FCMO- $L_1$ - $F$ .

In Figure 3.4 (d), similar to FCMO- $L_1$ - $F$ , FCMO- $L_2$ - $F$  also missed the identification of the three subclasses. However, there is a notable difference: some points belonging to the ulva subclass (orange) are incorrectly assigned to the turbid water subclass (dark blue). These two subclasses belong to different main GT classes, which has a more significant impact on the partitioning performance of FCMO- $L_2$ - $F$ .

We observed from Figure 3.4 (e) that all GT subclasses were successfully detected by using FCMO- $L_1$ - $WB$  except the subclass with data of mixed types (purple), while FCMO- $L_1$ - $F$  only identified 6 GT subclasses. Additionally, some points belonging to the sand subclass (grey) were assigned to the pebble subclass (blue). However, these two subclasses belong to the same main class of substrate, which does not significantly impact the partitioning performance of FCMO- $L_1$ - $WB$ . The partitioning result of AP with preference parameter set to minimum and damping rate of 0.9, displayed in Figure 3.4 (f), is similar with the result of FCMO- $L_1$ - $WB$ .

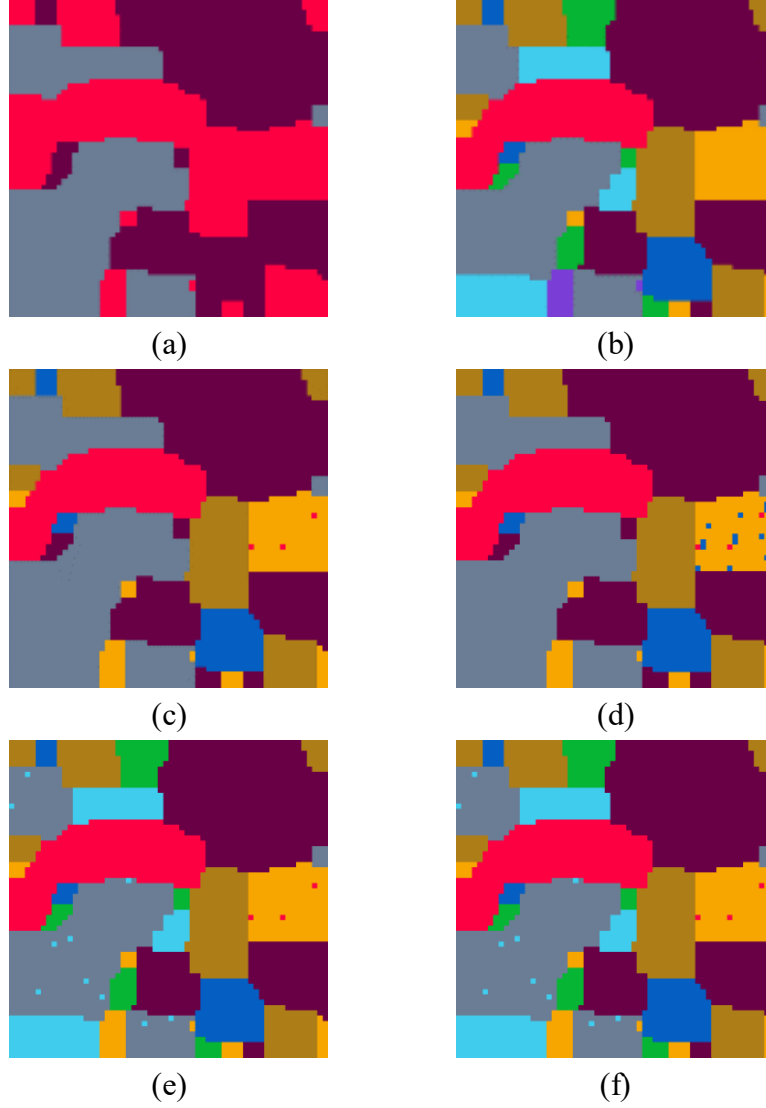


Figure 3.4: Partitioning results on Dataset 1. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c), (d) and (e): FCMO using criteria  $L_1-F$ ,  $L_2-F$  and  $L_1-WB$ , respectively, (f): AP – preference parameter set to minimum and damping rate of 0.9.

Table 3.5 presents the purity value, number of estimated classes, and CPU execution time obtained when partitioning Dataset 2 using FCMO with five criteria  $L_1-F$ ,  $L_2-F$ ,  $L_1-D$ ,  $L_1-SIL$ , and  $L_1-WB$ . The partitioning performance of AP on Dataset 2 is illustrated in Table 3.6.

As shown in Table 3.5, the purity of FCMO- $L_1-F$  (93.51%) is higher than that of FCMO- $L_2-F$  (85.39%). Additionally, the number of estimated classes by FCMO- $L_1-F$  is much closer to the number of GT subclasses (8 GT subclasses), indicating that FCMO- $L_1-F$  discovers more details than FCMO- $L_2-F$  during partitioning. In addition, both FCMO- $L_1-F$  and FCMO- $L_1-WB$  achieved the best performance with a purity of 93.51%. The reason their purity values are the same is that these two methods performed the same number of iterations. However, the

performances of FCMO- $L_1$ - $D$  and FCMO- $L_1$ - $SIL$  are poor on partitioning Dataset 2 and cannot be considered competitive.

Table 3.5: Partitioning performances of FCMO using different similarity criteria and evaluation criteria on Dataset 2.

Methods	Purity (%)	NC	CPU (s)
FCMO- $L_1$ - $F$	<b>93.51</b>	<b>6</b>	30.97
FCMO- $L_2$ - $F$	85.39	4	37.54
FCMO- $L_1$ - $D$	68.53	2	<b>25.08</b>
FCMO- $L_1$ - $SIL$	69.48	3	30.75
FCMO- $L_1$ - $WB$	<b>93.51</b>	<b>6</b>	42.56

Table 3.6: Partitioning performance of the AP method on Dataset 2.

<i>Input parameters set by user:</i>		Purity (%)	NC	CPU (s)
<i>Preference parameter</i>	<i>Damping rate</i>			
Median	0.6	Non-convergence		
	0.7	Non-convergence		
	0.8	Non-convergence		
	0.9	95.36	20	50.46
Minimum	0.6	Non-convergence		
	0.7	Non-convergence		
	0.8	Non-convergence		
	0.9	93.8	12	66

Based on the partitioning performance displayed in Table 3.6, it becomes evident that the choice of input parameters significantly impacts the performance of the AP method. For instance, when the preference parameter is set to the median, and the damping rate is 0.8, the AP method does not converge. When the preference parameter is set to the median, and the damping rate is 0.9, the AP method achieves a purity of 95.36%, which is higher than the highest purity in Table 3.5 (93.51%). However, under this parameter setting, AP identifies 20 estimated classes, a number much larger than the GT (8 GT subclasses). This indicates that the dataset is over-partitioned. Conversely, with appropriate parameter values, the AP method achieves a purity of 93.8%, similar to the highest purity in Table 3.5 (93.51%).

Figure 3.5 presents the partitioning results on Dataset 2 using the FCMO method with criteria  $L_1$ - $F$ ,  $L_2$ - $F$  and  $L_1$ - $WB$  and the AP method. We can find that FCMO- $L_1$ - $F$  successfully

detected all main GT classes (Figure 3.5 (c)), while FCMO- $L_2-F$  failed to identify the arundo donax class (blue) (Figure 3.5 (d)), resulting in lower partitioning performance. In addition, we found that the partitioning results of FCMO- $L_1-WB$  and FCMO- $L_1-F$  on the Dataset 2 are same. According to Figure 3.5 (c), (e), and (f), the FCMO- $L_1-F$ , FCMO- $L_1-WB$  and AP methods incorrectly assigned some points belonging to the GT healthy peach trees subclass (grey) to the arundo donax subclass (blue). However, these two subclasses do not belong to the same main GT class, which leads to a decrease in partitioning performance. In addition, there were some points belonging to the GT healthy peach trees subclass (grey) were mistakenly assigned to the dense pinus halepensis subclass (red) by AP.

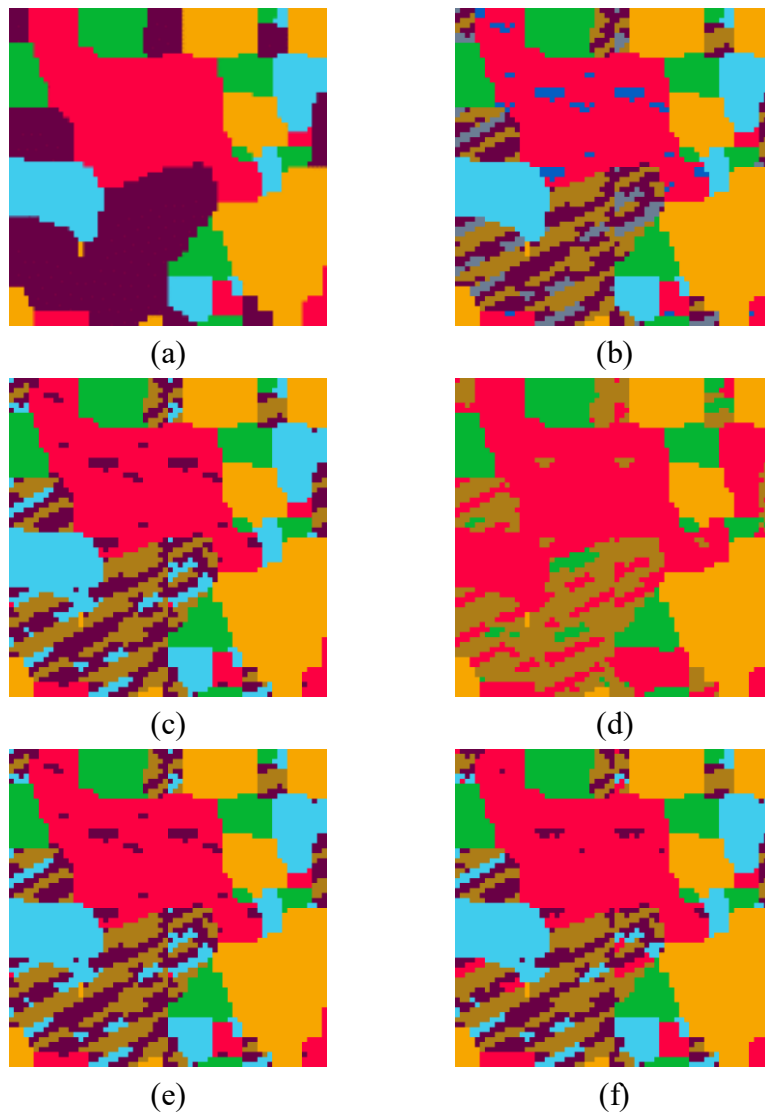


Figure 3.5: Partitioning results on Dataset 2. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c), (d) and (e): FCMO using criteria  $L_1-F$ ,  $L_2-F$  and  $L_1-WB$ , respectively, (f): AP – preference parameter set to minimum and damping rate of 0.9.

According to Table 3.3 and 3.5, both the FCMO- $L_I$ - $F$  and FCMO- $L_I$ - $WB$  methods achieve the highest purity in partitioning both Dataset 1 and Dataset 2. To further compare these two methods, we conducted an additional test.

Since FCMO is used to partition data chunks by our method, in this evaluation, we utilized FCMO- $L_I$ - $F$  and FCMO- $L_I$ - $WB$  to partition only data chunks of varying sizes in Dataset 2 instead of the entire dataset. Their partitioning performances are demonstrated in Table 3.7.

Table 3.7: Partitioning performances of FCMO- $L_I$ - $F$  and FCMO- $L_I$ - $WB$  on different sizes of data chunks in Dataset 2.

Methods	Chunk size	Purity (%)	NC	CPU (s)
FCMO- $L_I$ - $F$	128	94.23	4	0.54
	256	94.24	4	0.95
	512	91.95	4	2.31
	1024	92.52	4	4.14
	2048	91.29	4	7.48
FCMO- $L_I$ - $WB$	128	Non-convergence		
	256	99.41	8	0.91
	512	96.12	5	2.96
	1024	92.52	4	4.07
	2048	91.29	4	9.56

It is obvious that the overall performance of FCMO- $L_I$ - $WB$  is better than that of FCMO- $L_I$ - $F$ . We observed that with a chunk size of 128 pixels, the FCMO- $L_I$ - $WB$  method did not converge, indicating that FCMO- $L_I$ - $WB$  may not be suitable for partitioning datasets with small populations. Nevertheless, in the context of partitioning large streaming data, this limitation becomes negligible.

In summary, compared with FCMO using other similarity criteria and evaluation criteria, FCMO- $L_I$ - $WB$  demonstrates superior performance on both datasets. Although AP also exhibits strong partitioning performance, it has to predetermine the input parameters which greatly affect its performance. In contrast, FCMO- $L_I$ - $WB$  does not introduce parameters, ensuring a more stable partitioning performance. Based on the above analysis, FCMO- $L_I$ - $WB$  achieves the best partitioning performance on both datasets.

In order to obtain finer classes during the data chunk partitioning process to retain more detailed information for the subsequent fusion process, we modified the  $WB-L_I$  index and proposed the  $WB-L_{IM}$  index as the evaluation criterion of FCMO. We conducted a comparison between FCMO- $L_I$ -WBM and FCMO- $L_I$ -WB using data chunks containing 256 and 512 pixels from Dataset 2 to demonstrate FCMO- $L_I$ -WBM's capability to refine classes and extract more detailed information.

Figure 3.6 and 3.7 demonstrate the number of estimated classes and the size of each class (number of objects in the class) obtained by employing FCMO- $L_I$ -WB and FCMO- $L_I$ -WBM to partition data chunks of 256 and 512 pixels from Dataset 2.

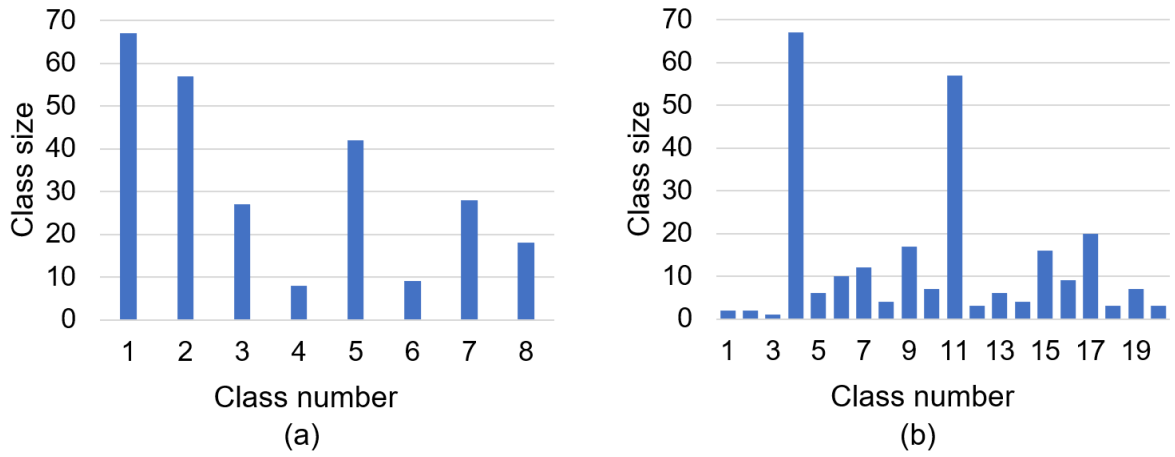


Figure 3.6: The sizes of estimated classes on data chunk of 256 pixels. (a): FCMO- $L_I$ -WB, (b): FCMO- $L_I$ -WBM.

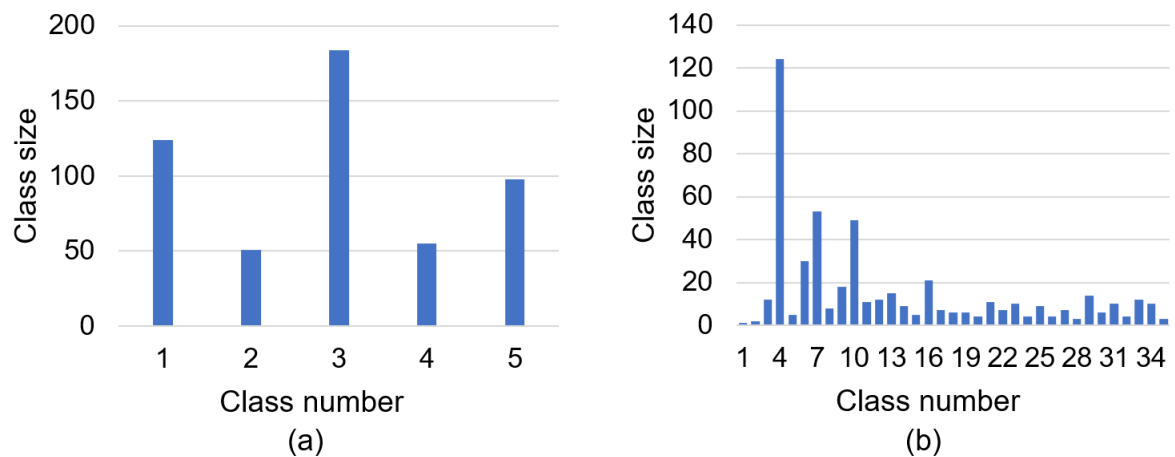


Figure 3.7: The sizes of estimated classes on data chunk of 512 pixels. (a): FCMO- $L_I$ -WB, (b): FCMO- $L_I$ -WBM.

Figure 3.6 and 3.7 clearly show that, when compared to the results of  $\text{FCMO-}L_I\text{-WB}$  (8 and 5 estimated classes with the chunk sizes of 256 and 512 pixels, respectively),  $\text{FCMO-}L_I\text{-WBM}$  identified more classes with smaller sizes (20 and 35 estimated classes with the chunk sizes of 256 and 512 pixels, respectively), which aligns with our goal of obtaining finer classes.

### 3.2.4 Evaluation of WFCMO

To assess the performance of WFCMO on exemplar sets, we generated a dataset comprising 335 objects. This dataset encompasses seven GT subclasses of varying sizes, which can be partitioned into two main GT classes. Each object in this dataset is characterized by two attributes. The details of the generated dataset are illustrated in Table 3.8.

Table 3.8: Details of the two main GT classes and seven GT subclasses in the generated dataset.










Labels of 2 main GT classes	Labels of 7 GT subclasses	
 $C_1$ (185 objects)		$SC_1$ (5 objects)
		$SC_2$ (150 objects)
		$SC_3$ (30 objects)
 $C_2$ (150 objects)		$SC_4$ (20 objects)
		$SC_5$ (20 objects)
		$SC_6$ (100 objects)
		$SC_7$ (10 objects)

Figure 3.8 demonstrates the partitioning results of WFCMO and FCMO on the generated dataset. The red filled points depict the exemplars of the seven GT subclasses, while the red hollow point represents the GT centroids of the two main classes. The black asterisk indicates the estimated centroids of the two main classes obtained by WFCMO when partitioning the exemplar set (red filled points). The red asterisk represents the estimated centroids of the two main classes obtained by  $\text{FCMO-}L_I\text{-F}$  during the partitioning of the exemplar set.

It is expected that the partitioning result on the exemplar set must be close to the GT centroids of  $C_1$  and  $C_2$ . According to Figure 3.8, it is evident that the distance between the black asterisk and the red hollow point is small. This observation highlights that, during the fusion process, the WFCMO method takes into account the weight of each exemplar, demonstrating good partitioning performance. In contrast,  $\text{FCMO-}L_I\text{-F}$  tends to identify biased estimated class



centroids without considering the sizes of classes. Based on these findings, it is better to use WFCMO in the fusion process instead of FCMO- $L_I$ - $F$  for partitioning exemplar sets.

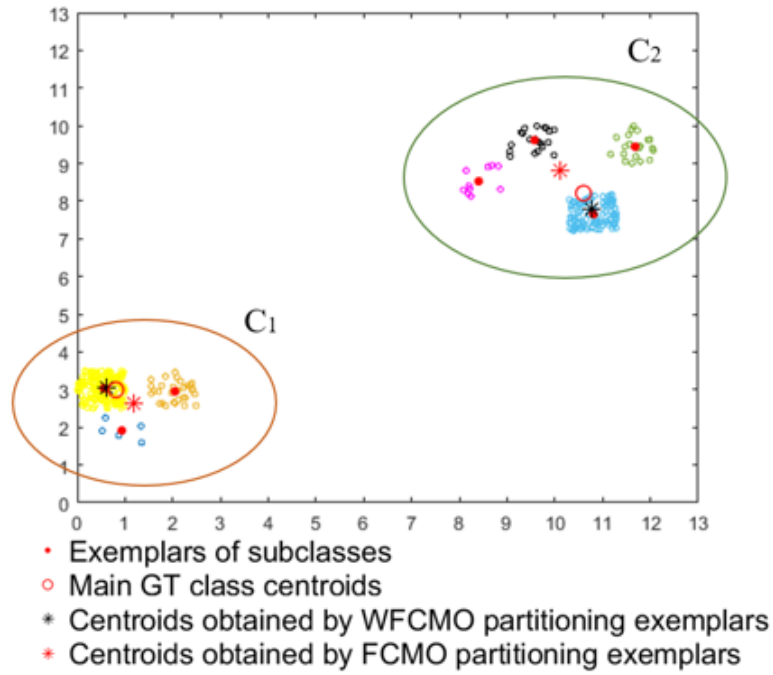


Figure 3.8: Partitioning results of WFCMO and FCMO on the generated dataset.

## Chapter 4

### Evaluation of developed method (STRFCM)

In this chapter, we assessed our proposed STRFCM method and compared it with two parametric data stream unsupervised partitioning methods, STRAP [14] and CODAS [15], using Dataset 1 and Dataset 2 presented in Subsection 3.2.3.1. In addition, we utilized a real-world dataset “Image Segmentation” [96] to conduct a comparison of our method with four parametric partitioning methods across the categories of supervised (AAPStream [17]), semi-supervised (CluStream [13]) and unsupervised (DenStream [16] and STRAP [14]). These compared methods have already been discussed in Section 2.3.

The “Image Segmentation” dataset, referred to as **Dataset 3**, is sourced from the UCI Machine Learning Repository [96]. It contains 2310 instances characterized by 19 features, such as the mean values of the R, G, and B components and the mean saturation value. These instances can be assigned to seven classes (brick face, sky, foliage, cement, window, path, and grass). The instances were randomly selected from a database containing seven outdoor images. The images were hand-segmented to create a partitioning for every pixel. Each instance is a 3x3 region.

#### 4.1 Evaluation protocol

In this section, we present the evaluation protocol for the partitioning performances of our proposed method, STRFCM, and two parametric unsupervised methods, STRAP and CODAS, using two hyperspectral datasets. The selection of STRAP and CODAS for comparison is grounded in their ability to obviate the need for predetermining the number of classes and their open-source nature.

The assessment of partitioning performance relies on external validity indices, purity and the kappa index [97]. Furthermore, during the evaluation, we also consider the number of estimated classes (NC), number of detected outliers (OT), and CPU execution time.

The kappa index measures the agreement degree between two or more observers who partition objects into different classes. In our assessment, one observer gives the GT data labels, and the other provides the estimated data labels. It is calculated as follows:

$$kappa = (p_o - p_e)/(1 - p_e) \quad (4.1)$$

where:

$p_o$  is the actual observed agreement among observers. It computes the sum of the number of correctly partitioned samples for each main GT class divided by the total number of samples, which is the overall classification accuracy.

$p_e$  represents the hypothetical probability of chance agreement. It calculates the sum of the number of samples partitioned by one observer for each main GT class times the number of samples partitioned by another observer, which is then divided by the square of the total number of samples.

An ideal partition is characterized by a kappa value of 100%, while a poor partition typically exhibits a kappa value close to 0%.

In order to reduce the execution time, our method partitions multiple data chunks in parallel utilizing four CPU cores. The partitioning process was executed on an Intel(R) Core (TM) i5-1135G7 processor, clocked at 2.4 GHz, with 16 GB of RAM.

During the assessment of our proposed method, the data in hyperspectral image is entered sequentially into data chunks of the same size, and these data chunks are partitioned in a time series manner. We gradually increase the data chunk size up to the size of Dataset 1 and 2 (excluded). The data chunk size is determined based on the number of pixels in each row of the image as the basic unit.

The input parameters of STRAP include the initial batch size, threshold distance, max cache, sliding window size, along with the preference parameter and damping rate of AP. CODAS uses the parameters threshold density and threshold radius. It is necessary to specify their input parameters before partitioning. During the assessment, we conducted extensive tests to determine optimal parameter values, which consumed a significant amount of time.

## 4.2 Evaluation on hyperspectral image of algae

In this section, we assess our proposed STRFCM method and compare it with two parametric unsupervised methods, STRAP and CODAS, by partitioning Dataset 1. The number of estimated classes, number of detected outliers, purity, kappa index, and execution time are considered to evaluate the partitioning performance.

Table 4.1 presents the partitioning performances of our method (STRFCM), STRAP, and CODAS on Dataset 1. Note that this table only contains a portion of representative partitioning

results of STRAP and CODAS. The comprehensive parameter tuning process entails a substantial number of tests, making it impractical to display the entirety of the results within this table.

Table 4.1: Partitioning performances of our method (STRFCM), STRAP, and CODAS on Dataset 1.

Performance of our proposed STRFCM method								
Chunk size		NC	OT	Purity (%)		Kappa (%)	CPU (s)	
240		8	None	100		100	50.56	
360		8		100		100	51.27	
720		8		100		100	183.64	
1200		8		100		100	1079.97	
1800		9		100		100	2132.63	
Performance of STRAP method								
Input parameters set by user:								
Preference parameter=median, sliding window size=3600 (full image)				NC	OT	Purity (%)	Kappa (%)	CPU (s)
Threshold distance	Initial batch	Damping rate	Max cache					
6000	240	0.8	10	8	0	100	100	0.28
	360			8	0	100	100	0.29
	720			9	0	100	100	0.29
	1200			9	0	100	100	0.30
	1800			10	0	100	100	0.31
7000	240	0.8	10	7	2	99.09	99.50	0.25
	360			7	2	99.09	99.50	0.22
	720			8	2	99.21	99.50	0.24
	1200			8	2	99.21	99.50	0.27
	1800			10	0	100	100	0.32
8000	240	0.8	10	7	0	99.09	99.50	0.25
	360			7	0	99.09	99.50	0.21
	720			8	0	99.21	99.50	0.35
	1200			8	0	99.21	99.50	0.28
	1800			10	0	100	100	0.31

6000	1200	0.8	10	8	0	100	100	0.28
			30	8	20	100	100	0.30
			50	8	31	99.93	99.96	0.30
			70	9	0	100	100	0.31
			100	9	0	100	100	0.30
Performance of CODAS method								
Input parameters set by user:			NC	OT	Purity (%)	Kappa (%)	CPU (s)	
Threshold density	Threshold radius							
3	200		40	73	100	100	1.13	
	300		16	13	100	100	0.86	
	400		12	1	100	100	0.59	
	500		10	1	100	100	0.59	
	600		9	1	100	100	0.41	
	700		7	0	100	100	0.43	
4	200		38	187	100	100	1.25	
	300		16	32	100	100	0.82	
	400		12	4	100	100	0.57	
	500		11	1	100	100	0.52	
	600		10	1	100	100	0.41	
	700		7	0	100	100	0.41	
5	200		29	282	100	100	1.06	
	300		18	44	100	100	0.76	
	400		11	12	100	100	0.53	
	500		10	5	100	100	0.53	
	600		9	5	100	100	0.45	
	700		8	0	100	100	0.40	

According to Table 4.1, our proposed method (STRFCM) achieved a purity of 100% on Dataset 1, and the chunk size has no effect on values of purity and kappa index. However, we found that as the chunk size increases, the execution time becomes longer. This happens because, during the chunk partitioning step, finer classes are obtained, leading to an increase in the number of iterations in FCMO. As the iteration count rises, so does the execution time. On

the other hand, larger data chunks require more time for each iteration, leading to a significant increase in execution time as the number of iterations grows.

Two examples of the partitioning results of chunk sizes of 240 pixels and 1800 pixels are presented in Figure 4.1. We can find that all main GT classes and 8 GT subclasses were detected, except for the subclass with data of mixed types (purple). This is because the mean spectral signature of the ulva subclass (orange) is close to that of the subclass with data of mixed types (purple) (Figure 3.2 (b)). Consequently, it becomes challenging for algorithms to distinguish between these two subclasses. Furthermore, we observed that several points belonging to the GT ulva subclass (orange) were assigned to the enteromorpha subclass (red) (Figure 4.1 (c) and (d)). However, these two subclasses are part of the main GT class of vegetation, which has almost no impact on the partitioning performance.

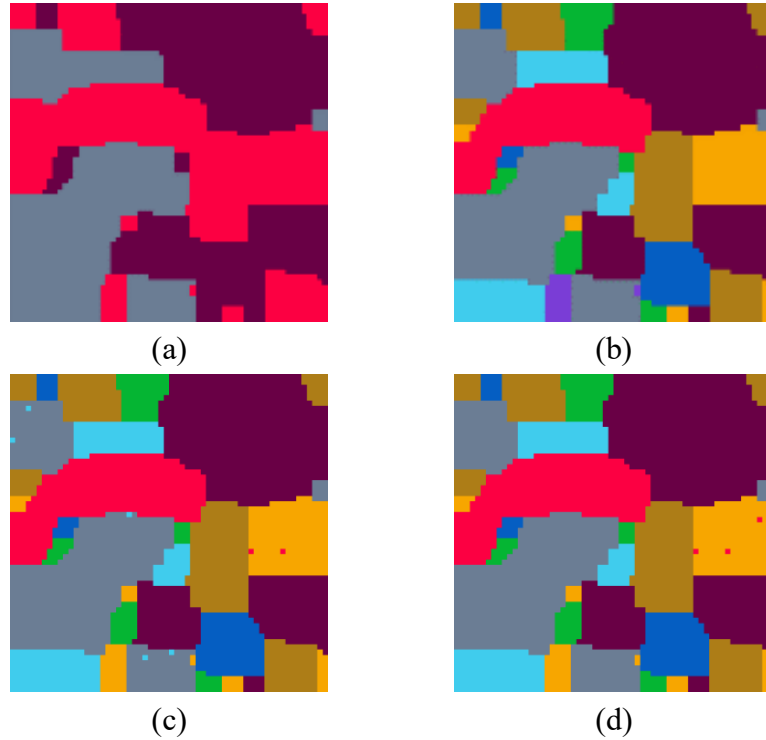


Figure 4.1: Partitioning results on Dataset 1 using our proposed method STRFCM. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c): data chunk size 240 pixels, (d): data chunk size 1800 pixels.

Figure 4.2 demonstrates the mean spectral signatures  $\pm$  standard deviation of the final estimated classes of Dataset 1 with the chunk size of 240 pixels. The black lines depicted in Figure 4.2 represent the mean spectral signatures of the corresponding GT subclasses. We can

observe that the mean spectral signatures of the estimated classes are similar to those of the GT subclasses, which means the partitioning performance of STRFCM is high.

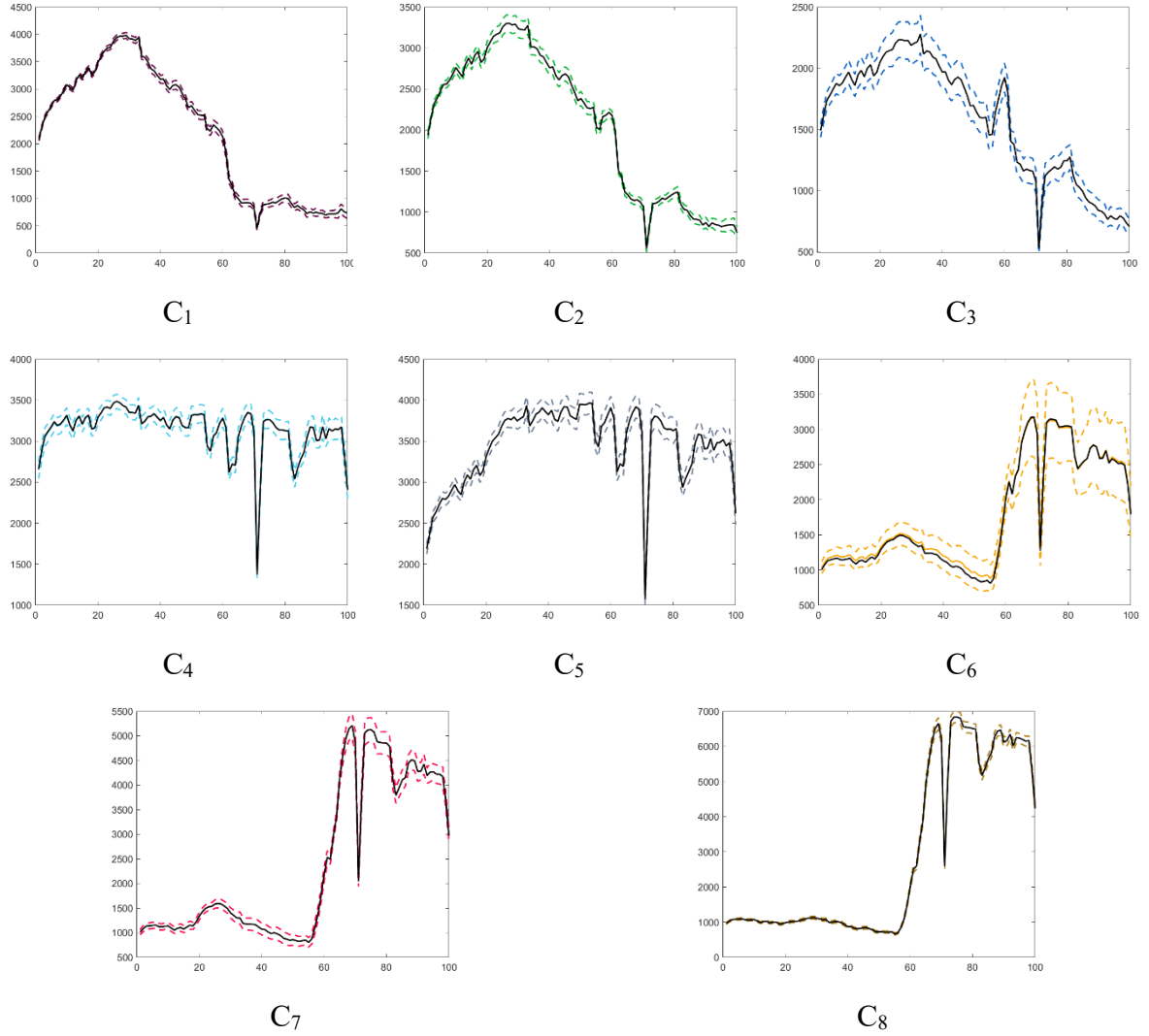


Figure 4.2: Mean spectral signatures  $\pm$  standard deviation of 8 estimated classes of Dataset 1 – data chunk size 240 pixels.

The partitioning performance of STRAP displayed in Table 4.1 demonstrates that the choice of parameter settings has an impact on its partitioning performance. In the majority of the partitioning results presented in this table, both purity and kappa index exhibit values of 100%. However, under certain parameter settings, the purity is lower than 100%, such as the partitioning result (purity of 99.09%) with the parameter setting: threshold distance of 7000, initial batch of 360, damping rate of 0.8, and max cache of 10. Additionally, STRAP occasionally detects outliers in certain cases.

Figure 4.3 demonstrates the partitioning results with the parameter setting: threshold distance of 6000, initial batch of 720, damping rate of 0.8, and max cache of 10 (referred as parameter setting A), and another parameter setting: threshold distance of 7000, initial batch of 240, damping rate of 0.8, and max cache of 10 (referred as parameter setting B). These settings yield purity values of 100% and 99.09% (the highest and lowest purity of STRAP), respectively.

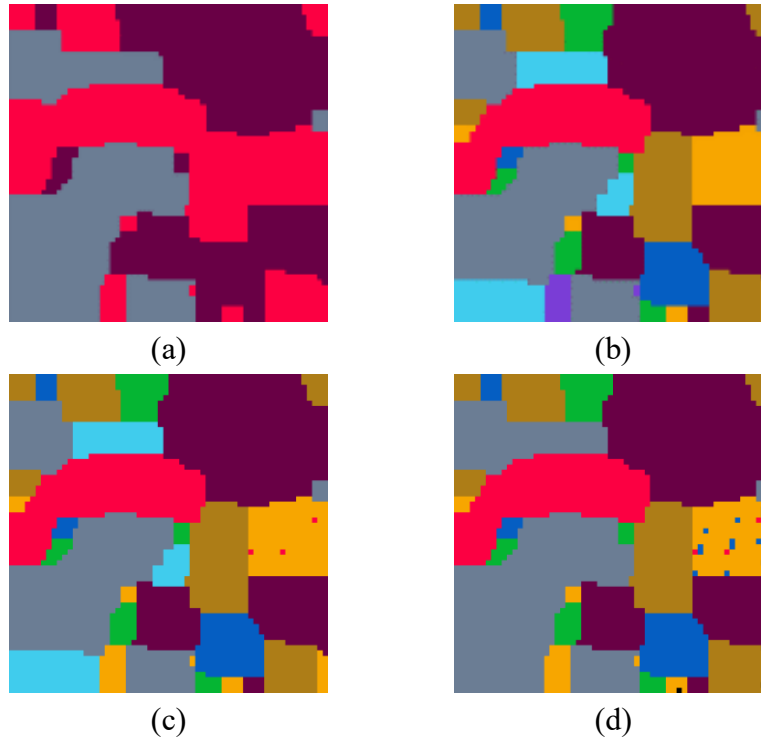


Figure 4.3: Partitioning results on Dataset 1 using STRAP with parameter settings A and B. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c): parameter setting A, (d): parameter setting B.

According to Figure 4.3 (c), like the partitioning results of STRFCM (Figure 4.1), all GT subclasses were successfully detected except for the subclass with data of mixed types (purple). Several points belonging to the ulva subclass (orange) were assigned to the enteromorpha subclass (red). Both of these subclasses belong to the vegetation class. The performance of the method was not significantly affected by this result. However, in Figure 4.3 (d), there were points belonging to the ulva subclass (orange) that were incorrectly assigned to the turbid water subclass (dark blue). These two subclasses do not belong to the same main class, which resulted in a degradation of performance. The black points in Figure 4.3 (d) represent outliers detected by STRAP.



In table 4.1, it is evident that the choice of parameters significantly impacts the partitioning performance of CODAS. The number of estimated classes varies from 7 to 40, and the number of detected outliers varies from 0 to 282. While CODAS consistently achieves a purity of 100%, it tends to over-partition the dataset and detect numerous outliers under certain parameter settings, which is not desirable. For instance, with the parameter setting of a threshold density of 5 and a threshold radius of 200 (referred as parameter setting B), despite achieving 100% purity, CODAS identified 29 classes, which is more than the 9 GT subclasses, and detected 282 outliers. CODAS gets the best performance under the parameter setting of a threshold density of 3 and a threshold radius of 600 (referred as parameter setting A). Figure 4.4 demonstrates the partitioning results of CODAS with parameter settings A and B.

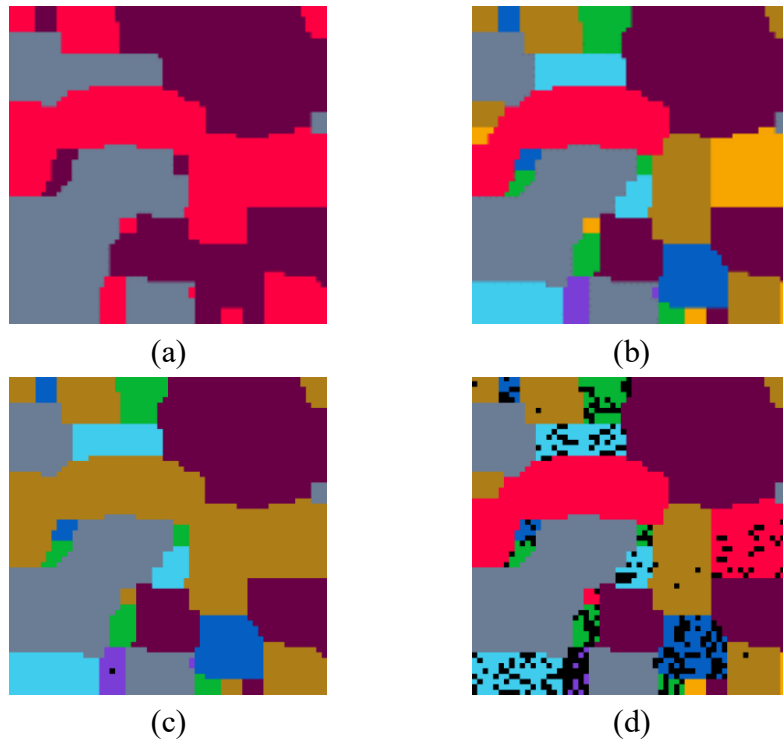


Figure 4.4: Partitioning results on Dataset 1 using CODAS with parameter settings A and B. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c): parameter setting A, (d): parameter setting B.

In Figure 4.4 (c), we observed that all main GT classes were successfully detected. However, the ulva subclass (orange) and the enteromorpha subclass (red) were not detected. In Figure 4.4 (d), we can find that CODAS detected an unusually high number of outliers (282), which is not desirable. These results highlight that the partitioning performance of CODAS is significantly influenced by the choice of parameter values.

In summary, our proposed STRFCM method achieved a purity of 100% on Dataset 1, and the chunk size had no discernible effect on purity and kappa index values. If the parameter values of STRAP and CODAS are well chosen, their highest purity and kappa values can also reach 100%. However, when the parameter values are inappropriate, the partitioning performances of STRAP and CODAS will be significantly affected, which is a great limitation. Furthermore, we observed that the execution time of our proposed method is longer than that of STRAP and CODAS. However, if we consider the time required to tune the parameters to achieve the optimal partition, our method takes less time. In addition, determining the optimal values of parameters often needs the knowledge of datasets in advance, and the parameter values may differ for different datasets, which greatly limits the applicability of methods. In conclusion, our proposed method outperforms the parametric partitioning methods STRAP and CODAS in partitioning Dataset 1.

### 4.3 Evaluation on hyperspectral image of invasive vegetation

In this section, we conduct a comparative analysis between our proposed STRFCM method and two parametric unsupervised methods, STRAP and CODAS, using Dataset 2 as the experimental dataset. Table 4.2 presents the partitioning performances of our method (STRFCM), STRAP, and CODAS on Dataset 2. It only demonstrates a portion of representative partitioning results of STRAP and CODAS.

Table 4.2: Partitioning performances of our method (STRFCM), STRAP, and CODAS on Dataset 2.

Performance of our proposed STRFCM method							
<i>Chunk size</i>	NC	OT	Purity (%)	Kappa (%)	CPU (s)		
256	11	None	<b>96.60</b>	94.41	<b>68.31</b>		
512	<b>10</b>		95.94	93.96	116.68		
1024	10		96.19	<b>94.65</b>	230.04		
2048	10		96.12	94.43	505.71		
Performance of STRAP method							
<i>Input parameters set by user:</i>							
<i>Preference parameter=median, sliding window size=4096 (full image)</i>			NC	OT	Purity (%)	Kappa (%)	CPU (s)

<i>Threshold distance</i>	<i>Initial batch</i>	<i>Damping rate</i>	<i>Max cache</i>					
1000	256	0.8	10	5	5	90.34	75.05	3.32
	512			6	1	83.91	75.05	2.32
	1024			7	0	91.57	89.00	2.80
	2048			9	7	93.94	93.84	4.92
3000	256	0.8	10	6	7	88.64	75.24	0.90
	256		100	10	15	94.59	90.59	0.27
	256		300	9	50	93.82	88.79	0.30
	256		700	10	400	94.15	83.74	0.26
	512		10	7	0	86.70	75.61	0.50
	1024			10	3	94.04	89.06	1.89
	2048			13	7	<b>95.04</b>	94.09	2.57
6000	256	0.6	10	<b>Non-convergence</b>				
	256	0.7		<b>Non-convergence</b>				
	256	0.8		6	5	91.99	75.14	<b>0.21</b>
	512			<b>8</b>	0	88.64	75.92	0.27
	1024			10	0	94.02	89.00	1.15
	2048			13	<b>0</b>	<b>95.04</b>	<b>94.10</b>	2.36
7000	256	0.8	10	6	0	92.01	75.16	0.22
	512			<b>8</b>	0	88.64	75.92	0.25
	1024			10	0	94.02	89.00	1.24
	2048			13	0	<b>95.04</b>	<b>94.10</b>	1.84
<b>Performance of CODAS method</b>								
<i>Input parameters set by user:</i>				<b>NC</b>	<b>OT</b>	<b>Purity (%)</b>	<b>Kappa (%)</b>	<b>CPU (s)</b>
<i>Threshold density</i>	<i>Threshold radius</i>							
4	500		24	87	98.56	83.37	0.97	
	520		22	78	98.33	82.52	1.01	
	540		24	<b>52</b>	98.64	83.88	0.98	
	550		21	<b>52</b>	97.21	50.38	0.94	
6	500		27	234	98.28	86.90	0.99	
	520		23	204	98.52	83.68	1.10	
	540		22	159	<b>99.25</b>	96.74	0.93	
	550		21	161	99.22	96.74	0.98	

8	500	24	408	99.24	<b>97.57</b>	1.00
	520	<b>15</b>	341	96.73	83.17	1.07
	540	17	297	96.78	96.83	1.01
	550	25	296	96.39	96.83	<b>0.92</b>

The partitioning results of STRFCM on Dataset 2 (Table 4.2) show that changing the chunk size has only a minor impact on the partitioning performance. The purity difference is less than 1%, and the kappa difference is also less than 1%. The highest purity and kappa values achieved are 96.60% and 94.65%, respectively. In addition, as the chunk size gradually increases, the execution time gradually becomes longer. The partitioning results with chunk sizes of 256 pixels and 512 pixels, which have the highest and lowest purity, are presented in Figure 4.5.

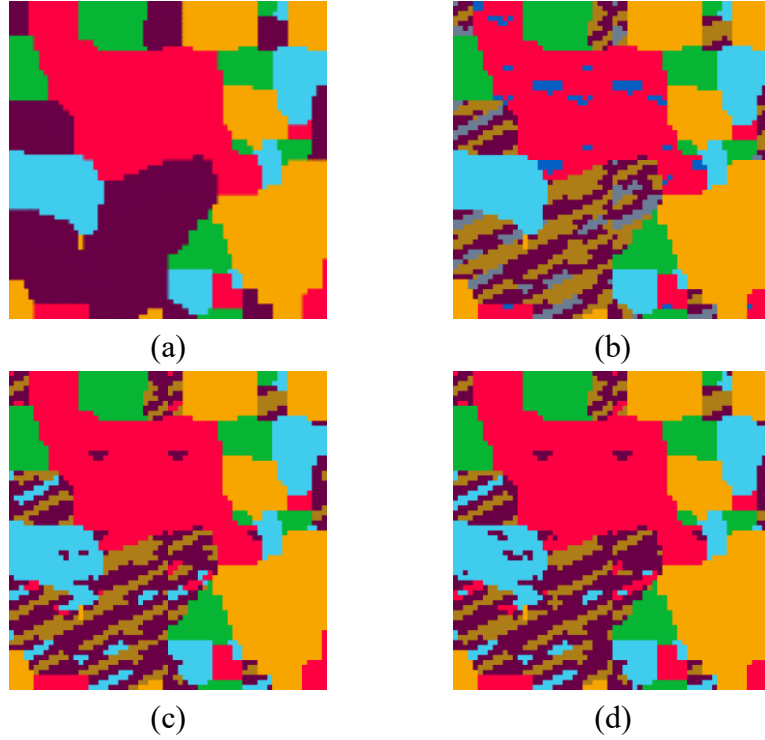


Figure 4.5: Partitioning results on Dataset 2 using our proposed method STRFCM. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c): data chunk size 256 pixels, (d): data chunk size 512 pixels.

In Figure 4.5, we can find that STRFCM detected all main GT classes. However, not all GT subclasses were identified, such as the healthy peach trees subclass (grey) and the sparse pinus halepensis subclass (dark blue). This is because the mean spectral signature of the healthy peach trees subclass (grey) is close to that of the arundo donax subclass (blue), and the mean spectral signature of the sparse pinus halepensis subclass (dark blue) is close to that of the early wilting

peach trees subclass (brown), as shown in Figure 3.3 (b). The algorithm mispartitioned the points in these classes.

Figure 4.6 displays the mean spectral signatures  $\pm$  standard deviation of the final estimated classes of Dataset 2 with the chunk size of 256 pixels. We can observe that the GT healthy peach trees subclass (grey) and the sparse pinus halepensis subclass (dark blue) were not detected. In addition, STRFCM found more classes (11 estimated classes) than the 8 GT subclasses which means it found more detailed information than what is presented in the GT. The estimated classes  $C_1$ ,  $C_{10}$  and  $C_{11}$  belong to the same main GT river class. Similarly, the estimated classes  $C_2$  and  $C_7$  are part of the main GT pinus halepensis class, the estimated classes  $C_3$ ,  $C_4$  and  $C_9$  belong to the same main GT peach trees class, the estimated class  $C_5$  belongs to the main GT arundo donax class, and the estimated classes  $C_6$  and  $C_8$  are part of the main GT buildings class.

According to Table 4.2, we can find that the input parameters have a huge impact on the partitioning performance of STRAP, leading to purity values ranging from 83.91% to 95.04%. Moreover, there are cases where STRAP fails to converge, such as the partitioning result with the parameter setting: threshold distance of 6000, initial batch of 256, damping rate of 0.7, and max cache of 10. The best partitioning performance is obtained with the parameter setting: threshold distance of 6000, initial batch of 2048, damping rate of 0.8, and max cache of 10, and another parameter setting: threshold distance of 7000, initial batch of 2048, damping rate of 0.8, and max cache of 10 (referred as parameter setting A). The worst partitioning performance is found with the parameter setting: threshold distance of 3000, initial batch of 256, damping rate of 0.8, and max cache of 700 (referred as parameter setting B). In this case, although the obtained purity is high (94.15%), STRAP incorrectly identifies an excessive number of objects as outliers, which is not ideal.

Figure 4.7 illustrates the partitioning results of STRAP with parameter settings A and B. In Figure 4.7 (c), we observed that all main GT classes were successfully identified. Similar to the results of STRFCM, it failed to detect the GT healthy peach trees subclass (grey) and the sparse pinus halepensis subclass (dark blue). Some points belonging to the GT healthy peach trees subclass (grey) were incorrectly assigned to the arundo donax subclass (blue). This misassignment negatively impacts the partitioning performance of STRAP. Figure 4.7 (d) demonstrates the poorest partitioning result with numerous points (400 pixels) being identified as outliers (black points). Furthermore, many of the points that belong to the GT arundo donax subclass (blue) were mistakenly assigned to the dense pinus halepensis subclass (red) by

STRAP, which is not desirable. These partitioning results demonstrate that the choice of parameter values has significant influence on the partitioning performance of STRAP.

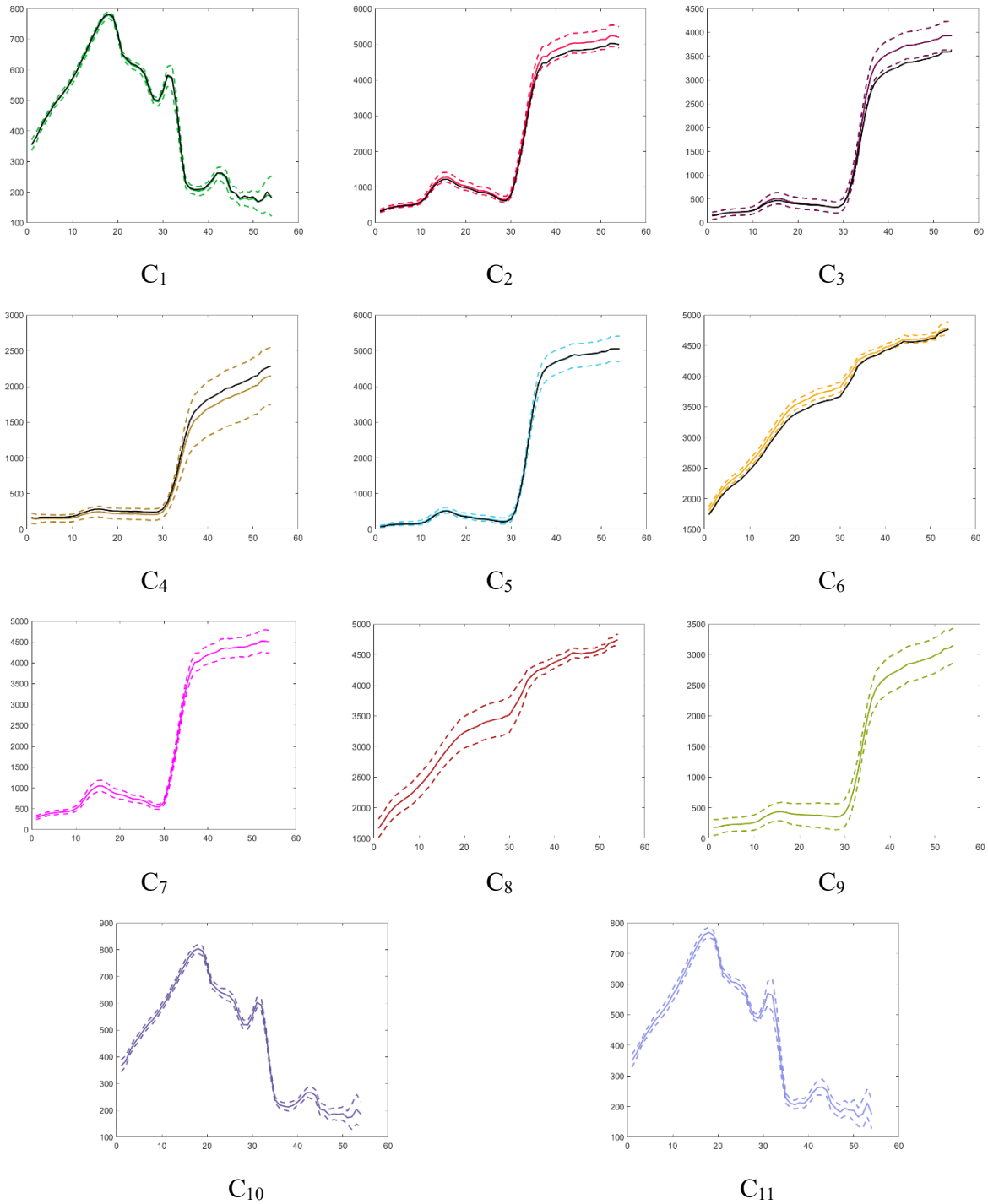


Figure 4.6: Mean spectral signatures  $\pm$  standard deviation of 11 estimated classes of Dataset 2 – data chunk size 256 pixels.

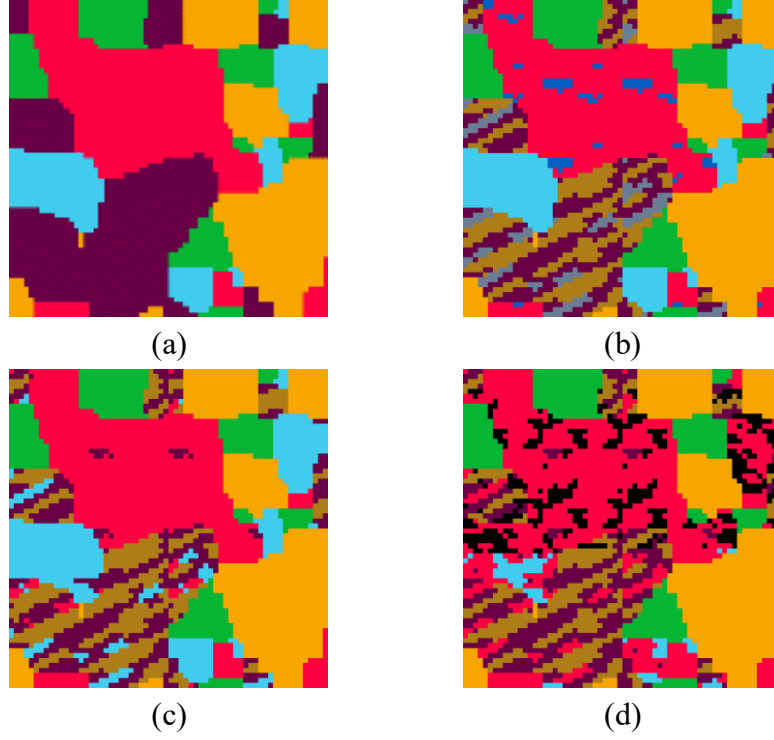


Figure 4.7: Partitioning results on Dataset 2 using STRAP with parameter settings A and B. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c): parameter setting A, (d): parameter setting B.

Table 4.2 also displays the partitioning performance of CODAS under various parameter settings. It becomes evident that the input parameters have a significant impact on the performance of CODAS. Across different parameter settings, CODAS detected varying numbers of classes and outliers. Despite achieving high purity ( $>96\%$ ), CODAS identified a greater number of classes than the 8 GT subclasses and labeled too many points as outliers. Therefore, the partitioning performance of CODAS on Dataset 2 falls short in comparison to that of STRFCM and STRAP. The highest purity (99.25%) is achieved with the parameter setting: threshold density of 6 and threshold radius of 540 (referred as parameter setting A). The lowest purity (96.39%) presented in this table is obtained with the parameter setting: threshold density of 8 and threshold radius of 550 (referred as parameter setting B). Figure 4.8 shows the partitioning results of CODAS with parameter settings A and B.

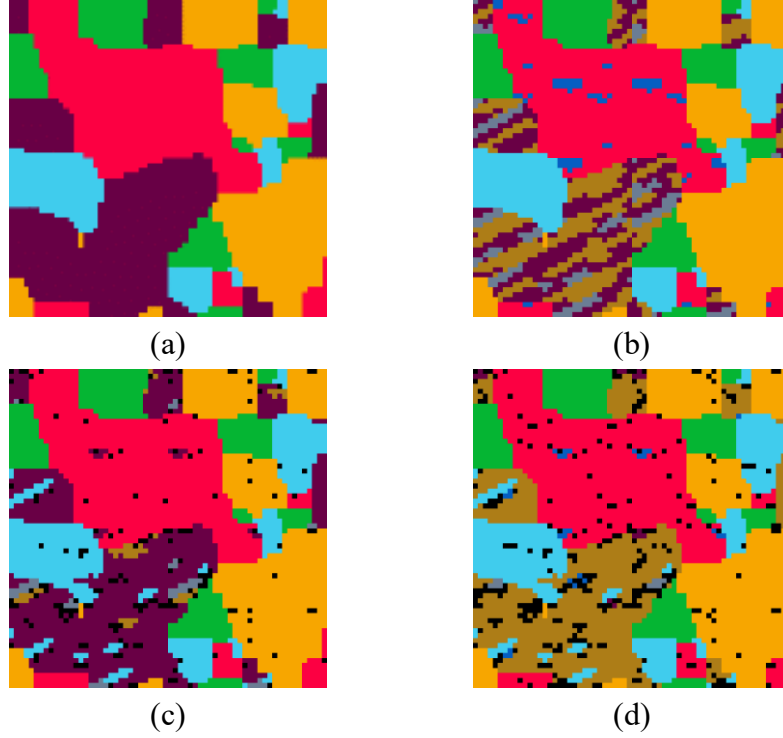


Figure 4.8: Partitioning results on Dataset 2 using CODAS with parameter settings A and B. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c): parameter setting A, (d): parameter setting B.

In Figure 4.8 (c), all main GT classes were identified. Numerous points (159 pixels) were identified as outliers (black points). Similar to results of STRFCM and STRAP, some points belonging to the GT healthy peach trees subclass (grey) were incorrectly assigned to the arundo donax subclass (blue), thus impacting the partitioning performance of CODAS. In Figure 4.8 (d), we can find that more points (296 pixels) were detected as outliers. Additionally, we observed that CODAS partitioned most of the points in the main GT peach trees class (brown) into its subclass, wilting peach trees (earthy yellow). However, according to the GT (Table 3.2), only about half of the points (501 pixels) in peach trees class (1189 pixels) belong to the subclass wilting peach trees.

Comparing with STRAP, the highest purity (96.60%) of our proposed STRFCM method is higher than that of STRAP (95.04%). In addition, the number of classes estimated by STRFCM is closer to the GT (8 GT subclasses), while the number of estimated classes of STRAP and CODAS under the best parameter settings are strongly biased (more than 8 GT subclasses). This is the reason why CODAS achieves a purity as high as 99.25%. Furthermore, we can observe that the choice of parameters can have a significant impact on the final partitioning results of STRAP and CODAS. Even though STRFCM has a longer execution time compared



to STRAP and CODAS, it remains globally more time-efficient, if we consider the time to set the parameters.

In conclusion, the proposed STRFCM method outperforms the STRAP and CODAS methods on hyperspectral image datasets, if we consider the purity, kappa index, number of estimated classes, instability of the results due to the numerous parameters involved, and the time required for parameter setting. The parameter tuning process for STRAP and CODAS is challenging and doesn't guarantee optimization of the solution. Moreover, the process of finding the optimal parameter settings for STRAP and CODAS often relies on prior knowledge of the data stream, which may not be available in some real scenarios.

#### 4.4 Evaluation on Image Segmentation dataset

Our proposed method was also compared with AAPStream [17], DenStream [16], STRAP [14], and CluStream [13] on Dataset 3, which contains 2310 instances. To ensure a fair comparison, we computed the partitioning purity of our method for the same stream lengths as those used in the compared methods (lengths of 500, 1000, 1500, 2000 and 2310), and the parameter settings of these four parametric methods followed their original papers. The performances of these parametric methods are consistent with the results reported in [17]. The results of our method were obtained using a constant data chunk size of 50.

The comparison results are presented in Figure 4.9. Our method consistently achieved the highest purity values across various stream lengths. Its average purity is 92.11%.

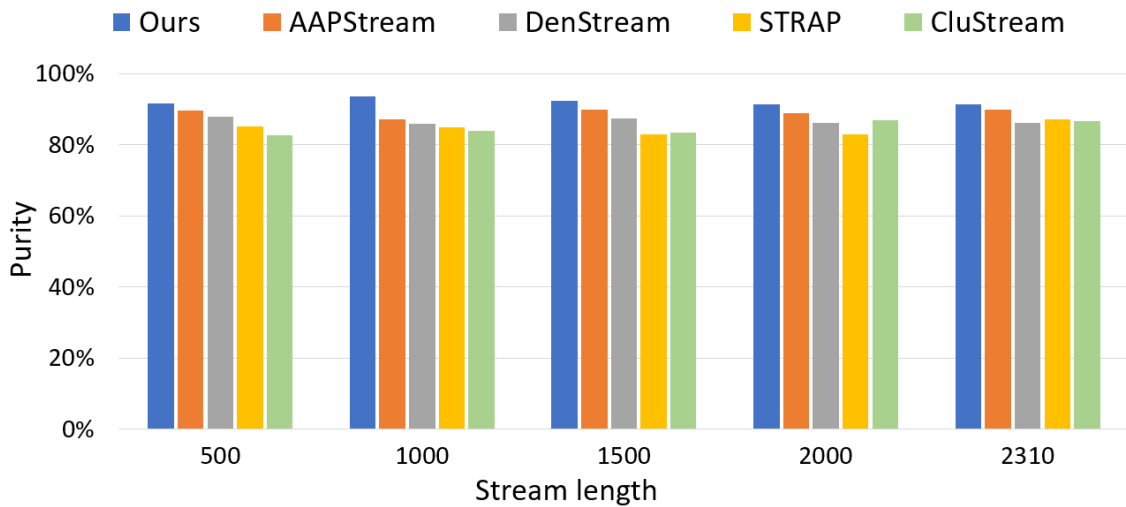


Figure 4.9: Purity criterion of five partitioning methods (STRFCM, AAPStream, DenStream, STRAP, and CluStream).

In conclusion, our proposed STRFCM method outperforms the other four parametric methods, achieving the highest purity across various stream lengths. It has a significant advantage as it does not require the parameter tuning process.

## 4.5 Discussion

Our proposed STRFCM method is an unsupervised and non-parametric method which can automatically partition data streams and estimate the number of classes without the need for prior information or the parameter tuning process. Based on the partitioning results, it is evident that the results of STRFCM remain relatively stable when applied to the partitioning of synthetic hyperspectral images. Variations in data chunk size have a negligible impact on its partitioning performance. Moreover, STRFCM outperforms the parametric partitioning methods STRAP and CODAS on hyperspectral images. In addition, when applied to partition the real-world dataset “Image Segmentation”, our method performs better than AAPStream, DenStream, STRAP and CluStream, achieving higher purity. The evaluation of STRFCM on the synthetic hyperspectral images and the real-world dataset demonstrates its ability to provide high-quality partitioning results.

# Chapter 5

## Conclusion and perspectives

### 5.1 Conclusion

Data stream partitioning is an important technique in data stream processing, enabling the real-time analysis of data streams. Unfortunately, many proposed methods require specifying the number of classes before partitioning and/or introducing user-defined parameters, the values of which may differ for different datasets. The limitation of these parametric methods is that they necessitate empirical parameter tuning process to determine appropriate parameter values to achieve optimal partition. This relies on the user's knowledge of the dataset and can be challenging to implement in real-world applications.

Therefore, an unsupervised and non-parametric data stream partitioning method was proposed to address this limitation. This method does not require predetermining the number of classes or parameter tuning. It only needs streaming data as input to automatically partition the data stream and estimate the optimal number of classes.

Our proposed STRFCM method mainly has two steps: data chunk partitioning step and fusion step. In the data chunk partitioning step, the data stream is divided into equally sized data chunks. STRFCM partitions these data chunks, and identifies the exemplar (centroid) of each class. In the fusion step, STRFCM partitions the exemplar set obtained during the chunk partitioning step to get the final optimal partition.

In order to partition data chunks, we first selected an unsupervised and non-parametric static dataset partitioning method, FCMO, which can automatically partition the dataset without introducing any input parameters. We verified that FCMO using  $L_1$  norm as its similarity criterion outperforms FCMO using  $L_2$  norm. We continued to optimize the FCMO method by selecting an appropriate partitioning validity index employed as the FCMO evaluation criterion. In order to obtain finer classes and retain more detailed information during the chunk partitioning step, a new validity index  $WB-L_{IM}$  based on the  $WB$  index was proposed. For the fusion step, we proposed WFCMO to partition the exemplar set, considering the sizes of the classes represented by exemplars.

STRFCM was evaluated on two synthetic hyperspectral images and a real-world dataset using two external metrics. It was compared with two parametric unsupervised partitioning

methods, STRAP and CODAS, on the hyperspectral images. Additionally, it was evaluated against two parametric unsupervised methods, DenStream and STRAP, and one semi-supervised method, CluStream, as well as one supervised method, AAPStream, on the real-world dataset “Image Segmentation”. The experimental results demonstrate that STRFCM outperforms these parametric methods and achieves high partitioning performance.

In conclusion, a novel unsupervised and non-parametric method called STRFCM was developed for partitioning data streams. This method is easy to apply by users, benefiting from the fact that it eliminates the need for prior information and obviates the requirement for empirical parameter tuning. It demonstrates efficiency in partitioning large, high spatial, and spectral dimensional data streams, especially hyperspectral data streams.

## 5.2 Perspectives

The work presented here opens to both algorithmic and applicative perspectives for further work on our proposed method. The first direction for further research involves completely eliminating the influence of the data chunk size. Our method exhibits good adaptability when partitioning the hyperspectral image data, and the impact of data chunk size on partitioning performance exhibits only marginal variations. However, this variation only renders the results relatively stable. Therefore, efforts should be focused on finding a solution that completely eliminates the influence of data chunk size on partitioning performance, ensuring consistently stable results for all types of data.

According to the assessment results, our proposed STRFCM method is well-suited for partitioning synthetic hyperspectral image data streams. An interesting future direction is to assess our method on real and large hyperspectral images with valid ground truth.

Another direction for further research considers reducing the execution time. We observed that as the chunk size increases, the algorithm’s execution time becomes longer, especially when partitioning very large data chunks. It is important to emphasize that, throughout our research, our primary focus has consistently been on ensuring the high-quality results of our method, rather than the time it consumes. To address the issue of execution times, one potential optimization strategy involves harnessing the power of higher-performance processors equipped with advanced parallel processing techniques. For instance, our current setup employs a 4-core processor. However, in the future, we can utilize an 8 or 16-core processor to process more partitioning tasks in parallel. This approach will enhance the efficiency of our method without compromising result quality.

# Bibliography

- [1] Stuart, M.B., McGonigle, A.J., Willmott, J.R.: Hyperspectral imaging in environmental monitoring: A review of recent developments and technological advances in compact field deployable systems. *Sensors*, 19(14), 3071 (2019).
- [2] Yang, B., Yang, M., Plaza, A., Gao, L., Zhang, B.: Dual-mode FPGA implementation of target and anomaly detection algorithms for real-time hyperspectral imaging. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6), 2950-2961 (2015).
- [3] Bindhu, V., Ranganathan, G.: Hyperspectral image processing in internet of things model using clustering algorithm. *Journal of ISMAC*, 3(02), 163-175 (2021).
- [4] Wan, L., Li, H., Li, C., Wang, A., Yang, Y., Wang, P.: Hyperspectral sensing of plant diseases: Principle and methods. *Agronomy*, 12(6), 1451 (2022).
- [5] Horstrand, P., Guerra, R., Rodríguez, A., Díaz, M., López, S., López, J.F.: A UAV platform based on a hyperspectral sensor for image capturing and on-board processing. *IEEE Access*, 7, 66919-66938 (2019).
- [6] Zhong, Y., Wang, X., Xu, Y., Jia, T., Cui, S., Wei, L., Ma, A., Zhang, L.: MINI-UAV borne hyperspectral remote sensing: A review. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 5908-5911 (2017).
- [7] Xiang, T., Xia, G., Zhang, L.: Mini-unmanned aerial vehicle-based remote sensing: Techniques, applications, and prospect. *IEEE Geoscience and Remote Sensing Magazine*, 7(3), 29-63 (2019).
- [8] Saha, D., Manickavasagan, A.: Machine learning techniques for analysis of hyperspectral images to determine quality of food products: A review. *Current Research in Food Science*, 4, 28-44 (2021).
- [9] Fabelo, H., Ortega, S., Kabwama, S., Callico, G.M., Bulters, D., Szolna, A., Pineiro, J.F., Sarmiento, R.: HELICoiD project: A new use of hyperspectral imaging for brain cancer detection in real-time during neurosurgical operations. In *Hyperspectral Imaging Sensors: Innovative Applications and Sensor Standards 2016*, 9860, 986002 (2016).
- [10] Zubaroğlu, A., Atalay, V.: Data stream clustering: A review. *Artificial Intelligence Review*, 54(2), 1201-1236 (2021).
- [11] Chehdi, K., Taher, A., Cariou, C.: Stable and unsupervised fuzzy C-means method and its validation in the context of multicomponent images. *Journal of Electronic Imaging*, 24(6), 061117 (2015).
- [12] Bezdek, J.C., Ehrlich, R., Full, W.: FCM: The fuzzy c-means clustering algorithm.

*Computers and Geosciences*, 10(2-3), 191-203 (1984).

- [13] Aggarwal, C.C., Philip, S.Y., Han, J., Wang, J.: A framework for clustering evolving data streams. In *Proceedings 2003 VLDB Conference*, 81-92 (2003).
- [14] Zhang, X., Furtlehner, C., Germain-Renaud, C., Sebag, M.: Data stream clustering with affinity propagation. *IEEE Transactions on Knowledge and Data Engineering*, 26(7), 1644-1656 (2013).
- [15] Hyde, R., Angelov, P.: A new online clustering approach for data in arbitrary shaped clusters. In *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, 228-233 (2015).
- [16] Cao, F., Estert, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, 328-339 (2006).
- [17] Abdulah, S., Atwa, W., Abdelmoniem, A.M.: Active clustering data streams with affinity propagation. *ICT Express*, 8(2), 276-282 (2022).
- [18] Ray, P.: A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3), 291-319 (2018).
- [19] Jana, D., Patil, J., Herkal, S., Nagarajaiah, S., Duenas-Osorio, L.: CNN and Convolutional Autoencoder (CAE) based real-time sensor fault detection, localization, and correction. *Mechanical Systems and Signal Processing*, 169, 108723 (2022).
- [20] Zhang, C., Jia, D., Wang, L., Wang, W., Liu, F., Yang, A.: Comparative research on network intrusion detection methods based on machine learning. *Computers & Security*, 121, 102861 (2022).
- [21] Shahraki, A., Taherkordi, A., Haugen, Ø., Eliassen, F.: Clustering objectives in wireless sensor networks: A survey and research direction analysis. *Computer Networks*, 180, 107376 (2020).
- [22] Nithya, N., Duraiswamy, K., Gomathy, P.: A survey on clustering techniques in medical diagnosis. *International Journal of Computer Science Trends and Technology (IJCTST)*, 1(2), 17-23 (2013).
- [23] Jones, A.K., Sielken, R.S.: Computer system intrusion detection: A survey. *Computer Science Technical Report*, 1-25 (2000).
- [24] Bolton, R.J., Hand, D.J.: Statistical fraud detection: A review. *Statistical Science*, 17(3), 235-255 (2002).
- [25] Feather, F., Siewiorek, D., Maxion, R.: Fault detection in an ethernet network using anomaly signature matching. *ACM SIGCOMM Computer Communication Review*,

23(4), 279-288 (1993).

- [26] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1-22 (1977).
- [27] Cortes, C., Vapnik, V.: Support-vector networks. *Machine Learning*, 20, 273-297 (1995).
- [28] Han, J., Pei, J., Tong, H.: Data mining: Concepts and techniques. *Elsevier* (2022).
- [29] Kim, T., Chen, I.R., Lin, Y., Wang, A.Y.Y., Yang, J.Y.H., Yang, P.: Impact of similarity metrics on single-cell RNA-seq data clustering. *Briefings in Bioinformatics*, 20(6), 2316-2326 (2019).
- [30] Lee, L.J.: Similarity-based approaches to natural language processing. *Harvard University* (1997).
- [31] Cvejic, N., Loza, A., Bull, D., Canagarajah, N.: A similarity metric for assessment of image fusion algorithms. *International Journal of Signal Processing*, 2(3), 178-182 (2005).
- [32] Singh, A., Yadav, A., Rana, A.: K-means with three different distance metrics. *International Journal of Computer Applications*, 67(10), 13-17 (2013).
- [33] Popat, S.K., Deshmukh, P.B., Metre, V.A.: Hierarchical document clustering based on cosine similarity measure. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, 153-159 (2017).
- [34] Zhang, P., Wang, X., Song, P.X.K.: Clustering categorical data based on distance vectors. *Journal of the American Statistical Association*, 101(473), 355-367 (2006).
- [35] Bisandu, D.B., Prasad, R., Liman, M.M.: Data clustering using efficient similarity measures. *Journal of Statistics and Management Systems*, 22(5), 901-922 (2019).
- [36] Pawar, S., Gururaj, H.M., Chiplunar, N.N.: Text summarization using document and sentence clustering. *Procedia Computer Science*, 215, 361-369 (2022).
- [37] Jain, A.K.: Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8), 651-666 (2010).
- [38] Jain, A.K., Dubes, R.C.: Algorithms for clustering data. *Prentice-Hall* (1988).
- [39] He, Z., Deng, S., Xu, X.: Improving k-modes algorithm considering frequencies of attribute values in mode. *Computational Intelligence and Security: International Conference*, 157-162 (2005).

- [40] Rosenberger, C., Chehdi, K.: Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation. In *Proceedings of 15th International Conference on Pattern Recognition*, 1, 656-659 (2000).
- [41] Linde, Y., Buzo, A., Gray, R.: An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1), 84-95 (1980).
- [42] Frey, B.J., Dueck, D.: Clustering by passing messages between data points. *Science*, 315(5814), 972-976 (2007).
- [43] Chehdi, K., Soltani, M., Cariou, C.: Pixel classification of large-size hyperspectral images by affinity propagation. *Journal of Applied Remote Sensing*, 8(1), 083567 (2014).
- [44] Levine, M.D., Nazif, A.M.: Dynamic measurement of computer generated image segmentations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2), 155-164 (1985).
- [45] Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603-619 (2002).
- [46] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 96(34), 226-231 (1996).
- [47] Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2, 169-194 (1998).
- [48] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2), 49-60 (1999).
- [49] Zhang, X.: Contributions to large scale data clustering and streaming with affinity propagation. Application to autonomic grids. PhD thesis. *PARIS: University PARIS-SUD* (2010).
- [50] Sheikholeslami, G., Chatterjee, S., Zhang, A.: Wavecluster: A multi-resolution clustering approach for very large spatial databases. *VLDB*, 98, 428-439 (1998).
- [51] Murtagh, F., Contreras, P.: Methods of hierarchical clustering. *CSIR*, 1, 1-21 (2011).
- [52] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 94-105 (1998).
- [53] Mansalis, S., Ntoutsis, E., Pelekis, N., Theodoridis, Y.: An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining: The ASA Data Science*



*Journal*, 11(4), 167-187 (2018).

- [54] Carnein, M., Trautmann, H.: Optimizing data stream representation: An extensive survey on stream clustering algorithms. *Business & Information Systems Engineering*, 61, 277-297 (2019).
- [55] Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., Carvalho, A.C.D., Gama, J.: Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1), 1-31 (2013).
- [56] Ghesmoune, M., Lebbah, M., Azzag, H.: State-of-the-art on clustering data streams. *Big Data Analytics*, 1, 1-27 (2016).
- [57] Barbakh, W., Fyfe, C.: Online clustering algorithms. *International Journal of Neural Systems*, 18(3), 185-194 (2008).
- [58] Guha, S., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 359-366 (2000).
- [59] Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: Streamkm++ a clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17, 2-4 (2012).
- [60] Arthur, D., Vassilvitskii, S.: K-means++ the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027-1035 (2007).
- [61] Zhou, A., Cao, F., Qian, W., Jin, C.: Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15, 181-214 (2008).
- [62] Ren, J., Ma, R.: Density-based data streams clustering over sliding windows. In *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, 5, 248-252 (2009).
- [63] Liu, L.X., Huang, H., Guo, Y.F., Chen, F.C.: rDenStream, a clustering algorithm over an evolving data stream. In *2009 International Conference on Information Engineering and Computer Science*, 1-4 (2009).
- [64] Lin, J., Lin, H.: A density-based clustering over evolving heterogeneous data stream. In *2009 ISECS International Colloquium on Computing, Communication, Control, and Management*, 4, 275-277 (2009).
- [65] Hassani, M., Spaus, P., Gaber, M.M., Seidl, T.: Density-based projected clustering of data streams. In *Scalable Uncertainty Management: 6th International Conference*, 311-324 (2012).
- [66] Amini, A., Saboohi, H., Herawan, T., Wah, T.Y.: MuDi-Stream: A multi density

- clustering algorithm for evolving data stream. *Journal of Network and Computer Applications*, 59, 370-385 (2016).
- [67] Gong, S., Zhang, Y., Yu, G.: Clustering stream data by exploring the evolution of density mountain. In *Proceedings of the VLDB Endowment*, 11(4), 393-405 (2017).
- [68] Al Abd Alazeez, A., Jassim, S., Du, H.: EDDS: An enhanced density-based method for clustering data streams. In *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*, 103-112 (2017).
- [69] Fahy, C., Yang, S.: Finding and tracking multi-density clusters in online dynamic data streams. *IEEE Transactions on Big Data*, 8(1), 178-192 (2019).
- [70] Yan, X., Razeghi-Jahromi, M., Homaifar, A., Erol, B.A., Girma, A., Tunstel, E.: A novel streaming data clustering algorithm based on fitness proportionate sharing. *IEEE Access*, 7, 184985-185000 (2019).
- [71] Chen, Y., Tu, L.: Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 133-142 (2007).
- [72] Amini, A., Wah, T.Y., Teh, Y.W.: DENGRIS-Stream: A density-grid based clustering algorithm for evolving data streams over sliding window. In *Proceedings of the International Conference on Data Mining and Computer Engineering*, 206-210 (2012).
- [73] Jia, C., Tan, C., Yong, A.: A grid and density-based clustering algorithm for processing data stream. In *2008 Second International Conference on Genetic and Evolutionary Computing*, 517-521 (2008).
- [74] Wan, L., Ng, W.K., Dang, X.H., Yu, P.S., Zhang, K.: Density-based clustering of data streams at multiple resolutions. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3), 1-28 (2009).
- [75] Tareq, M., Sundararajan, E.A., Mohd, M., Sani, N.S.: Online clustering of evolving data streams using a density grid-based method. *IEEE Access*, 8, 166472-166490 (2020).
- [76] Xu, R., Wunsch, D.: Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3), 645-678 (2005).
- [77] Everitt, B.S., Landau, S., Leese, M., Stahl, D.: Cluster analysis. *John Wiley & Sons* (2011).
- [78] Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of internal clustering validation measures. In *2010 IEEE International Conference on Data Mining*, 911-916 (2010).
- [79] Da Silva, L.E.B., Melton, N.M., Wunsch, D.C.: Incremental cluster validity indices for

online learning of hard partitions: Extensions and comparative study. *IEEE Access*, 8, 22025-22047 (2020).

- [80] Sharma, S.: Applied multivariate techniques. *John Wiley & Sons* (1995).
- [81] Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. *Journal of Intelligent Information Systems*, 17, 107-145 (2001).
- [82] Hubert, L., Arabie, P.: Comparing partitions. *Journal of Classification*, 2, 193-218 (1985).
- [83] Zhao, Q., Xu, M., Fränti, P.: Sum-of-squares based cluster validity index and significance analysis. In *International Conference on Adaptive and Natural Computing Algorithms*, 313-322 (2009).
- [84] Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1), 1-27 (1974).
- [85] Maulik, U., Bandyopadhyay, S.: Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12), 1650-1654 (2002).
- [86] Pakhira, M.K., Bandyopadhyay, S., Maulik, U.: Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3), 487-501 (2004).
- [87] Bandyopadhyay, S., Maulik, U.: Nonparametric genetic clustering: Comparison of validity indices. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(1), 120-125 (2001).
- [88] Dunn, J.C.: Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1), 95-104 (1974).
- [89] Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53-65 (1987).
- [90] Yang, M.S., Wu, K.L.: A new validity index for fuzzy clustering. In *10th IEEE International Conference on Fuzzy Systems*, 1, 89-92 (2001).
- [91] Lughofer, E.: Extensions of vector quantization for incremental clustering. *Pattern Recognition*, 41(3), 995-1011 (2008).
- [92] Davies, D.L., Bouldin, D.W.: A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2, 224-227 (1979).
- [93] Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(08), 841-847 (1991).

- [94] Halkidi, M., Vazirgiannis, M., Batistakis, Y.: Quality scheme assessment in the clustering process. In *Principles of Data Mining and Knowledge Discovery: 4th European Conference*, 265-276 (2000).
- [95] Ball, G.H., Hall, D.J.: ISODATA, a novel method of data analysis and classification. Technical report. *Stanford University* (1965).
- [96] UC Irvine Machine Learning Repository, <https://archive.ics.uci.edu>, last accessed 2023/12/21.
- [97] Cohen, J.: A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37-46 (1960).

## List of figures

Figure 2.1: Partitioning of objects in three classes. To each class is associated its centroid or exemplar (*).	16
Figure 2.2: AP message passing process [42]. (a): availability and responsibility messages, (b): iterations of message passing.	24
Figure 2.3: DBSCAN [46]. (a): arbitrary-shaped classes, (b): point $p$ and point $q$ are density-connected.	26
Figure 2.4: Grid-based partitioning: imposing grids on data space [49].	26
Figure 2.5: Diagram of CluStream algorithm.	30
Figure 2.6: Diagram of STRAP algorithm [14].	32
Figure 2.7: Illustration of micro-cluster regions showing (a): micro-cluster radius in red and micro-cluster core radius in green, (b): micro-clusters combined to the global clusters [15].	36
Figure 3.1: Flow chart of the proposed unsupervised partitioning method.	45
Figure 3.2: Mean spectral signatures of the GT classes in Dataset 1. (a): 3 main GT classes, (b): 9 GT subclasses.	54
Figure 3.3: Mean spectral signatures of the GT classes in Dataset 2. (a): 5 main GT classes, (b): 8 GT subclasses.	54
Figure 3.4: Partitioning results on Dataset 1. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c), (d) and (e): FCMO using criteria $L_1-F$ , $L_2-F$ and $L_1-WB$ , respectively, (f): AP – preference parameter set to minimum and damping rate of 0.9.	57
Figure 3.5: Partitioning results on Dataset 2. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c), (d) and (e): FCMO using criteria $L_1-F$ , $L_2-F$ and $L_1-WB$ , respectively, (f): AP – preference parameter set to minimum and damping rate of 0.9.	59
Figure 3.6: The sizes of estimated classes on data chunk of 256 pixels. (a): FCMO- $L_1-WB$ , (b): FCMO- $L_1-WBM$ .	61
Figure 3.7: The sizes of estimated classes on data chunk of 512 pixels. (a): FCMO- $L_1-WB$ , (b): FCMO- $L_1-WBM$ .	61
Figure 3.8: Partitioning results of WFCMO and FCMO on the generated dataset.	63
Figure 4.1: Partitioning results on Dataset 1 using our proposed method STRFCM. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c): data chunk size 240 pixels, (d): data chunk size 1800 pixels.	68
Figure 4.2: Mean spectral signatures $\pm$ standard deviation of 8 estimated classes of Dataset 1 – data chunk size 240 pixels.	69
Figure 4.3: Partitioning results on Dataset 1 using STRAP with parameter settings A and B. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c): parameter setting A, (d): parameter setting B.	70
Figure 4.4: Partitioning results on Dataset 1 using CODAS with parameter settings A and B. (a): GT1 image (3 main GT classes), (b): GT2 image (9 GT subclasses), (c): parameter setting A, (d): parameter setting B.	71

Figure 4.5: Partitioning results on Dataset 2 using our proposed method STRFCM. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c): data chunk size 256 pixels, (d): data chunk size 512 pixels. ....	74
Figure 4.6: Mean spectral signatures $\pm$ standard deviation of 11 estimated classes of Dataset 2 – data chunk size 256 pixels. ....	76
Figure 4.7: Partitioning results on Dataset 2 using STRAP with parameter settings A and B. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c): parameter setting A, (d): parameter setting B. ....	77
Figure 4.8: Partitioning results on Dataset 2 using CODAS with parameter settings A and B. (a): GT1 image (5 main GT classes), (b): GT2 image (8 GT subclasses), (c): parameter setting A, (d): parameter setting B. ....	78
Figure 4.9: Purity criterion of five partitioning methods (STRFCM, AAPStream, DenStream, STRAP, and CluStream).....	79

# List of tables

Table 2.1: Common partitioning validity indices.....	41
Table 3.1: Dataset 1 – hyperspectral image of algae, GT images and GT class details. ....	52
Table 3.2: Dataset 2 – hyperspectral image of invasive vegetation, GT images and GT class details.....	53
Table 3.3: Partitioning performances of FCMO using different similarity criteria and evaluation criteria on Dataset 1.....	55
Table 3.4: Partitioning performance of the AP method on Dataset 1. ....	55
Table 3.5: Partitioning performances of FCMO using different similarity criteria and evaluation criteria on Dataset 2.....	58
Table 3.6: Partitioning performance of the AP method on Dataset 2. ....	58
Table 3.7: Partitioning performances of FCMO- $L_I$ - $F$ and FCMO- $L_I$ - $WB$ on different sizes of data chunks in Dataset 2. ....	60
Table 3.8: Details of the two main GT classes and seven GT subclasses in the generated dataset. ....	62
Table 4.1: Partitioning performances of our method (STRFCM), STRAP, and CODAS on Dataset 1. ....	66
Table 4.2: Partitioning performances of our method (STRFCM), STRAP, and CODAS on Dataset 2. ....	72

---

**Titre :** Partitionnement de grands flux de données d'images hyperspectrales.

**Mots clés :** flux de données, partitionnement non supervisé, estimation, images hyperspectrales, optimisation.

**Résumé :** Avec le développement de systèmes de prise de décision automatisés et optimisés, le partitionnement de grands flux de données, qui ne dépend pas d'échantillons d'apprentissage, attire de plus en plus l'attention. Dans l'état de l'art, la majorité des méthodes de partitionnement de flux de données sont paramétriques, ce qui nécessite la spécification d'un ou plusieurs paramètres définis par l'utilisateur et/ou du nombre de classes avant le processus de partitionnement. Dans les applications pratiques, obtenir des connaissances *a priori* sur l'ensemble de données et déterminer les valeurs de paramètres optimales à l'avance est un défi. Par conséquent, notre recherche se concentre sur le développement d'une méthode non supervisée et non paramétrique facile à utiliser par les utilisateurs, bénéficiant du fait qu'elle élimine le besoin de connaissances *a priori* et supprime la nécessité de régler les paramètres de manière empirique. La méthode développée peut estimer de manière automatique le nombre de classes et partitionner le flux de données. Elle est efficace pour partitionner un flux de données de grandes tailles spatiale et spectrale, en particulier les flux de données hyperspectraux. La méthode proposée a été évaluée sur des bases de données réelles et synthétiques. Selon plusieurs critères d'évaluation objectifs, elle surpasse les cinq méthodes de partitionnement de flux de données comparées (trois méthodes paramétriques non supervisées, une méthode semi-supervisée et une méthode supervisée utilisant l'apprentissage actif).

---

**Title:** Partitioning of large hyperspectral image data streams.

**Keywords:** data stream, unsupervised partitioning, estimation, hyperspectral images, optimization.

**Abstract:** With the development of automated and optimized decision-making systems, large data stream partitioning, which does not rely on training samples, has attracted more and more attention. In the state-of-the-art, a majority of data stream partitioning methods are parametric which require the specification of one or more user-defined parameters and/or the number of classes before the partitioning process. In practical applications, obtaining prior information about the dataset and determining optimal parameter values in advance can be challenging. Therefore, our research focuses on the development of an unsupervised and non-parametric method which is easy to apply by users, benefiting from the fact that it eliminates the need for prior information and obviates the requirement for empirical parameter tuning. The developed method can automatically estimate the number of classes and partition the data stream. It is efficient to partition large and high spatial and spectral dimensional data streams, especially hyperspectral data streams. Our proposed method was assessed on real-world and synthetic databases. According to several objective evaluation criteria, it outperforms the five compared data stream partitioning methods (three parametric unsupervised methods, one semi-supervised method and one supervised method using active learning).